Bu kitaba sığmayan daha neler var!

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

Kişiselleştirilmiş Öğrenme ve Raporlama

Canlı Ders

Zengin İçerik

Sosyal Etkileşim

Puan ve Armalar

EBA Portfolyo

www.eba.gov.tr

ISBN: 978-975-11-5675-4

Google play

Download on the App Store

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

BİLİŞİM TEKNOLOJİLERİ ALANI PROGRAMLAMA TEMELLERİ 9 DERS KİTABI

MESLEKİ ve TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

PROGRAMLAMA TEMELLERİ 9

DERS KİTABI

T.C. MİLLÎ EĞİTİM BAKANLIĞI

MESLEKİ VE TEKNİK ANADOLU LİSESİ
BİLİŞİM TEKNOLOJİLERİ ALANI

PROGRAMLAMA TEMELLERİ 9

DERS KİTABI

Yazarlar

Dr. Selçuk Yusuf Arslan
Ahmet VURAL
Devrim ALTINKURT
Özgü ASKER

DEVLET KİTAPLARI

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 7533
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 1573

Dil Uzmanı
Mesut ÖZDEMİR
Osman Nuri GÜVEN

Program Geliştirme Uzmanı
Ahmet ALİŞ

Ölçme ve Değrlendirme Uzmanı
Aydemir KELEŞ

Görsel Tasarım Uzmanı
Özden ALTUN
Serkan KOCABAŞ

Grafik Tasarım Uzmanı
Gözde Yıldırım EVCİ
Hasan Basri YILMAZ

ISBN: 978-975-11-5675-4

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

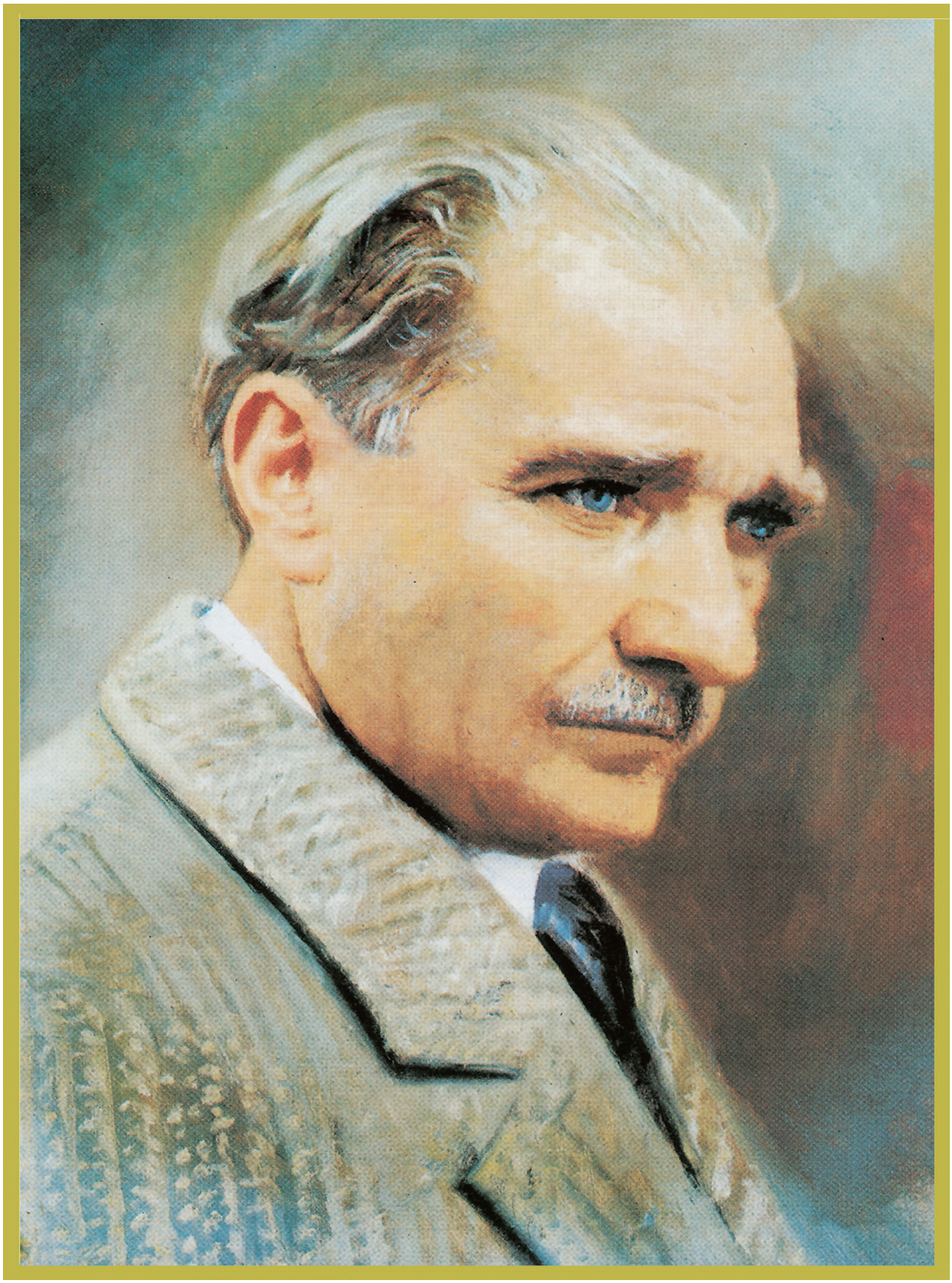
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

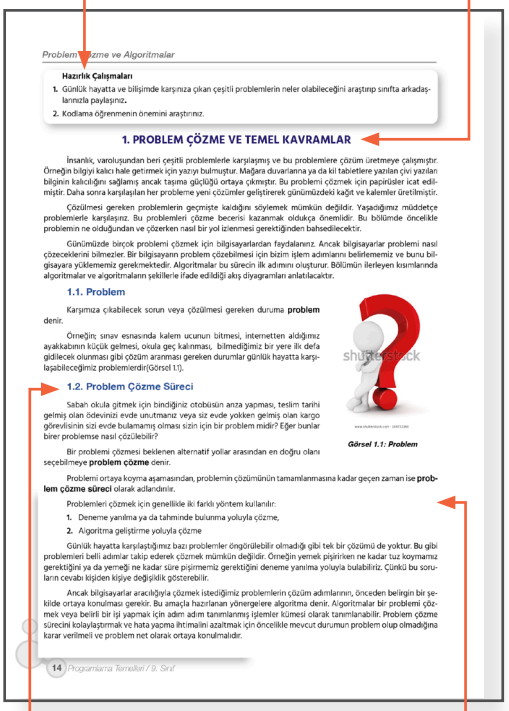
Ünite adını gösterir.

Öğrenme birimi numarasını gösterir.

Karekod okuyucu ile taratarak resim, video, animasyon, soru ve çözümleri vb. ilave kaynaklara ulaşabileceğiniz karekod.
Detaylı bilgi için http://kitap.eba.gov.tr/karekod

Derse başlamadan yapılacak olan hazırlıkları gösterir.

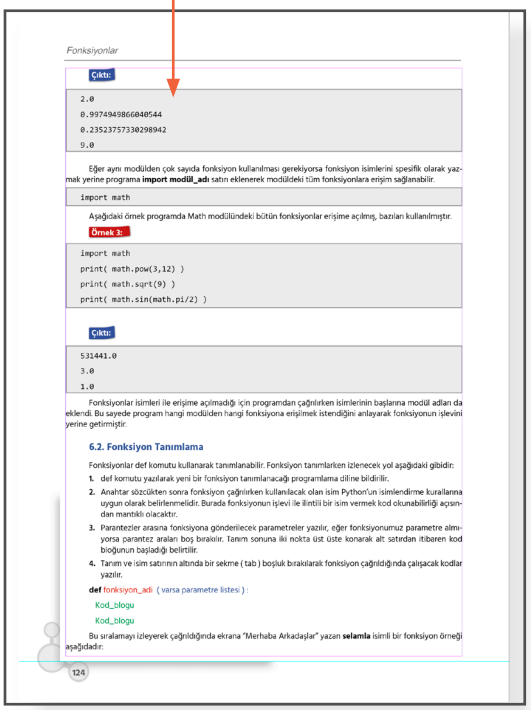
Konu başlığını gösterir.

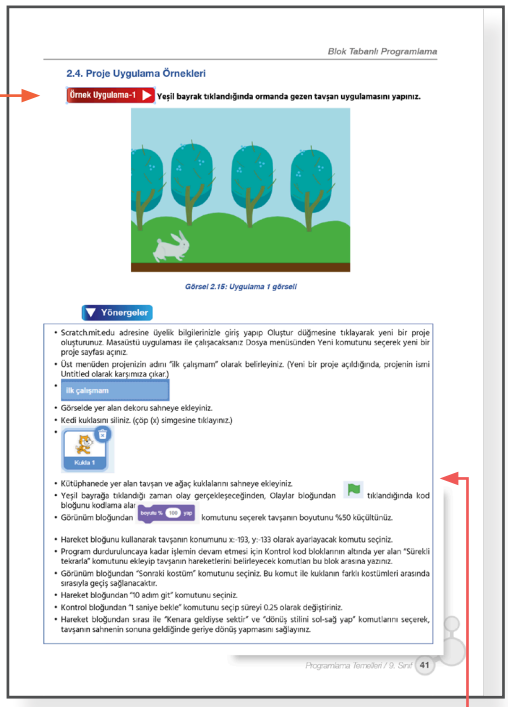


Alt konu başlıklarını gösterir.

Konu anlatımını gösterir.

Programlama kodlarını gösterir.





Yapılacak örnek uygulamaları gösterir.

Örnek uygulamaların yönergesini gösterir.

ÖĞRENME BİRİMİ 1

PROBLEM ÇÖZME VE ALGORİTMALAR

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Problem çözmenin önemini kavrayacak,
- Problem çözme kavramlarını bilecek,
- Probleme çözüm üretirken problem çözme aşamasındaki adımları izleyebilecek,
- Problem çözmede kullanılabilecek aritmetiksel ve mantıksal operatörler ile karar operatörlerini listeleyebilecek,
- İşlem önceliğini kavrayabilecek,
- Problem çözmek için algoritmaları kullanabilecek,
- Doğrusal, döngüsel ve mantıksal algoritmalar hazırlayabilecek,
- Problem çözmede algoritma hatalarını bulup düzeltebilecek,
- Problemin çözümü için gerekli algoritma ve akış şemalarını doğrusal, döngüsel yapıları ve karar mantık yapılarını kullanarak hazırlayabileceksiniz.

Anahtar Kelimeler:

Problem, problem çözme, algoritma, operatör, değişken, mantıksal algoritma, döngüsel algoritma, algoritma hatası, akış şeması.

Hazırlık Çalışmaları

1. Günlük hayatta ve bilişimde karşınıza çıkan çeşitli problemlerin neler olabileceğini araştırıp sınıfta arkadaşlarınızla paylaşınız.
2. Kodlama öğrenmenin önemini araştırınız.

1. PROBLEM ÇÖZME VE ALGORİTMALAR

1.1. Problem Çözme ve Temel Kavramlar

İnsanlık, varoluşundan beri çeşitli problemlerle karşılaşmış ve bu problemlere çözüm üretmeye çalışmıştır. Örneğin bilgiyi kalıcı hâle getirmek için yazıyı bulmuştur. Mağara duvarlarına ya da kil tabletlere yazılan çivi yazıları bilginin kalıcılığını sağlamış ancak taşıma güçlüğü ortaya çıkmıştır. Bu problemi çözmek için papirüsler[1] icat edilmiştir. Daha sonra karşılaşılan her probleme yeni çözümler geliştirilerek günümüzdeki kâğıt ve kalemler üretilmiştir.

Çözülmesi gereken problemlerin geçmişte kaldığını söylemek mümkün değildir. İnsanoğlu yaşam boyu çeşitli problemlerle karşılaşır. Bu problemleri çözme becerisi kazanmak oldukça önemlidir. Bu bölümde öncelikle problemin ne olduğundan ve problemi çözerken nasıl bir yol izlenmesi gerektiğinden bahsedilecektir.

Günümüzde birçok problemi çözmek için bilgisayarlardan faydalanılır. Ancak bilgisayarlar problemi nasıl çözeceğini bilmez. Bir bilgisayarın problem çözebilmesi için işlem adımlarının belirlenmesi ve bunun bilgisayara yüklenmesi gerekmektedir. Algoritmalar bu sürecin ilk adımını oluşturur. Bölümün ilerleyen kısımlarında algoritmalar ve algoritmaların şekillerle ifade edildiği akış diyagramları anlatılacaktır.

1.1.1. Problem

Karşılaşılabilecek soruna veya çözülmesi gereken duruma problem denir. Örneğin; sınav esnasında kalem ucunun bitmesi, internetten alınan ayakkabının küçük gelmesi, okula geç kalınması, bilmediğimiz bir yere ilk defa gidilecek olunması gibi çözüm aranması gereken durumlar günlük hayatta karşılaşılabilecek problemlerdir (Görsel 1.1).

Görsel 1.1: Problem

1.1.2. Problem Çözme Süreci

Sabah okula gitmek için bindiğiniz otobüsün arıza yapması, teslim tarihi gelmiş olan ödevinizi evde unutmanız veya siz evde değilken gelen kargo görevlisinin sizi evde bulamamış olması bir problem midir? Sizin için bunlar birer problemse nasıl çözülebilir?

Bir problemi çözmesi beklenen alternatif yollar arasından en doğru olanı seçebilmeye problem çözme denir.

Problemi ortaya koyma aşamasından, problemin çözümünün tamamlanmasına kadar geçen zaman ise problem çözme süreci olarak adlandırılır.

Problemleri çözmek için genellikle iki farklı yöntem kullanılır:

1. Deneme yanılma ya da tahminde bulunma yoluyla çözme
2. Algoritma geliştirme yoluyla çözme

Günlük hayatta karşılaşılan bazı problemler öngörülebilir olmadığı gibi bu problemlerin tek bir çözümü de yoktur. Bu gibi problemleri belli adımlar takip ederek çözmek mümkün değildir. Örneğin yemek pişirilirken ne kadar tuz koyulması gerektiği ya da yemeğin ne kadar süre pişirilmesi gerektiği deneme yanılma yoluyla bulunabilir. Çünkü bu soruların cevabı kişiden kişiye değişiklik gösterecektir. Ancak bilgisayarlar aracılığıyla çözmek istediğimiz problemlerin çözüm adımlarının, önceden belirgin bir şekilde ortaya konulması gerekir. Bu amaçla hazırlanan yönergelere algoritma denir. Algoritmalar bir problemi çözmek veya belirli bir işi yapmak için adım adım tanımlanmış işlemler kümesidir. Problem çözme sürecini kolaylaştırmak ve hata yapma ihtimalini azaltmak için öncelikle mevcut durumun problem olup olmadığına karar verilmeli ve problem net olarak ortaya konulmalıdır.

[1] Papirüs, eski Mısırlıların bu bitkinin saplarından yaptıkları kâğıt (“https://sozluk.gov.tr”).

Problem net bir şekilde ortaya konulup iyi analiz edildikten sonra çözüm için uygun planlamalar yapılmalıdır. Daha sonra çözüm için farklı yollar düşünülüp bu çözüm yolları arasından uygun olabilecek çözüm yolu seçilmelidir. Problem çözüldükten sonra kontrol edilmelidir.

Şekil 1.1: Problem çözme süreci

Her bir aşamada yapılması gereken işlemler aşağıda açıklanmıştır.

1. Problemi tanımlama: Problemin ne olduğu belirgin bir şekilde ortaya konulmalıdır.
2. Problemi anlama: Problemin kaynağının ne olduğu ve problem çözüldükten sonra beklenen faydalar belirlenmelidir. Bir problem ne kadar iyi anlaşılırsa çözümü o kadar kolay olacaktır.
3. Alternatif çözüm yollarını belirleme: Problemi çözmesi beklenen tüm alternatifler sıralanmalıdır.
4. En uygun çözümü seçme: Bir önceki adımda belirlenen alternatifler arasından en uygun olanının seçilmesi gerekir. Bunun için alternatiflerin artıları ve eksileri yazılabilir. Çoğu zaman çözüme en hızlı ulaştıran alternatif doğru çözüm olarak görünse de bu durum her zaman geçerli değildir. En hızlı çözümün güvenli olmadığı, maliyet açısından kabul edilebilir olmadığı ya da uzun ömürlü olmadığı durumlar ortaya çıkabilir. Böyle durumlarda tüm faktörler göz önünde bulundurularak en uygun çözüm seçilmelidir.
5. Çözümü uygulama: Bir önceki adımda belirlenen çözüm yöntemi kullanılarak problemi çözme işi gerçekleştirilir. Çözüm adımlarının kafa karışıklığına yol açmayacak bir şekilde ortaya konulması gerekir.
6. Çözümü test etme: Uygulanan çözümün beklentileri yerine getirip getirmediği test edilmelidir. Uygulanan çözümün hataları varsa bunları gidermek için önceki işlem basamaklarına dönülmesi gerekebilir.

Örnek 1: Sabahları okula geç kalma problemini, problem çözme adımlarını kullanarak çözmeye çalışalım.

Problemi tanımlama: Okula geç kalma durumunun bir problem olduğunun farkına varılması problemi tanımlama aşamasıdır.

Problemi anlama: Problemin kaynağının ne olduğu tespit edilmelidir. Geç kalmaya birçok şey neden olabilir. Bunlar; uygun ulaşım alternatifini seçmeme, geç uyumaya bağlı geç uyanma, hazırlanma aşamasının uzun sürmesi, okulun çok uzak olması vs. Akşamları bilgisayarda çok fazla oyun oynamaktan kaynaklı geç yatılabildiği, bu nedenle okula geç kalındığı düşünülerek buna yönelik bir çözüm bulmaya çalışılmalıdır.

Alternatif çözüm yolları belirleme: Bu problemi çözmek için aşağıdaki alternatifler kullanılabilir.

Bilgisayarı evden çıkarmak
Ebeveynlerimizden yatma zamanı geldiğinde bizi uyarmasını istemek
Bilgisayardan oyunları kaldırmak
Oyun için ayrılacak günlük süre belirleyip buna uymak

En uygun çözümü seçme: Yukarıda belirlenen her bir alternatifin artıları ve eksileri yazılarak en uygun olanı seçilmeye çalışılmalıdır.

Bilgisayarı evden çıkarma: Hızlı bir çözümdür ancak bazı araştırma ve ödevlerin bilgisayar kullanarak yapıldığını ve evdeki diğer bireylerin de bilgisayar kullandığını düşünürsek uygulanabilir değildir.

Ebeveynlerden yatma zamanı geldiğinde bizi uyarmasını isteme: İşe yarayabilecek bir yöntem olmasına rağmen kişisel sorumluluklarımızı başkalarına yüklemiş olacağımızdan ve iç disiplinimizi sağlamada yetersiz kalacağından mantıklı değildir.

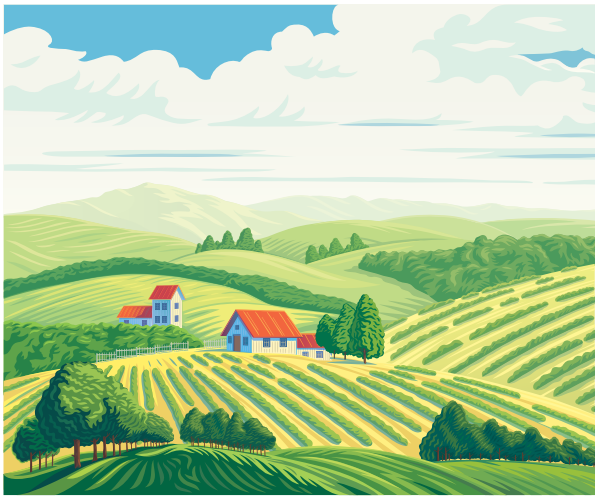
Bilgisayardan oyunları kaldırma: En hızlı alternatiftir ancak bir alışkanlığı bir anda bırakmak kolay olmayacağı için uygulanabilir değildir.

Oyun için ayrılacak günlük süre belirleyip buna uyma: En uygun alternatiftir.

Çözümü uygulama: Kendinize bilgisayarda günlük bir saat oyun oynama limiti koyduğunuzu ve bir süre bunu uyguladığınızı düşününüz.

Çözümü test etme: Okula geç kalma davranışının azalıp azalmadığına bakılarak çözüm test edilebilir.

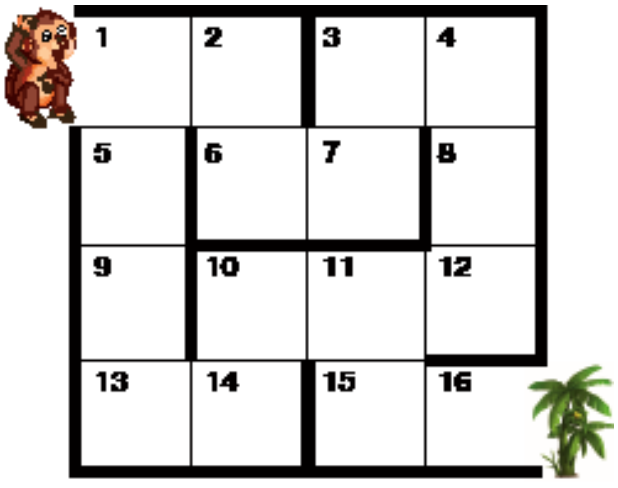
Örnek 2: Bir dağ köyünün tepesinde çiftliği bulunan Hasan Amca'nın Ali ve Ahmet adında iki oğlu varmış. Ali her zaman düzenli ve planlı, Ahmet ise oldukça aceleci birisiymiş. Günlerden bir gün Hasan Amca hem kalıcı hem de insanlara faydalı olsun diye köye bir çeşme yaptırmaya karar vermiş. Çeşme yapımı için gerekli malzemelerin karşıki köyden temin edilmesi gerekiyormuş. Bunun için iki oğlunu yanına çağırıp "Hanginiz malzemeleri alıp işe önce başlarsa çiftlik yönetimini ona devredeceğim." demiş. Hep aceleci oluşuyla tanınan Ahmet, alınacaklar listesini hazırlamadan ve güzergâhı belirlemeden yola çıkmış. Bir de karşıdaki köye erken ulaşabilmek için kısa yoldan gitmeye çalışırken yolunu kaybetmiş. Üstüne üstlük malzeme listesini hazırlamadığı için de eksik malzemelerle ve vakit kaybederek çiftliğe geri dönmüş. Ali ise önce iş planını yapmış. Alınacaklar listesini ve yol güzergâhını önceden belirlediği için erkenden malzemeleri alıp gelerek çeşmeyi yapmaya başlamış ve hızlı bir şekilde bitirmiş. Böylelikle de çiftliğin yönetimi babası tarafından ona hediye edilmiş.



Görsel 1.2: Dağ köyü

Bu örnek; problemi çözmeye çalışırken planlamanın ne kadar önemli olduğunu, plansız hareket edildiğinde boşa emek harcanacağını ve sonuca ulaşmanın çok zor olacağını göstermektedir.

Örnek 3: Aşağıdaki görselde 16 hücreli bir labirent verilmiştir. Labirentte bir başlangıç ve bir bitiş noktası vardır. Amaç; sevimli maymunun duvarlardan atlamadan, herhangi bir hücreye ikinci kez uğramadan ve en kısa yoldan muz ağacına gidebilmesidir.

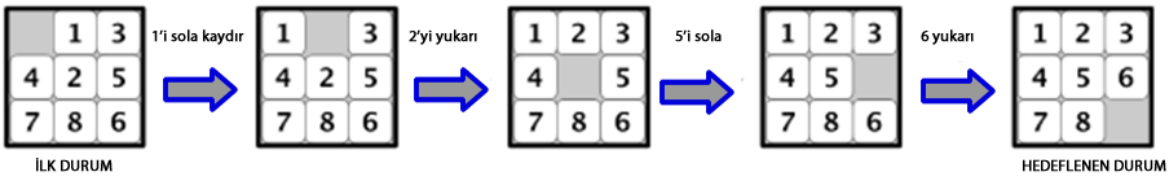


Görsel 1.3: Labirent oyunu örneği

Bulmacalar; yaratıcılığı geliştiren ve analitik düşünmeyi sağlayan, iyi tanımlanmış kurallara ve hedeflere sahip, problem çözme becerilerini geliştiren oyunlardır.

Sevimli maymun, 1-5-9-13-14-10-11-15-16 veya 1-2-6-7-3-4-8-12-11-15-16 adımlarını izleyerek muz ağacına ulaşabilir. Birden fazla çözümü olan örnek problemde en uygun çözüm en kısa çözüm olacaktır. Ancak bir problemi çözerken en kısa çözüm her zaman en doğru çözüm olmayabilir.

Örnek 4: 3*3'lük bir ızgara üzerinde 1'den 8'e kadar numaralandırılmış fayanslar yer almaktadır. Buradaki amaç, mümkün olduğunca az hamle kullanarak fayansları hedefteki görüntüye gelecek şekilde yeniden düzenlemektir.



Görsel 1.4: Problem çözme örneği

İzlenecek adımlarda fayanslar yatay veya dikey olarak boş olan karelere kaydırılır. Yukarıda başlangıç tahtasından hedef tahtasına giderken izlenecek adımlar gösterilmiştir.

1.2. Problem Çözmede Temel İşlemler

Bilgisayarlar, tüm işlemleri matematiksel işlemler yaparak gerçekleştirir. Bu nedenle problem çözmede kullanılabilecek temel matematiksel işlem ve kavramları bilmek gerekir. Bunlar aritmetiksel ve mantıksal operatörler ile karşılaştırma operatörleridir.

Operatörler önceden tanımlanmış birtakım matematiksel ya da mantıksal işlemleri yapmak için kullanılan özel karakter ya da karakterler topluluğudur (Algan, 2008).

1.2.1. Aritmetiksel Operatörler

Toplama, çıkarma, çarpma, bölme, üs alma ve mod alma gibi matematik işlemlerinin yapıldığı operatörlerdir.

Tablo 1.1: Aritmetiksel Operatörler Tablosu

Operatör Adı	Sembolü	Örnek
Toplama	+	3+6
Çıkarma	-	6-3
Çarpma	*	3*6
Bölme	/	6/3
Üs Alma	**	6**3
Mod alma	%	6%3

Mod alma; bir sayının başka bir sayıya bölümünden kalan sayıdır.
Örneğin; 6%3=0 iken 5%3=2 sonucunu verir.

1.2.2. Karşılaştırma Operatörleri

Karşılaştırma işlemi yapılması gereken durumlarda kullanılan operatörlerdir.

Tablo 1.2: Karşılaştırma Operatörleri Tablosu

Operatör Adı	Sembolü	Örnek
Eşittir	==	ad=='özge'
Eşit Değildir	!=	ad!='özge'
Büyüktür	>	a>45
Küçüktür	<	a<45
Büyük Eşittir	>=	5>=a
Küçük Eşittir	<=	a<=5

Karşılaştırma operatörleri, karşılaştırma sonunda true (doğru) veya false (yanlış) değeri döndürür.

Örnek 1: 2>1 ifadesi "true" değerini döndürürken, 5<2 ifadesi "false" değerini döndürür.

Örnek 2: i=50 olsun. i<30 ifadesi "false" değerini döndürecektir.

Sıra Sizde: 4==4 ifadesi hangi değeri döndürür?

1.2.3. Mantıksal Operatörler

"ve", "veya", "değil" gibi mantıksal işlemleri yapan operatörlerdir.

"Ve" operatörü, iki veya daha fazla koşulun tümünün doğru olduğu durumlarda "doğru" sonucunu veren operatördür. Günlük hayattaki kullanımıyla aynıdır.

Örneğin makarna yapmak için su, tuz ve makarna gerekir. Bunlardan herhangi biri olmadan makarna yapılamaz.

"Veya" operatöründe, iki veya daha fazla koşuldan en az birinin doğru olması durumunda sonuç "doğru" olur. Bu da günlük hayatta kullandığımız gibidir.

Örneğin bir havuzu boşaltmak için kırmızı, yeşil ve mavi renkte üç farklı musluğumuz olsun. Havuzun boşalması için kırmızı musluğu veya yeşil musluğu veya mavi musluğu açmamız yeterli olacaktır. Dilersek iki ya da üç musluğu aynı anda açarak da havuzu boşaltabiliriz.

"Değil" operatörü ise mantıksal bir durumu tersine çevirir. Sonucu "doğru" olan bir mantıksal sınamayı "yanlış"a, sonucu "yanlış" olan bir mantıksal sınamayı ise "doğru"ya çevirir. Günlük hayattaki olumsuzluk ifadelerine karşılık gelir.

Örneğin annelerimiz meyve alırken pazarcıya "Sağlamlarından ver." diyebilir. "Çürüklerinden verme." deseler de pazarcı yine aynı şeyi anlayacaktır. Çünkü bir meyve ya çürük ya da sağlam olur. İkinci ifadede kullanılan olumsuzluk ekinin koşulu tersine çevirdiğine dikkat ediniz.

Tablo 1.3: Mantıksal Operatörler Tablosu

Operatör Adı	Sembolü	Örnek
Ve (and)	and	a<4 and a>8
veya (or)	or	a<4 or a<3
değil (not)	not	not(a==b)

Mantıksal operatörlerin doğruluk tablosunda gösterilmiş hâli aşağıdaki gibidir.

Tablo 1.4: Mantıksal Operatörlerin Doğruluk Tablosunda Gösterilişi

ve			veya			değil	
a	b	a and b	a	b	a or b	a	a'
1	1	1	1	1	1	1	0
0	1	0	0	1	1	0	1
1	0	0	1	0	1		
0	0	0	0	0	0		

1.2.4. İşlem Önceliği

Aritmetik işlemler yapılırken kullanılan operatörlerde öncelik sıralaması vardır.

Bu sıralama;

Parantez ( )
Üs alma **
Çarpma - Bölme *, /
Toplama - Çıkarma +,- şeklindedir.

Örnek 1: 10+4*3/(8+4)=? işlemini yapınız.

Çözüm: =10+4*3/12
=10+12/12
=10+1
=11 olacaktır.

Örnek 2: $(2^2+4)-8$=? işlemini yapınız.

Çözüm: =(4+4)-8
=8-8
=0 olacaktır.

Sıra Sizde: 20+16/4-10*1+5=? işleminin sonucu nedir?

1.3. Algoritmalar

Algoritma kelimesi bir İslam Bilgini olan El-Harezmi'nin (780-850) isminin Latince karşılığından gelmektedir. El-Harezmi matematik, gök bilim ve coğrafya alanlarında çalışmış, cebirin temelini oluşturmuş, bugünkü bilgisayar bilimi ve elektroniğin temeli olan 2'lik (binary) sayı sistemini ve 0'ı (sıfır) bulmuş önemli bir bilim insanıdır.

Programlamanın öğrenilebilmesi için öncelikle algoritmanın ne olduğuna ve nasıl geliştirilmesi gerektiğine cevap bulunmalıdır.

Problem çözme yöntemlerinden biri olan algoritma geliştirmek kodlamaya atılan ilk adımdır. Algoritma mantığı iyice kavrandıktan sonra bu mantık ile birlikte bir programlama dili kullanılarak yazılım geliştirme süreci başlar.

Algoritma, belirli bir mantığı olan, farklı düşünebilmeyi ve problem çözmeyi öğretmek için tasarlanan bir yoldur. Başka bir ifadeyle bir problemi çözmeye giden yolun basit, net ve belirli bir sıraya göre tasarlanmış hâlidir.

Algoritmalar;
Açık ve net olmalıdır.
Kullanılacak olan girdiler iyi tanımlanmış olmalıdır.
Çıktılar açık ve anlaşılır olmalıdır.
Algoritmalar hızlı olmalıdır.
Sonlu ve uygulanabilir olmalıdır.

Görsel 1.5: Algoritma

Örnek 1: : Evimize gelen arkadaşımıza Türk kahvesi ikram edelim mi?

2 kişilik sade Türk kahvesi hazırlama algoritması aşağıdaki gibi olacaktır:

Adım 1- Başla

Adım 2- Cezveye iki fincanlık su koy

Adım 3- Cezveye 2 tatlı kaşığı kahve koy

Adım 4- Ocağı yak

Adım 5- Cezveyi ocağa koy

Adım 6- Kahveyi karıştır

Adım 7- Bir süre kahvenin olmasını bekle

Adım 8- Kahve köpürmeye başladı mı? Hayır ise 6. Adıma git

Adım 9- Kahveyi fincanlara doldur

Adım 10- Bitir

Örnekte görüldüğü gibi her algoritma bir başlama komutu ile başlar ve bitirme komutu ile sona erer. Problemi çözmek için problem küçük parçalara bölünür ve hedeflenen sonuca ulaşmak için atılan adımlar net ve uygulanabilir niteliktedir.

Örnek 2: Öğretmeninizin sizi soru çözmek için tahtaya kaldırmış olduğunu düşününüz. Algoritmayı çözmek için işlem basamakları aşağıdaki gibi olacaktır:

Adım 1- Başla

Adım 2- Ayağa kalk

Adım 3- Tahtanın önüne git

Adım 4- Tahta kalemini al

Adım 5- Soruyu çöz

Adım 6- Soruyu doğru çözdün mü? Evet ise 8. Adıma git

Adım 7- Tahtayı sil ve 5. Adıma git

Adım 8- Yerine otur

Adım 9- Bitir

Sıra Sizde: Sizler de çay demleme algoritmasını metinsel olarak hazırlayınız.

1.3.1. Sözde Kod (Pseudo-code)

Konuşma dili ile programlama dili arasında, algoritma geliştirmek için kullanılan yapay kodlara sözde kod denir. Sözde kodlar, günlük dille ifade edilmiş (metinsel olarak tanımlanmış) problemlerin programlamaya yaklaştırılmış hâlidir.

Değişken: Her seferinde farklı değerler alabilen ifadelerdir.

Sabit: Değeri değişmeyen ifadelerdir.

Örnek 1: Girilen sayının karesini bulan algoritmayı metinsel ve sözde kod kullanarak yazınız.

Metinsel algoritma	Sözde kod
Adım 1- Başla	Adım 1- Başla
Adım 2- Sayıyı oku	Adım 2- Oku a
Adım 3- Sayının karesini hesapla	Adım 3- kare=a*a
Adım 4- Sonucu ekrana yaz	Adım 4- Yaz kare
Adım 5- Bitir	Adım 5- Bitir

Bu örnekte "a" ile "kare" birer değişkendir.

Örnek 2: İki sayıyı çarpıp sonucu ekrana yazdıran algoritmayı metinsel ve sözde kod kullanarak yazınız.

Metinsel algoritma	Sözde kod
Adım 1- Başla	Adım 1- Başla
Adım 2- Birinci sayıyı oku	Adım 2- sayi1'i oku
Adım 3- İkinci sayıyı oku	Adım 3- sayi2'yi oku
Adım 4- İki sayıyı çarp	Adım 4- carpim=sayi1*sayi2
Adım 5- Çarpımı ekrana yaz	Adım 5- Yaz carpim
Adım 6- Bitir	Adım 6- Bitir

Bu örnekte sayi1, sayi2 ve carpım birer değişkendir.

Örnek 3: Bir öğrencinin matematik dersinden aldığı iki notun ortalamasını hesaplayan algoritmayı metinsel ve sözde kod kullanarak yazınız.

Metinsel algoritma	Sözde kod
Adım 1- Başla	Adım 1- Başla
Adım 2- Birinci notu oku	Adım 2- Oku not1
Adım 3- İkinci notu oku	Adım 3- Oku not2
Adım 4- Ortalamayı hesapla	Adım 4- ortalama=(not1+not2)/2
Adım 5- Ortalamayı ekrana yaz	Adım 5- Yaz ortalama
Adım 6- Bitir	Adım 6- Bitir

Sıra Sizde: Dik üçgenin alanını hesaplayan metinsel algoritmayı ve sözde kodu yazınız.

Örnek 4: Klavyeden girilen iki sayının büyük olanından küçük olanını çıkaran algoritmayı yazınız.

Metinsel algoritma
Adım 1- Başla
Adım 2- Oku sayi1
Adım 3- Oku sayi2
Adım 4- Eğer sayi1>sayi2 ise sonuc=sayi1-sayi2
Adım 5- Değilse sonuc=sayi2-sayi1
Adım 6- Yaz sonuc
Adım 7- Bitir

Mantıksal algoritmalar: Algoritma içinde karşılaştırma yapma veya karar vermeyi gerektiren durumlar için kullanılır.

Örnek 5: Suyun sıcaklığına göre maddenin katı, sıvı veya gaz olma durumunu gösteren algoritmayı hazırlayınız.

Adım 1- Başla

Adım 2- Oku sicaklik

Adım 3- Eğer sicaklik<=0 ise "katı" yaz

Adım 4- Eğer sicaklik>0 ve sicaklik<100 ise "sıvı" yaz

Adım 5- Eğer sicaklik >=100 ise "gaz" yaz

Adım 6- Bitir

Sıra Sizde: Klavyeden girilen yaş değeri 18'den büyük ve eşitse "Reşittir", aksi hâlde "Reşit değildir" yazan algoritmayı hazırlayınız.

Örnek 6: Ekrana 5 defa "merhaba" yazdıran algoritmayı yazınız.

Adım 1- Başla

Adım 2- sayac=0

Adım 3- yaz "merhaba"

Adım 4- sayac=sayac+1

Adım 5- Eğer sayac<5 ise git Adım 3

Adım 6- Bitir

Döngüsel algoritmalar: Algoritma içinde tekrar eden işlemler(döngü) için kullanılır.

Örnek 7: 0'dan 100'e kadar olan çift sayıları ekrana yazdıran algoritmayı hazırlayınız.

Adım 1- Başla

Adım 2- sayac=0

Adım 3- Yaz sayac

Adım 4- sayac=sayac+2

Adım 5- Eğer sayac<=100 ise git Adım 3

Adım 6- Bitir

Sayaç: Bir işlemin belli bir sayıda artması veya azalması şeklindeki sayma işlemlerinde kullanılan değişken.

Sıra Sizde: Klavyeden girilen 5 adet sayıdan 20'den küçük olanların sayısını gösteren algoritmayı hazırlayınız.

1.3.2. Problem Çözmede Algoritma Hataları

Bir problemin çözümüyle ilgili algoritma hazırlayıp işleme aldığımızda hatalı sonuçlar meydana gelebilir. Bu gibi hatalara algoritma hataları denir.

Örnek 1: Evimizin bahçesinde kedimiz ve köpeğimiz olduğunu, bunlara ayrı ayrı mama kapları kullandığımızı düşünelim (Kedinin kabı mavi renkli, köpeğin kabı beyaz renkli olsun.).

Kediyi besleme problemini algoritma hâlinde aşağıdaki şekilde yazalım.

Adım 1- Başla

Adım 2- Kedi mamasını al

Adım 3- Yemek kabına koy

Adım 4- Bitir

Bu algoritmayı adım adım inceleyecek olursak;

Adım 1- Başla (Başla komutu ile işlemleri başlattık.).

Adım 2- Kedi mamasını al (Kedi mamasını aldık.).

Adım 3- Yemek kabına koy (Burada kedi mamasını kaba koyacağız ancak kabın rengi belirtilmediği için hangi kaba mama koyacağımızı bilmiyoruz.).

Karşılaşılan bu gibi hatalara algoritma hataları denilir ve bu hatalar, yazılmış olan kodlar adım adım incelenerek çözülür.

Örnek 2: Ellerimizi yıkama işleminin algoritmasını aşağıdaki şekilde yazalım.

Adım 1- Başla

Adım 2- Musluğu aç

Adım 3- Ellerini yıka

Adım 4- Bitir

Bu algoritmayı adım adım inceleyecek olursak;

Adım 1- Başla (Başla komutu ile işlemleri başlattık.).

Adım 2- Musluğu aç (Musluk açıldı.).

Adım 3- Ellerini yıka (Burada eller yıkanıyor ancak önemli bir mantık hatası var. Çünkü ellerimizi sabunla yıkamamız gerekir. Ayrıca işlemi bitirmeden önce musluğu kapatmamız gerekir.).

Algoritmayı düzenleyerek yeniden yazacak olursak;

Adım 1- Başla

Adım 2- Musluğu aç

Adım 3- Ellerine sabun al

Adım 4- Ellerini yıka

Adım 5- Musluğu kapat

Adım 6- Bitir

şeklinde olmalıdır.

Sıra Sizde: Sınıfınızda üçer kişilik gruplar oluşturunuz. Herkes ayrı ayrı ATM'den para çekme algoritmasını yazdıktan sonra her grupta algoritmaları birlikte inceleyerek hatalarını gidermeye çalışınız.

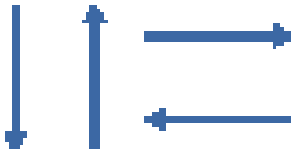
Görsel 1.6: Akış diyagramı

1.4. Akış Diyagramları

Algoritma ile adım adım yapılan işlemlerin, özel semboller (geometrik şekiller) kullanılarak gösterilmesine akış diyagramı denir.

Akış diyagramı hazırlanırken kullanılan şekiller ve anlamları Tablo 1.5'te gösterilmiştir.

Tablo 1.5: Akış Diyagramı Sembol ve Görevleri

ŞEKİLLER	KULLANIM YERİ
	Başla-Bitir Algoritmanın başladığını ve bittiğini gösteren semboldür.
	Aritmetik işlemlerin ve değişkene değer atama işlemlerinin yapıldığı semboldür.
	Veri giriş ve değişken tanımlamalarının yapıldığı sembol-dür.
	Önceden tanımlı işlem / fonksiyonları çalıştırmak için kulla-nılan semboldür.
	Çıktı almak ve ekran görüntüsü oluşturmak için kullanılan semboldür.
	Döngü işlemleri için kullanılan semboldür.
	Karar verme / karşılaştırma işlemleri için kullanılan sembol-dür.
	Akış noktalarını bağlamak için kullanılan semboldür.
	Akış yönünü gösterir. Akış, okun yönüne göre ilerlemekte-dir.

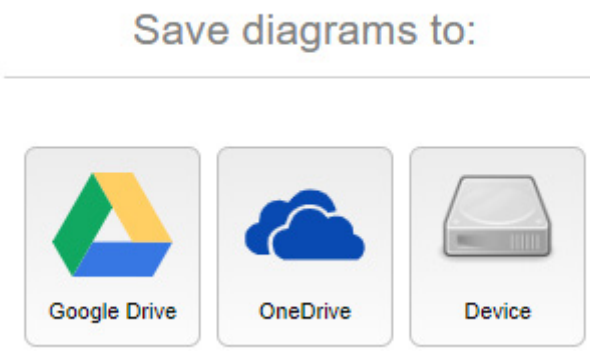
1.4.1. Flowchart (Akış Diyagramı) Hazırlama Programının Kurulumu

Akış diyagramı hazırlama programı kullanılarak hazırlanmak istenilen akış şemaları bilgisayar ortamında kolaylıkla çizilebilir.

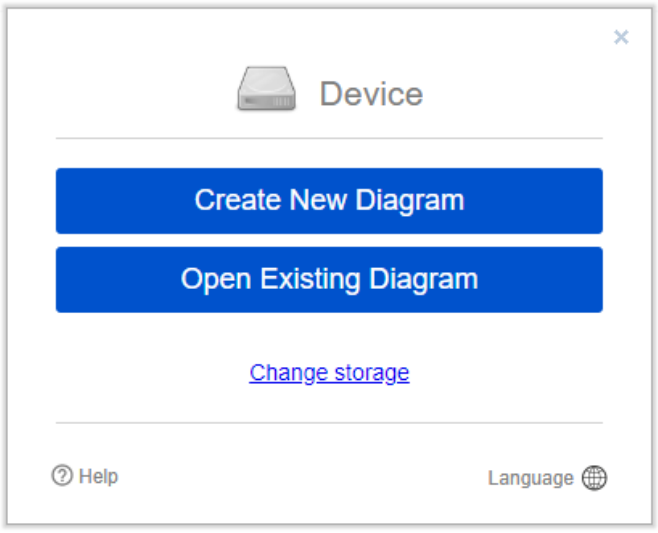
Akış diyagramı hazırlama programı bilgisayara kurulum gerektirmeden çevrimiçi kullanılabileceği gibi bilgisayara kurularak internet bağlantısı olmadan da kullanılabilir.

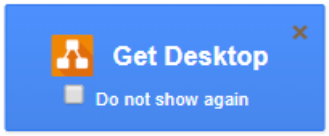
Akış diyagramı hazırlama programına draw.io internet adresinden erişilebilir.

Akış diyagramı hazırlama programı tarayıcıda çalıştırıldığı zaman aşağıdaki görüntü ile karşılaşılır:



Akış diyagramı programı ile çalışılacak ortam buradan seçilir.

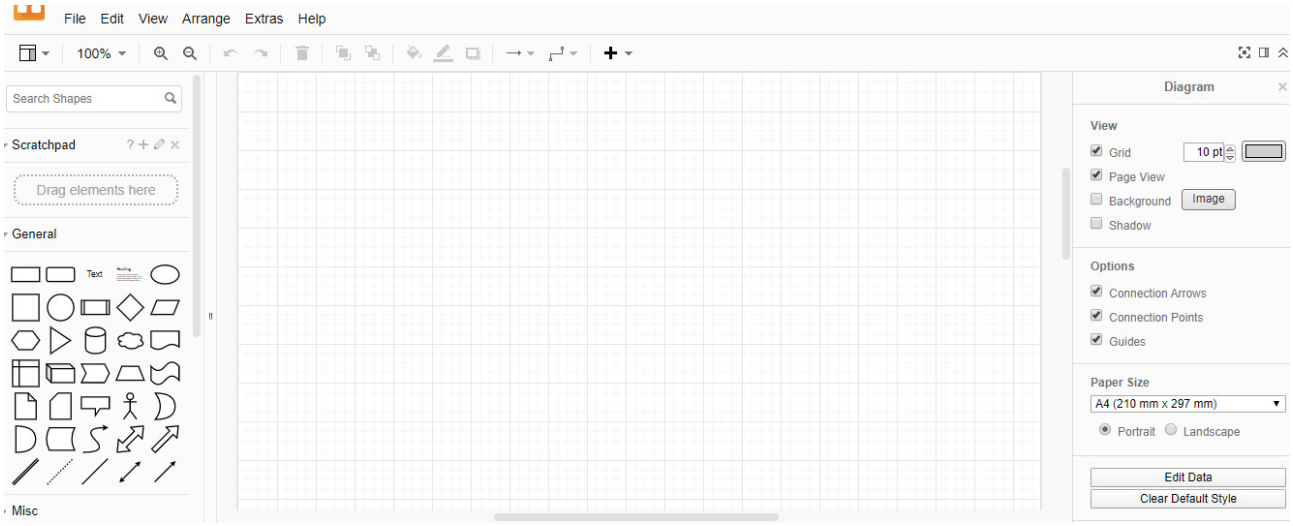




Görsel 1.7: Flowchart programı kurulum

"Get desktop" seçeneği programın .exe dosyasını bilgisayara indirerek internet bağlantısı olmadan da programın kullanılmasını sağlar.

Sırası ile Create New Diagram ardından da Blank Diagram seçilip Create düğmesine tıklandığı zaman akış şemalarının hazırlanacağı arayüz ekranı açılır (Görsel 1.8).



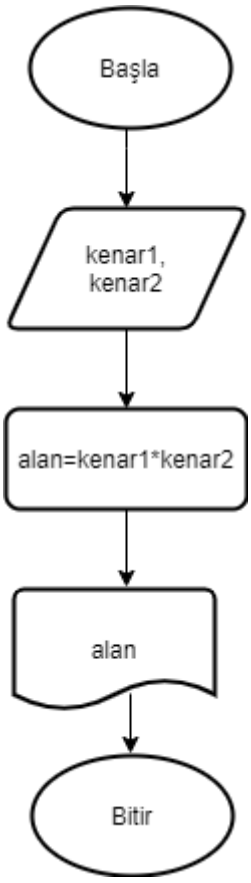
Görsel 1.8: Akış şeması arayüzü

Kullanılacak olan nesneler sürükle bırak yöntemi ile tasarım alanına aktarılarak akış şemaları basit bir şekilde oluşturulur.

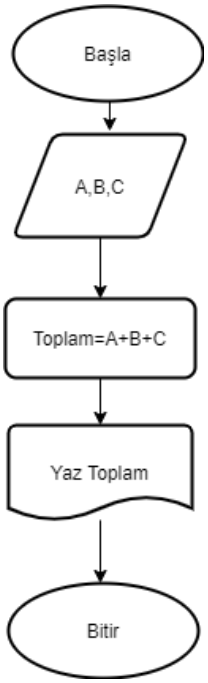
1.4.2. Doğrusal Akış Şeması Örnekleri

http://kitap.eba.gov.tr/KodSor.php?KOD=22313

Örnek 1: Dikdörtgenin alanını hesaplayan algoritmayı ve akış şemasını hazırlayınız.

ALGORİTMA	AKIŞ ŞEMASI
A1-Başla A2-Oku kenar1 A3-Oku kenar2 A4-alan=kenar1*kenar2 A5-Yaz alan A6-Bitir	

Örnek 2: Klavyeden girilecek 3 sayıyı toplayıp sonucu ekrana yazdıran programın algoritmasını ve akış şemasını hazırlayınız.

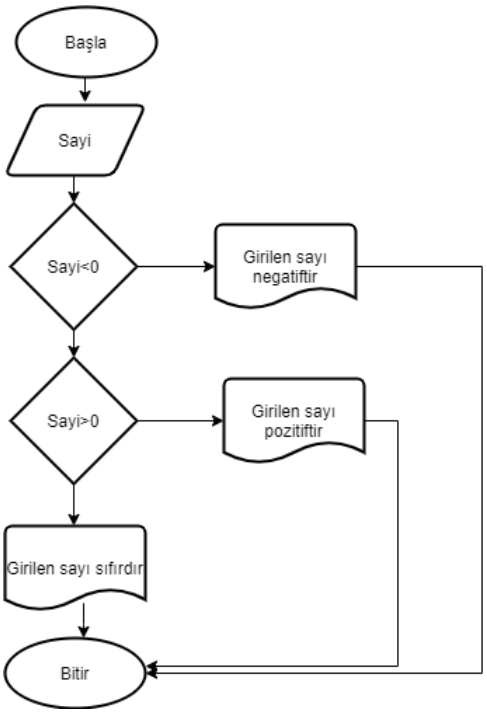
ALGORİTMA	AKIŞ ŞEMASI
A1-Başla A2-Sayıları giriniz; A,B,C A3-Toplam=A+B+C A4-Yaz Toplam A5-Bitir	

Sıra Sizde: Dikdörtgenin çevresini hesaplayan algoritmayı ve akış şemasını hazırlayınız.

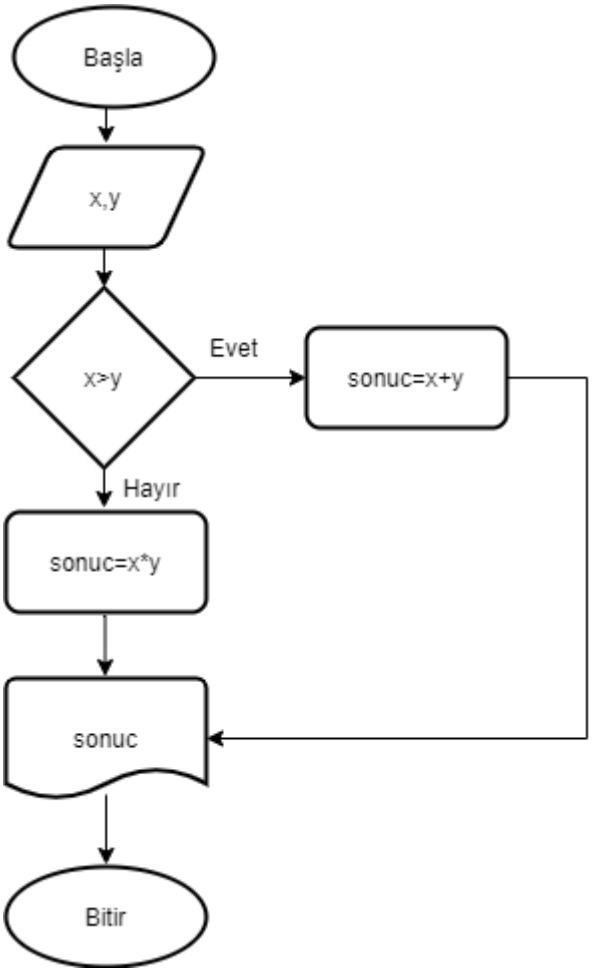
http://kitap.eba.gov.tr/KodSor.php?KOD=22319

1.4.3. Karar İfadeleri Kullanılarak Hazırlanan Akış Şeması Örnekleri

Örnek 1: Klavyeden girilen bir sayının negatif mi, pozitif mi yoksa sıfır mı olduğunu yazdıran programın algoritmasını ve akış şemasını hazırlayınız.

ALGORİTMA	AKIŞ ŞEMASI
A1- Başla A2- Sayıyı gir;Sayi A3- Eğer Sayi<0 ise yaz "girilen sayı negatiftir" ve A6 ya git A4- Eğer Sayi>0 ise yaz "girilen sayı pozitiftir" ve A6 ya git A5- Girilen sayı sıfırdır A6- Bitir	

Örnek 2: Klavyeden girilen iki sayıdan birincisi büyük ise toplama, ikincisi büyük ise çarpma işlemi yapan algoritmayı ve akış şemasını hazırlayınız.

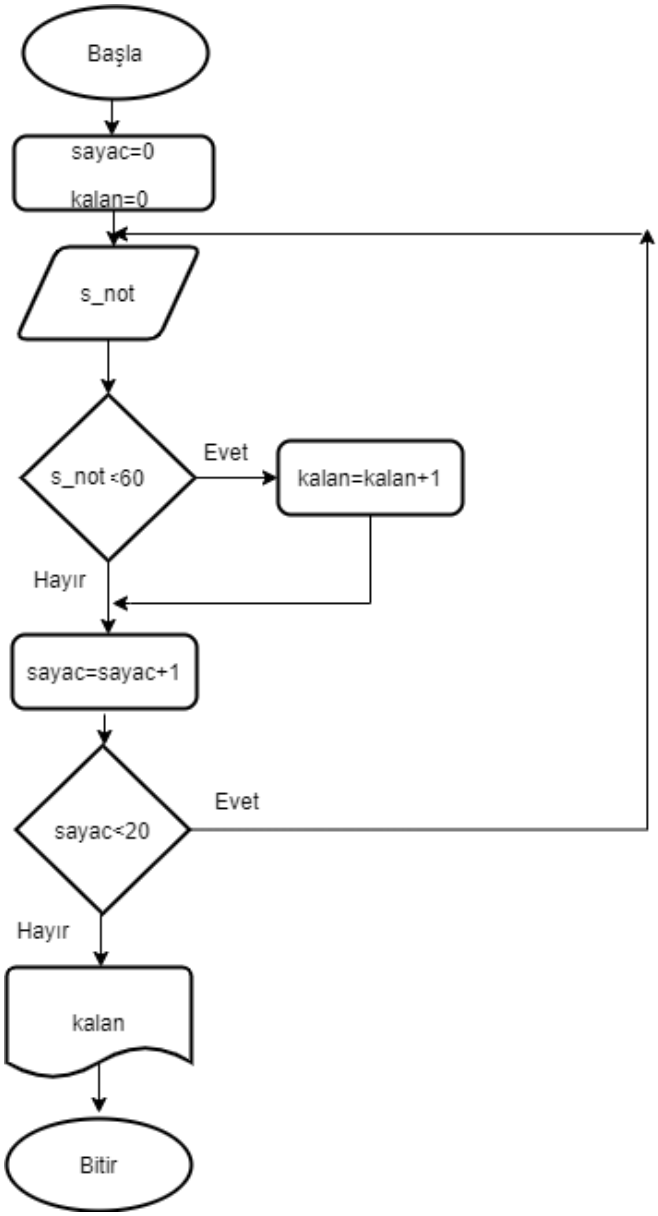
ALGORİTMA	AKIŞ ŞEMASI
A1-Başla A2-Sayıları gir; x,y A3-Eğer x>y ise sonuc=x+y ve git A5 A4-Değilse sonuc=x*y A5-Yaz sonuc A6-Bitir	

Sıra Sizde: Klavyeden girilen iki sayıdan büyük olanı ekrana yazdıran algoritmayı ve akış şemasını çiziniz.

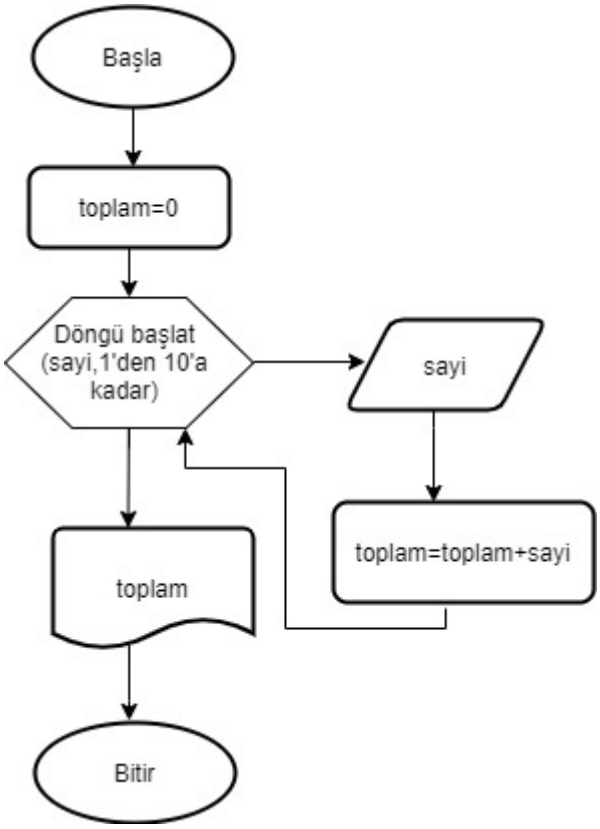
1.4.4. Döngüler Kullanılarak Hazırlanan Akış Şeması Örnekleri

http://kitap.eba.gov.tr/KodSor.php?KOD=22320

Örnek 1: 20 Öğrencinin Programlama Temelleri dersi birinci sınav notları giriliyor. Geçme notu 60 olan sistemde, kalan öğrenci sayısını bulan algoritmayı ve akış şemasını hazırlayınız.

ALGORİTMA	AKIŞ ŞEMASI
A1- Başla A2- sayac=0, kalan=0 A3- Sınav Notunu Gir, s_not A4- Eğer s_not<60 ise kalan = kalan + 1 A5- sayac = sayac + 1 A6- Eğer sayac <= 20 ise, Git A3 A7- Yaz kalan A8- Bitir	

Örnek 2: Klavyeden girilen 10 sayının toplamını hesaplayan algoritmayı ve akış diyagramını hazırlayınız.

ALGORİTMA	AKIŞ ŞEMASI
A1- Başla A2- toplam=0, A3- Döngü başlat (sayi,1'den 10'a kadar) A4- toplam=toplam+sayi A5- Döngüyü bitir A6- Yaz toplam A7- Bitir	

Sıra Sizde: Sıfır girilinceye kadar girilmiş olan sayıların karesini hesaplayan algoritmayı ve akış şemasını hazırlayınız.

ÖLÇME VE DEĞERLENDİRME 1

1. Aşağıdakilerden hangisi bir problemin çözümü için doğru değildir?

   A) Bir problemi çözebilmek için problemin net bir şekilde ortaya konması gerekir.

   B) Problem ne kadar iyi anlaşılırsa çözüm de aynı ölçüde kolay olacaktır.

   C) Problemi çözmek için farklı yol ve yöntemler denenmelidir.

   D) Çözüm adımları kafa karıştırmayacak şekilde olmalıdır.

   E) Çözüme ulaşabilmek için her zaman tek bir yol yeterlidir.

2. Sürekli kilo alma probleminin sebebinin ne olduğunu tespit etme problem çözme basamaklarının hangisine aittir?

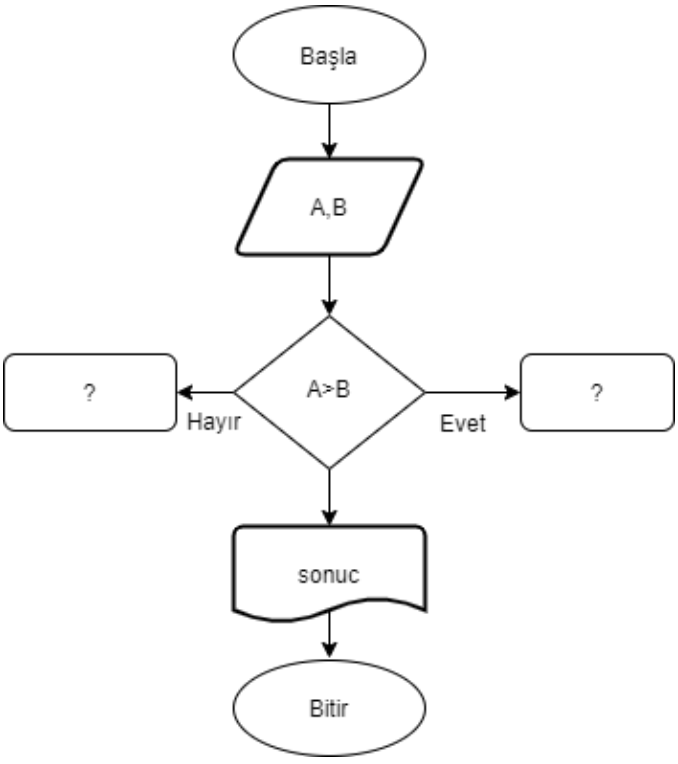
   A) Problemi tanımlama

   B) Problemi anlama

   C) Alternatif çözüm yolu belirleme

   D) Çözümü uygulama

   E) Çözümü test etme



3. Yukarıdaki tabloda klavyeden girilen iki sayının büyük olanından küçük olanını çıkarıp sonucu ekrana yazdıran programın akış şeması verilmiştir. Programın doğru sonuç vermesi için soru işareti olan yerlere aşağıdakilerden hangisi getirilmelidir?

   A) Evet: sonuc=A+1, Hayır: sonuc=A-B

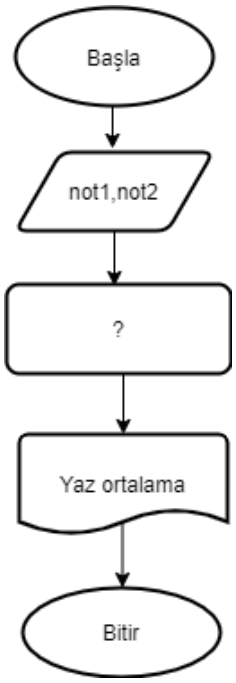
   B) Evet: sonuc=A-B, Hayır: sonuc=B-A

   C) Evet: sonuc=B-A, Hayır: sonuc=A-B

   D) Evet: sonuc=sonuc-A, Hayır: sonuc=sonuc-B

   E) Hiçbiri

4. Klavyeden girilen iki notun ortalamasının bulunması istenmektedir. Algoritma aşağıdaki gibidir.



Buna göre soru işareti olan yere aşağıdakilerden hangisi gelmelidir?

A) ortalama=not1+not2

B) ortalama=(not1)+(not2)

C) ortalama=(not1+not2/2)

D) ortalama=(not1+not2)/2

E) ortalama=toplam-(not1+not2)

5. Aşağıda karışık hâlde verilmiş adımları doğru şekilde düzenleyerek "makarna pişirme" algoritmasını hazırlayınız.

Bitir, suyu kaynat, tencereye su koy, başla, makarnayı ekle, tuz ekle, makarnayı pişir, makarnanın suyunu süz.

Adım 1:……

Adım 2:……

Adım 3:……

Adım 4:……

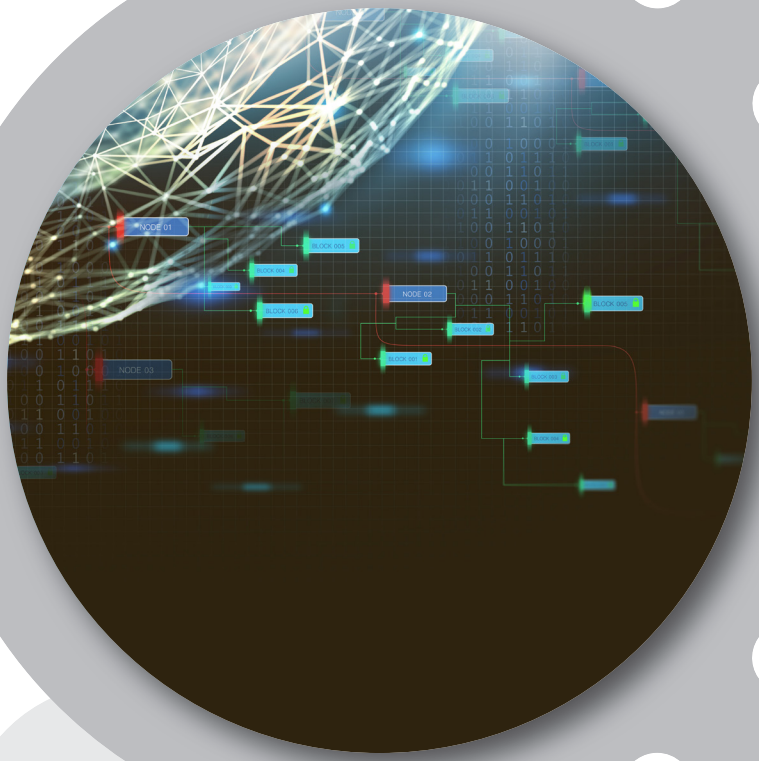
Adım 5:……

Adım 6:……

Adım 7:……

Adım 8:……

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.



ÖĞRENME BİRİMİ 2

BLOK TABANLI PROGRAMLAMA

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Blok tabanlı programlama kavramını tanımlayabilecek,
- Blok tabanlı programın kurulumunu yapabilecek,
- Blok tabanlı programa çevrimiçi kayıt yapabilecek,
- Blok tabanlı programı kullanarak kendi karakterlerinizi oluşturabilecek,
- Blok tabanlı programda yaptığınız çalışmalara sesler ekleyebilecek,
- Blok tabanlı programda koordinat düzlemi üzerinde şekiller çizebilecek,
- Blok tabanlı programı kullanarak matematiksel işlemler yapabilecek,
- Blok tabanlı programı kullanarak animasyonlar tasarlayabilecek,
- Blok tabanlı programı kullanarak bilgisayar oyunları hazırlayabileceksiniz.

Anahtar Kelimeler:

Kodlama, blok programlar, oyun, dekor, kukla, kostüm, animasyon.

Hazırlık Çalışmaları

1. Blok tabanlı programların neler olduğunu araştırınız.
2. Blok tabanlı programlarla neler yapılabileceğini araştırınız.
3. Blok tabanlı programların masaüstü ve internet ortamında kullanımının avantajlarının neler olduğunu araştırınız.

2. BLOK TABANLI PROGRAMLAMA

2.1. Blok Tabanlı Programlama Ortamı

Blok tabanlı kodlama programları; programın sunduğu hazır kod bloklarını kullanarak oyunlar, etkileşimli hikâyeler, projeler ve animasyonlar hazırlayıp paylaşmak için kullanılan görsel kodlama ortamlarıdır.

Blok tabanlı programlar yaratıcı düşünme ve problem çözme becerilerinin gelişmesine imkân sağlar. Kod bloklarının sürükle bırak yöntemi kullanılarak art arda sıralanmasıyla verilen problemin çözümü gerçekleştirilir.

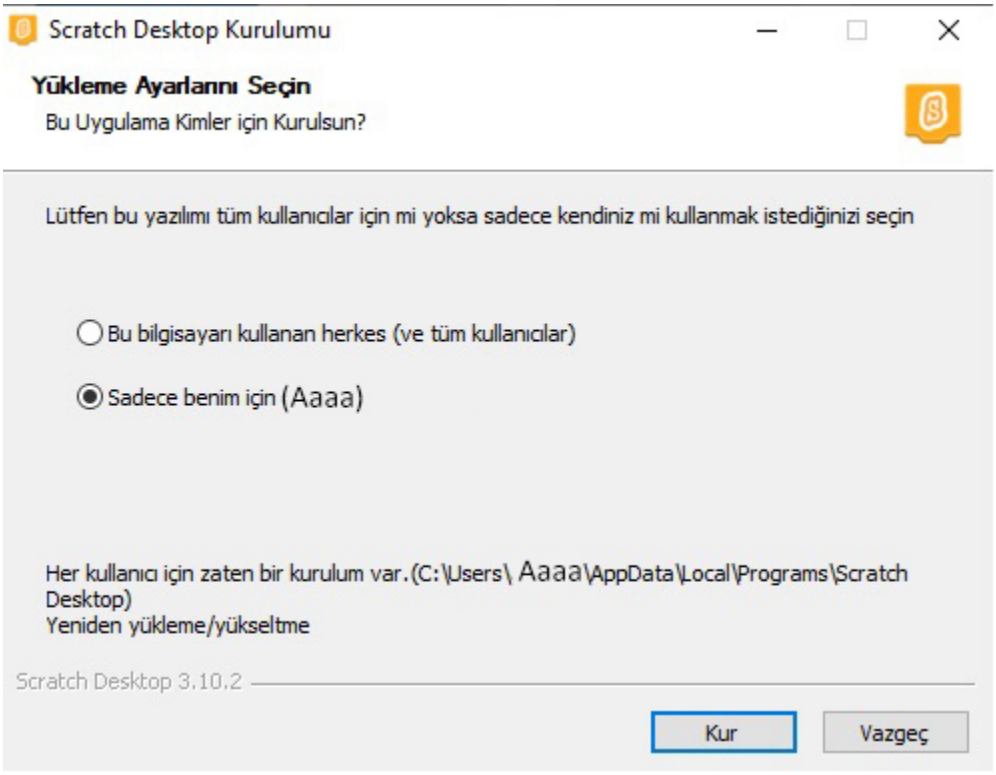
Blok tabanlı programlar çevrimiçi (on-line) paylaşım özelliği sayesinde, yapılmış olan çalışmaları dünya üzerindeki programı kullanan herkesle paylaşarak sosyalleşmeye de imkân sağlar.

Blok tabanlı kodlama programı olan Scratch, MIT Medya Lab'ında yer alan Lifelong Kindergarten grubu tarafından geliştirilmiş bir projedir ve ücretsizdir.

Blok tabanlı kodlama programı, internet üzerinden çevrimiçi olarak kullanılabileceği gibi bilgisayara indirip kurulum yapılarak çevrimdışı (off-line) olarak da kullanılabilir.

2.1.1. Kurulum

Scratch programını çevrimdışı kullanabilmek için https://scratch.mit.edu/download adresini ziyaret ettikten sonra bilgisayarınızda kurulu olan işletim sistemini seçip programı bilgisayara indirip ".exe" uzantılı dosyayı çalıştırarak yönergeler doğrultusunda programı bilgisayarınıza kurabilirsiniz.



Görsel 2.1: Scratch desktop (masaüstü) kurulumu

2.1.2. Hesap Oluşturma

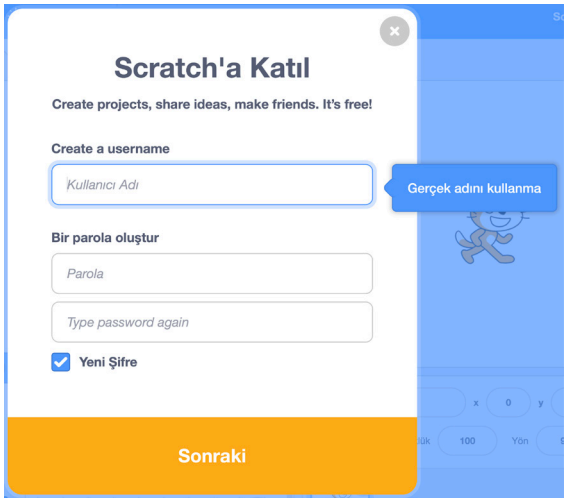
Programı internet üzerinden çevrimiçi kullanmak için https://scratch.mit.edu/ adresini ziyaret edip yönergeleri takip ederek çevrimiçi kayıt yapılabilir.

Uygulama: Hesap oluşturma

Scratch programında hesap oluşturmak için aşağıdaki yönergeleri uygulayınız.

Scratch programı web sitesine giriniz.

Scratch'a Katıl düğmesine tıklayarak çevrimiçi kayıt yapınız.



Görsel 2.2: Scratch hesap oluşturma

Kullanıcı adı belirleyiniz (Gerçek adınızın olmaması tavsiye edilir.).

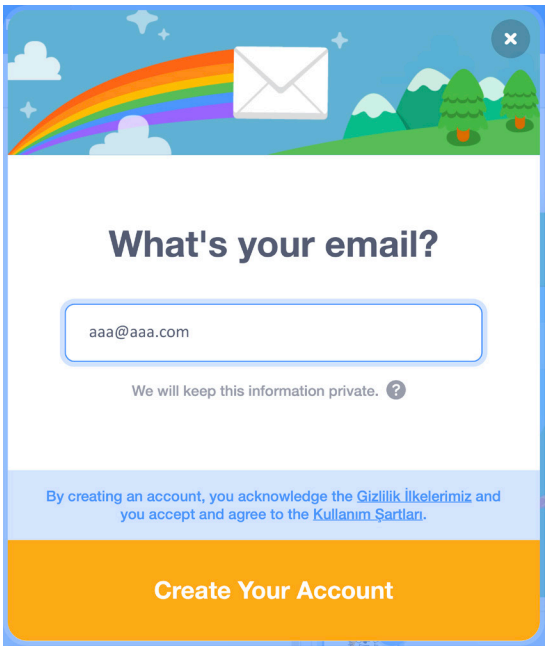
Bir parola belirleyiniz.

Yaşadığınız ülkeyi seçiniz.

Doğum tarihinizi seçiniz (Ay-yıl olarak).

Cinsiyetinizi seçiniz.

E-mail adresinizi yazınız ve sözleşmeyi kabul edip hesabı oluşturunuz.

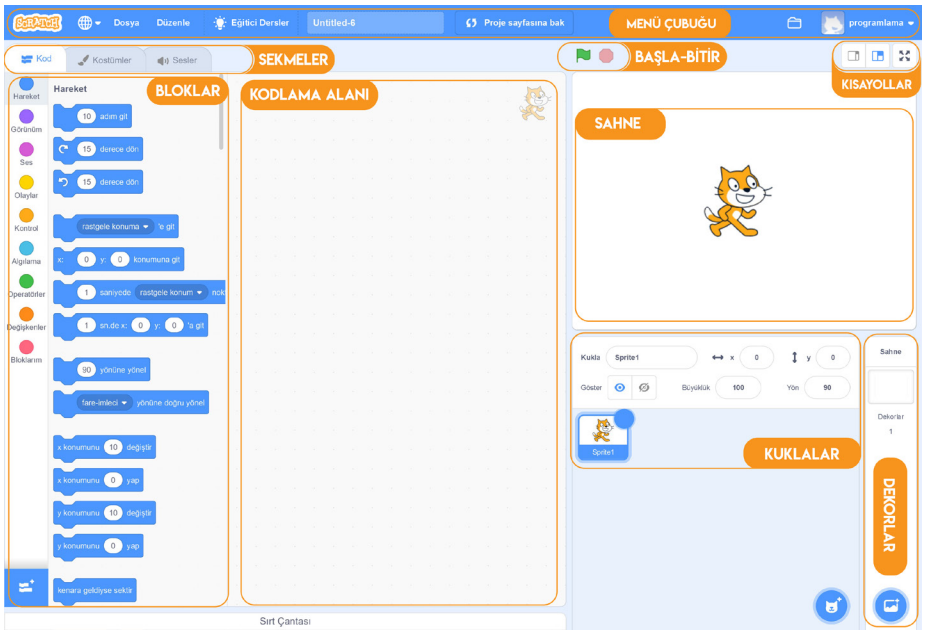


Görsel 2.3: Kişisel hesap oluşturma

DİKKAT: Hazırlanan projelerin paylaşılabilmesi için posta adresinin doğrulanması gerektiğinden geçerli bir e-posta adresi girilmelidir.

2.2. Blok Tabanlı Programın Arayüzü

Scratch programı, kullanımı kolay ve anlaşılabilir bir arayüze sahiptir. Programın web sitesine üyelik bilgileriyle giriş yapılıp Oluştur butonuna (düğmesine) tıklandığında Görsel 2.4'teki arayüz ekranı gelecektir.



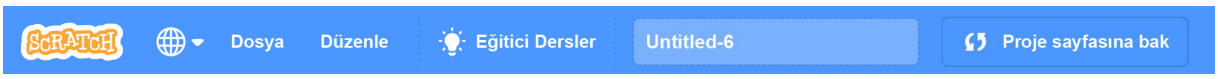
Görsel 2.4: Scratch programı çevrimiçi arayüz ekranı

Bloklar: Scratch programında kullanılan blokların yer aldığı bölümdür.

Kodlama alanı: Blokların sürüklenmesiz ve alt alta yerleştirilip birleştirilmesi ile kodlamanın yapıldığı bölümdür.

Sahne: Projenin ön izlemesinin yapıldığı bölümdür. Scratch programında hazırlanan her şey bu bölümde hayat bulur.

Kuklalar: Kuklalarla ilgili işlemlerin (isim, yön, boyut, görünürlük, konum) yapıldığı bölümdür.



Görsel 2.5: Menü çubuğu

Dünya sembolü, komutların istenilen dilde kullanılmasını sağlar.

Dosya menüsünde yer alan;

Yeni: Yeni bir proje oluşturmak için scratch programını açar.

Şimdi kaydet: Oluşturulmuş olan çalışmayı çevrimiçi hesaba kaydeder.

Kopya olarak kaydet: Yapmış olduğunuz çalışmanın bir kopyasını oluşturur.

Bilgisayarından yükle: Bilgisayarda kayıtlı olan scratch projesini internet ortamına aktarır.

Bilgisayarına kaydet: Çevrimiçi (on-line) olarak yapmış olduğunuz bir çalışmayı bilgisayar ortamına kaydetmeyi sağlar.

Scratch programında hazırlanmış olan dosyaların uzantısı .sb2'dir.

Düzenle menüsünde yer alan;

Geri getir: Silinen karakteri geri getirmek için kullanılır.

Turbo modu aç: Scratch'ın kodlarını daha hızlı çalıştırmasını sağlar.

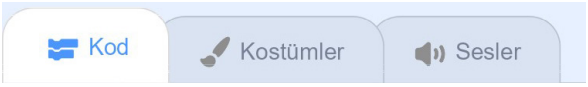
Eğitici Dersler İçeriğinde yer alan örnek projelerin nasıl yapıldığı ile ilgili eğitici derslerin bulunduğu bölümdür.

Üst bölümde yer alan klasör şekline tıklandığında kendinize ait paylaşılmış ve paylaşılmamış olan tüm projeler listelenir. Buradan herhangi bir projenin "İçine bak" denildiği zaman projenin açılması sağlanır.

Başlat(yeşil) ve durdur(kırmızı) düğmeleri: Hazırlanan projenin çalıştırılıp durdurulması için kullanılır.

Kısa yollar: Sahnenin konumunun belirlendiği araçlardan oluşan kısımdır.

Sekmeler: Kod, Kostümler ve Sesler olmak üzere üç sekme vardır. Kod sekmesi altında kodlamada kullanılacak olan bloklar, kostümler sekmesinde kuklarda kullanılacak kostümlerin seçilip düzenlenebileceği alan bulunur. Sesler sekmesinde ise seçili ses ile ilgili ayarların yapıldığı bölüm yer alır.



Görsel 2.6: Sekmeler

Dekor bölümü: Sahnenin arka planının belirlendiği bölümdür.

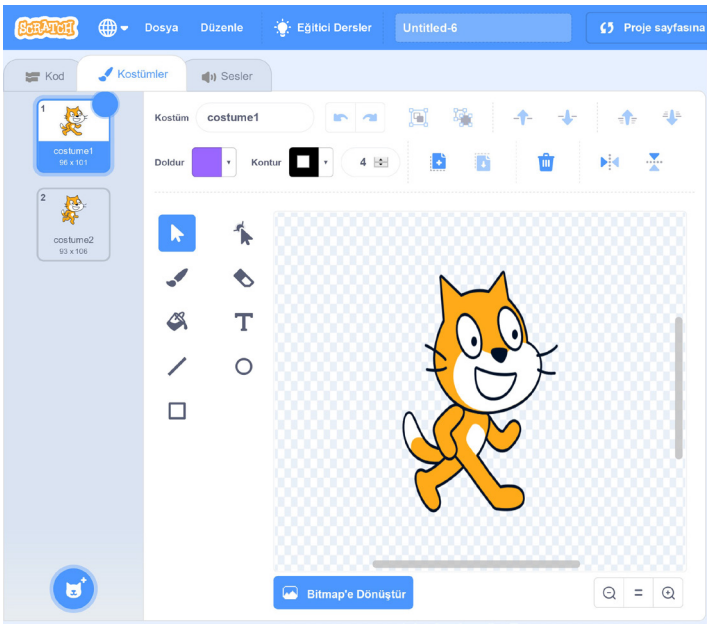
2.2.1. Kuklalar (Karakterler)

Scratch ortamında hazırlanan projeler kukla denilen objelere hayat verilmesiyle oluşturulur. Programın sunduğu hazır kuklalar kullanılabileceği gibi bilgisayarda bulunan bir resim veya nesne de kukla olarak kullanılabilir. Kuklalar bölümünden farklı bir kukla seçilmediği sürece karşımıza çıkan ilk kukla "kedi" dir.

Görsel 2.7: Kedi kuklası

DİKKAT: Kuklalar x ve y doğrularının kesişim noktasında yer alır. Program ilk açıldığı zaman kukla, sahnenin tam ortasında (0,0) konumunda yer alır. Kuklanın sahne üzerinde bulunduğu konum, koordinat sistemi gibi düşünülebilir. Koordinat sistemi yatay ve dikey iki doğrunun kesişiminden oluşur. x ekseni -240 ile 240, y ekseni ise -180 ile 180 arasındadır. Kuklalar ileriye doğru gittiğinde x konumu artar, geriye doğru gittiğinde ise x konumu azalır. Yukarıya doğru gerçekleşen bir harekette y konumu artar, aşağıya iniş durumunda y konumu azalır.

Bazı kuklalar birden fazla kostüme sahiptir. Bir kuklanın hangi kostümlere sahip olduğunu görebilmek ve o kostümlerle ilgili ayarlar yapmak için "Kostümler" sekmesine tıklanır.



Görsel 2.8: Kostüm ekleme ve düzenleme

Kostüm ekleme bölümünden kütüphanede yer alan kuklaların kostümlerinden seçim yapabilir, kendiniz bir kostüm çizebilir, kameranızı kullanarak bir fotoğraf çekip kostüm olarak kullanabilir veya bilgisayarınızdaki bir görseli ekleyerek kostüm olarak kullanabilirsiniz.

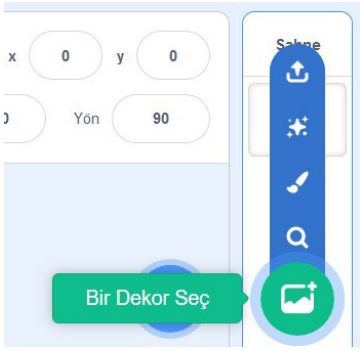
Bir Kostüm Seç simgesine tıklayarak kuklaya farklı kostümler eklenebilir.

Görsel 2.9: Kostüm kütüphanesi

2.2.2. Dekorlar

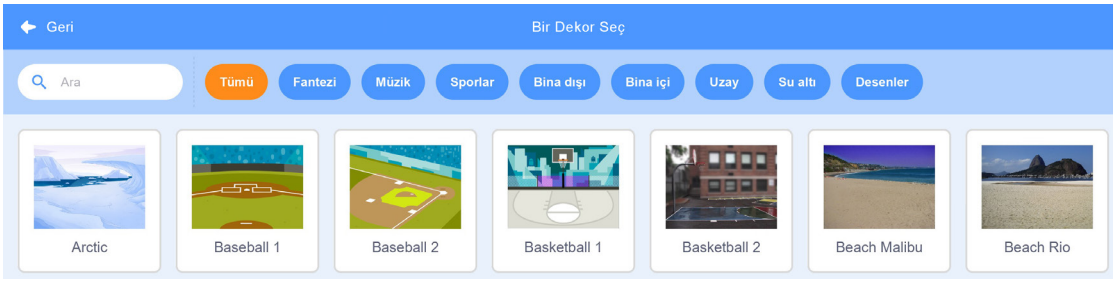
Scratch programında sahnenin arka plan görüntüsüne verilen addır. Sahnenin sağ alt bölümünde, kuklaların yanındaki kısımda yer alır. Dekor ekleyerek sahnenin görüntüsü istenilen şekilde değiştirilebilir.

Dekor eklemek için program kütüphanesinden bir dekor seçebilir, bilgisayarınızda yer alan bir görseli dekor olarak ekleyebilir, dekor çizebilir veya bilgisayarınızın kamerasını kullanarak fotoğraf çekip dekor olarak kullanabilirsiniz.



Görsel 2.10: Dekor ekleme

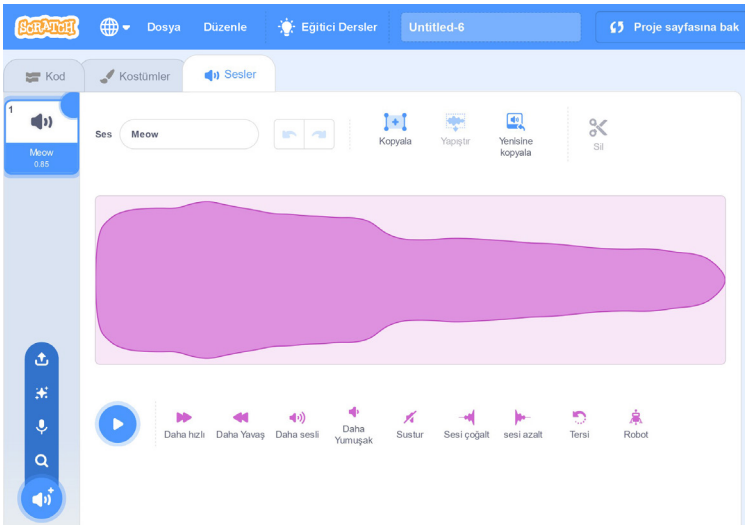
Bir Dekor Seç simgesine tıklandığı zaman listelenen dekor kütüphanesinden projeye uygun dekor seçilebilir.



Görsel 2.11: Dekor kütüphanesi

2.2.3. Sesler

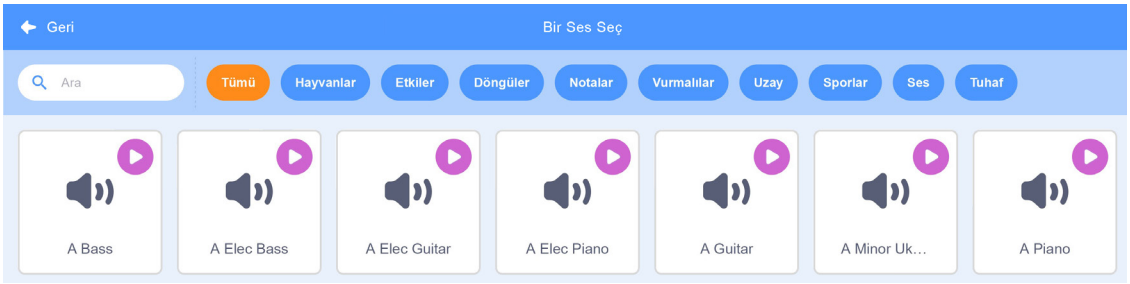
Scratch programında hazırlanan projelerdeki sahne veya karakterlere ses eklemek mümkündür. Bunun için sesler sekmesi kullanılır.



Görsel 2.12: Ses ekleme ve düzenleme

Ses ekleme ve düzenleme ekranını kullanarak kütüphanede yer alan seslerden birini seçebilir, bilgisayarınızda da yer alan bir sesi ekleyebilir veya ses kaydı yapabilirsiniz.

Bir Ses Seç simgesine tıklandığında programdaki seslerin listelendiği ekran açılacaktır. Buradan istenilen ses seçilip kullanılabilir.



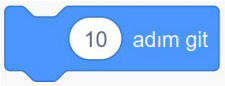
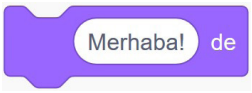
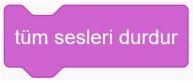
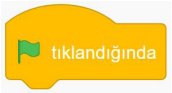
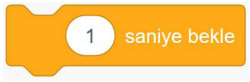
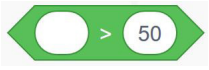
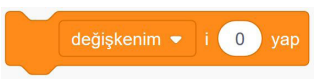
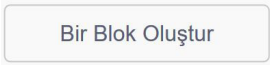
Görsel 2.13: Ses kütüphanesi

2.2.4. Kod Blokları

Scratch'ta hazırlanan projeler kod bloklarının belli bir sıraya göre bir araya getirilmesi ile oluşturulur. Her blok yaptığı işlemlere göre farklı kategorilerde gruplanmıştır.

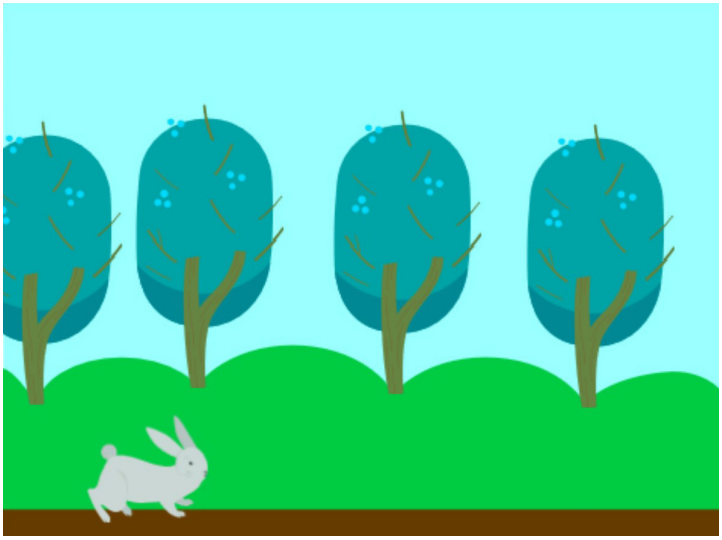
Görsel 2.14: Kod blokları

Tablo 2.1: Blokların Görevleri

BLOK ADI	BLOK ÖRNEĞİ	İŞLEVİ
HAREKET BLOĞU		Kuklaların konumu, yönü ve hareketinin belirlenmesi için kullanılan bloklardır.
GÖRÜNÜM BLOĞU		Kuklaya söyletilmek istenen bir sözün baloncuklar içinde söyletilmesi, kuklanın sahnedeki kılığı veya boyutu gibi görüntü ile ilgili ayarlar için kullanılan bloklardır.
SES BLOĞU		Kuklalara ses eklemek ve seslerin kontrolünü sağlamak için kullanılan bloklardır.
OLAYLAR BLOĞU		Program başlatmak, kuklayı tıklamak, dekor değişikliği, ses yüksekliği ve bir haber alıp haber vermek gibi bir olayın tetiklenmesi gerektiği durumlarda kullanılan bloklardır.
KONTROL BLOĞU		Bir işlemin birden fazla tekrar etmesi, belirlenen süre boyunca beklemesi, akışın kontrol edilmesi, dizilerin durdurulması ve ikiz işlemler için kullanılan bloklardır.
ALGILAMA BLOĞU		Kullanıcıya soru sorduran ve aldığı cevaba göre işlem yaptıran, bir tuşa veya fareye basılı olup olmadığı, bir rengin başka bir renge değip değmediği gibi algılama işlemleri için kullanılan bloklardır.
OPERATÖRLER BLOĞU		Aritmetiksel, mantıksal ve karşılaştırma operatörü işlemleri ile mod alma, birleştirme, yuvarlama gibi matematiksel işlemlerin yapıldığı bloklardır.
DEĞIŞKENLER BLOĞU		Değişkenler oluşturup değişkenlerle ilgili işlemlerin yapıldığı bloklardır.
BLOKLARIM BLOĞU		Sıklıkla kullanılacak işlemler için oluşturulan pembe renkli, kişiye veya programa özel bloklardır.

2.3. Proje Uygulama Örnekleri

Örnek 1: Yeşil bayrak tıklandığında ormanda gezen tavşan uygulamasını yapınız.

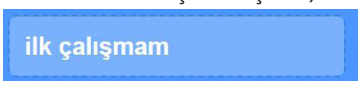


Görsel 2.15: Örnek 1 görseli

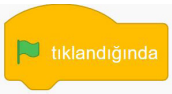
http://kitap.eba.gov.tr/KodSor.php?KOD=22321

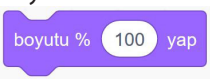
Yönergeler

- Scratch.mit.edu adresine üyelik bilgilerinizle giriş yapıp Oluştur düğmesine tıklayarak yeni bir proje oluşturunuz. Masaüstü uygulaması ile çalışacaksanız Dosya menüsünden Yeni komutunu seçerek yeni bir proje sayfası açınız.
- Üst menüden projenizin adını "ilk çalışmam" olarak belirleyiniz (Yeni bir proje açıldığında, projenin ismi Untitled olarak karşımıza çıkar.).



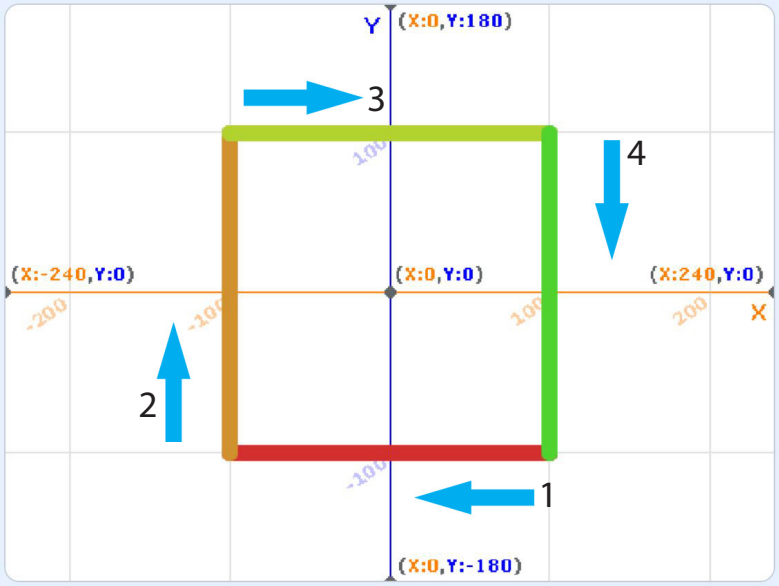
- Görselde yer alan dekoru sahneye ekleyiniz.
- Kedi kuklasını siliniz [çöp (x) simgesine tıklayınız.].

- Kütüphanede yer alan tavşan ve ağaç kuklalarını sahneye ekleyiniz.
- Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundaki 

kod bloğunu kodlama alanına sürükleyiniz.
- Görünüm bloğundan 

komutunu seçerek tavşanın boyutunu %50 küçültünüz.
- Hareket bloğunu kullanarak tavşanın konumunu x:-193, y:-133 olarak ayarlayacak komutu seçiniz.
- Program durduruluncaya kadar işlemin devam etmesi için Kontrol kod bloklarının altında yer alan "Sürekli tekrarla" komutunu ekleyip tavşanın hareketlerini belirleyecek komutları bu blok arasına yazınız.
- Görünüm bloğundan "Sonraki kostüm" komutunu seçiniz. Bu komut ile kuklanın farklı kostümleri arasında sırasıyla geçiş sağlanacaktır.
- Hareket bloğundan "10 adım git" komutunu seçiniz.
- Kontrol bloğundan "1 saniye bekle" komutunu seçip süreyi 0.25 olarak değiştiriniz.
- Hareket bloğundan sırası ile "Kenara geldiyse sektir" ve "dönüş stilini sol-sağ yap" komutlarını seçerek tavşanın sahnenin sonuna geldiğinde geriye dönüş yapmasını sağlayınız.

Örnek 2: Yeşil bayrağa tıklandığında koordinat düzleminin (100,-100) noktasından başlayarak 200x200 pixel(adım) ölçülerinde ve her kenarı farklı renk olan bir kare çizdiren uygulamayı yapınız.



Görsel 2.16: Örnek 2 görseli

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

Yönergeler

"Renkli kare" adında yeni bir proje oluşturunuz.

Sahneye xy-grid dekorunu ekleyiniz.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

Kalem bloğunu, bloklarınız arasına ekleyiniz. Bunun için kod blokları kategorisinin alt tarafında yer alan  simgesine tıklayınız ve açılan pencereden kalem  eklentisini seçiniz.

Kuklanın ekranda görünmemesi için Görünüm bloğundan "Gizle" kod bloğunu kodlama alanına sürükleyiniz.

Önceden kalan çizimler olmaması için Kalem bloğundan "tümünü sil" komutunu seçip kalemi temizleyiniz.

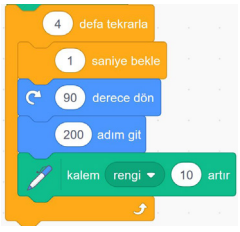
"Kalemi kaldır" komutunu seçiniz.

Kalem kod bloğundan kalem rengi ve kalınlığını belirleyiniz.

Çizimin başlayacağı konumu belirlemek için hareket bloğundan x ve y değerlerini belirleyecek komutu seçiniz. (X:100,y:-100) yapınız.

Çizim yapmak için Kalem bloğundan "kalemi bastır" komutunu seçiniz.

Çizeceğimiz şekil kare olduğundan Kontrol bloğu altında yer alan "10 defa tekrarla" komutunu seçip 10 sayısını 4 olarak değiştiriniz. Blok içindeki komutların 4 defa tekrar etmesi sağlanacaktır.

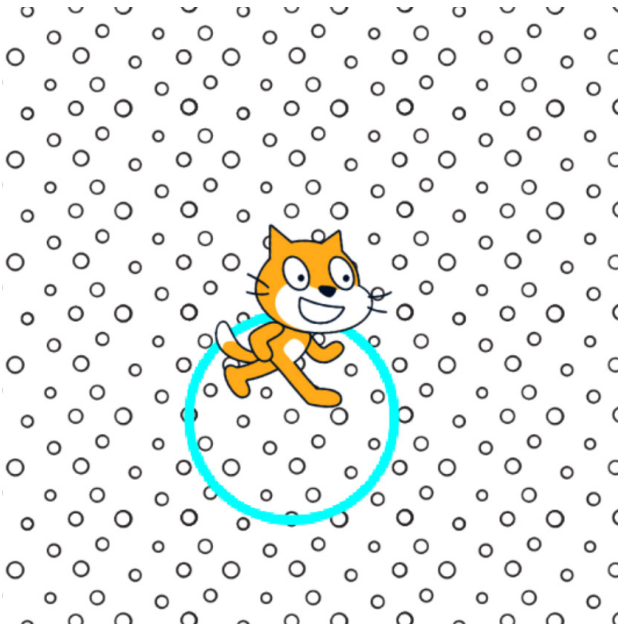


Kontrol bloğundan "1 saniye bekle" kod bloğunu seçerek çizimi daha rahat görebilmeyi sağlayınız.

Hareket bloğundan "90 derece dön" komutunu seçerek çizmeye başlayacağınız yönü belirleyiniz. Hareket bloğundan, belirlenen yöne kaç adımlık (pixellik) çizileceğini seçiniz.

Kalem bloğundan, "kalem rengi 10 artır" komutunu seçerek her kenarın farklı renkte olmasını sağlayınız.

Örnek 3: Kediye (0,0) noktasından başlayarak bir daire çizdiren scratch uygulamasını hazırlayınız.



Görsel 2.17: Örnek 3 görseli

Yönergeler

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

"Daire çizen kedi" adında yeni bir proje oluşturunuz.

Sahneye resimdeki "circles" dekorunu ekleyiniz.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

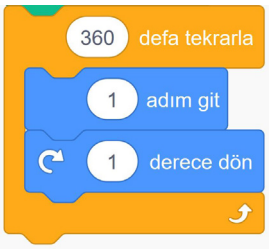
Çizimin başlayacağı noktayı belirlemek için Hareket bloğundan "x :0, y:0 konumuna git" komutunu seçiniz.

Önceden kalan çizimler olmaması için Kalem bloğundan "tümünü sil" komutunu seçip kalemi temizleyiniz.

Kalem bloğunu kullanarak sırasıyla kalem kalınlığını 5, kalem rengini 50 olarak ayarlayınız.

Çizim yapmak için Kalem bloğundan "kalemi bastır" komutunu seçiniz.

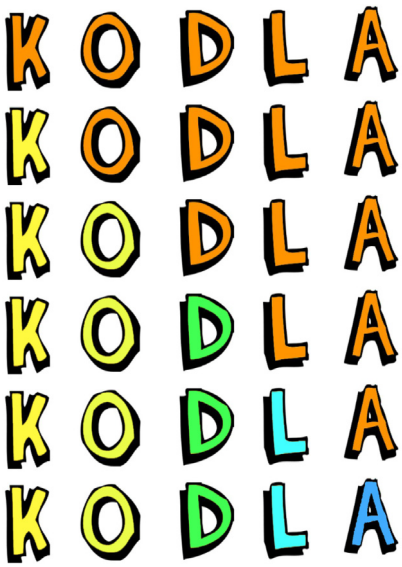
Kalemin her adımda 1 adım (pixel) ilerleyip 1 derece dönmesi ve bunu 360 defa tekrar etmesi için Kontrol ve Hareket bloklarından yararlanarak kalemi bastır komutunun altına aşağıdaki blok kodları yerleştiriniz.



 simgesine tıklayıp programı çalıştırınız.

Sıra Sizde: Yeşil bayrağa tıklandığı zaman tüm kenar renkleri birbirinden farklı olan bir üçgen çiziniz.

Örnek 4: Yeşil bayrağa tıklandığında "Kodla" yazısının harflerindeki renklerin tek tek değişmesini sağlayan scratch animasyon uygulamasını yapınız.



Görsel 2.18: Örnek 4 görseli

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

Yönergeler

"Renkli yazı" adında yeni bir proje oluşturunuz.

Kedi kuklasını silip "K-O-D-L-A" kuklalarını sahneye y:0 olacak şekilde yan yana ekleyiniz.

Block-K kuklası seçili iken aşağıdaki işlemleri sırasıyla yapınız.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

Tıklandığında bloğunun altına önceden uygulanmış olan görsel efektlerin temizlenmesi için görünüm bloğunun altında yer alan "görsel etkileri temizle" bloğunu ekleyiniz.

Kontrol bloğundan 1 saniye bekle seçildikten sonra renk etkisini %5, parlaklık etkisini %25 değiştirecek komutları alt alta ekleyiniz.

Tüm kuklalar için aynı komutları ufak değişiklikler yapıp kullanmak için yukarıdaki blokları sırt çantasına sürükleyiniz. Daha sonra sırt çantasındaki kodların her bir kuklanın kodlama alanına sürükleyip eklenmesini sağlayınız.

"K" harfi için olan 1 saniye bekle ve renk etkisini 5 değiştir komutlarını, "O" harfi için 2 saniye ve 25, "D" harfi için 3 saniye ve 50, "L" harfi için 4 saniye ve 75, "A" harfi içinse 5 saniye ve 100 yapıp renk ve süre geçişlerini sağlayınız.

 simgesine tıklayıp animasyonu çalıştırınız.

DİKKAT: Sadece çevrimiçi editörde yer alan sırt çantası özelliği ile farklı çalışmalarda veya karakterlerde kullanılmak istenen kodlar, sırt çantası bölümüne taşınarak daha sonra kullanılmak üzere saklanabilmektedir.

Örnek 5: Yeşil bayrağa tıklandığında eklediğiniz Dragonfly kuklasının fare hareket yönüne göre imleci takip etmesini sağlayan scratch uygulamasını yapınız.

Görsel 2.19: Örnek 5 görseli

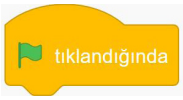
Yönergeler

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

"Kelebek" adında yeni bir proje oluşturunuz.

Dekor ekleyiniz.

Kedi kuklasını siliniz ve dragonfly kuklasını ekleyiniz.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

Kelebeğin boyutunu %50 oranında küçültünüz.

 düğmesi tıklanana kadar olay gerçekleşeceğinden Kontrol bloğundan "Sürekli tekrarla" komutunu seçiniz ve bu komut arasına aşağıdaki komutları yazınız. Böylelikle 2 farklı kostüme sahip dragonfly kuklası imleç hareketine göre şekil ve pozisyon değiştirecektir.

Sıra Sizde: Klavyeden boşluk tuşuna basılmasıyla birlikte "Butterfly 1" kuklasının sürekli olarak kostüm değiştirip rastgele konumlarda uçmasını sağlayan scratch uygulamasını yapınız.

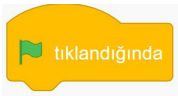
Örnek 6: Yeşil bayrağa tıklandığında davul eşliğinde dans edip dans sırasında klavyeden boşluk tuşuna basıldığında renk değiştiren kedi uygulamasını yapınız.

Görsel 2.20: Örnek 6 görseli

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

Yönergeler

"Dans eden kedi" adında yeni bir proje oluşturunuz.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

Görseldeki dekoru sahneye ekleyiniz.

Sahneye dekor amaçlı kullanılacak olan araba kuklasını ekleyiniz.

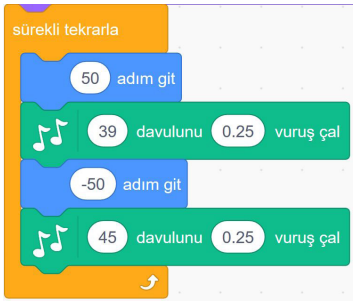
Kedi kuklasının proje başladığı zaman ilk konumlanacağı yeri belirlemek için Hareket bloğunu kullanarak x konumunu -172 ve y konumunu -96 olarak ayarlayınız.

Kuklanın iki saniye boyunca ekrana "Dansımı izlemeye ne dersiniz?" mesajını vermesini sağlayınız. Bunun için Görünüm bloğunu kullanınız.

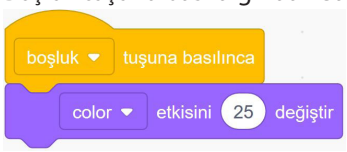
Çalgıları ve davulları çalabilmek için Müzik bloğunu, bloklarınız arasına ekleyiniz. Bunun için kod blokları kategorisinin alt tarafında yer alan  simgesine tıklayınız ve açılan pencereden müzik  eklentisini seçiniz.

Mesajın ardından Kontrol kod bloklarının altında yer alan Sürekli tekrarla kod bloğunu ekleyiniz. Bu blok arasına eklenen kodlar program durdurulunca kadar çalışacaktır.

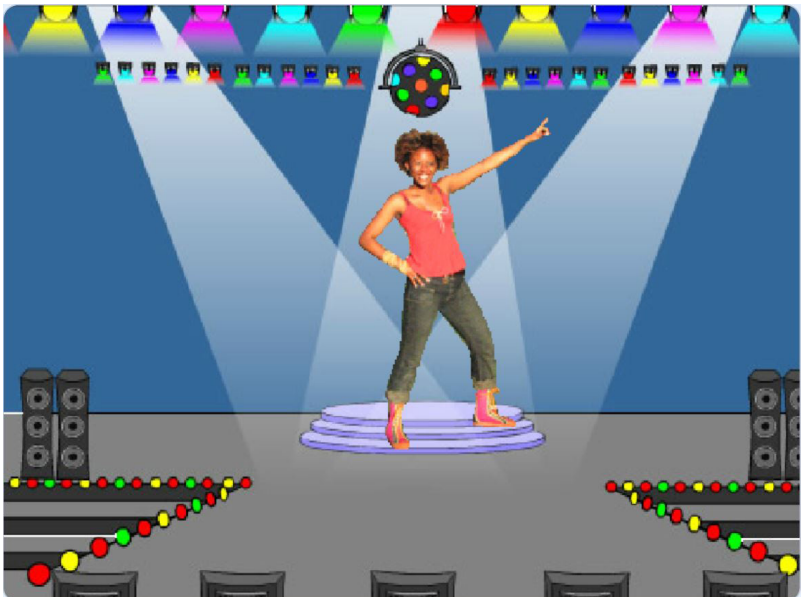
Davul ve vuruş ayarları ile ileri ve geri gitmeyi sağlayacak komutları, sürekli tekrarla bloğunun içine yerleştiriniz.



Boşluk tuşuna basıldığında kedi kuklasının renginin değişmesi için aşağıdaki blokları ekleyiniz.



Örnek 7: Müzik eşliğinde dans eden kadın uygulamasını yapınız.



Görsel 2.21: Örnek 7 görseli

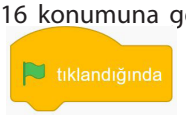
Yönergeler

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

"Dans" adında yeni bir proje oluşturunuz.

Resimdeki dekoru sahneye ekleyiniz.

Kedi kuklasını siliniz ve resimdeki "Cassy Dance" kuklasını sahnede x: 26, y: 16 konumuna gelecek şekilde yerleştiriniz.

Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.

4 farklı kostüme sahip olan kuklamızın program her çalıştığında Cassy-a kostümünde başlaması için Görünüm bloğundan "cassy-a kuklasına geç" komutunu kod bloklarının altına yerleştiriniz.

2 saniye boyunca kuklanın "hadi dans edelim:)" mesajının görünmesini sağlayacak komutu Görünüm bloğundan ekleyiniz.

Kontrol bloğunda yer alan sürekli tekrarla bloğu arasında 0.5 saniye aralıklarla kostüm ve renk etkisi komutu kullanarak kuklanın kostüm rengini değiştirmeyi sağlayacak kodları yazınız.

"Cassy Dance" kuklası ile ilgili olaylar gerçekleştiği süre boyunca arka planda ses çalması için ayrı bir blok grubu oluşturmak gerekir. Ses eklemek için Sesler sekmesinden bir ses seç  düğmesine tıklanır.

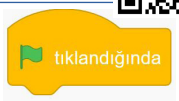
İstenilen ses seçildikten sonra Ses bloğu kullanılarak "….. sesi bitene kadar çal" komutu Sürekli tekrarla bloğu arasına yerleştirilir.

Örnek 8: Yeşil bayrağa tıklandığında girilen sayıların eşit olup olmadığı mesajını veren scratch uygulamasını yazınız.

Görsel 2.22: Örnek 8 görseli

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

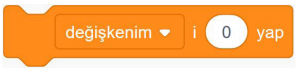
Yönergeler

- "Sayıların eşitliği" adında yeni bir proje oluşturunuz.
- Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

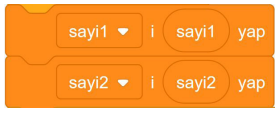
kod bloğunu kodlama alanına sürükleyiniz.
- İki sayının karşılaştırmasını yapmak için sayi1 ve sayi2 adında iki değişken tanımlayınız. Bunun için Değişkenler bloğundan 

komutunu seçiniz.

Görsel 2.23: Değişken ekleme

- Görünüm bloğundan "Merhaba de" komutunu çalışma alanına ekleyip "Sayı değerlerini belirleyiniz." şeklinde değiştiriniz.
- Kontrol bloğundan "1 saniye bekle" kod bloğunu seçip 10 saniye olarak değiştirerek karşılaştırılacak sayı değerlerini ayarlamak için süre tanıyınız.
- Değişkenler bloğunda yer alan 

komutunu kodlama alanına sürükleyip aşağıdaki şekilde değiştiriniz.



- Sayi1'in sayi2 ile eşit olup olmadığının kontrolünü yapmak için Kontrol bloğundan "Eğer-İse" karar ifadesini seçip aşağıdaki gibi düzenleyiniz.

DİKKAT: Birden fazla kukla kullanılan durumlarda hangi kukla ile ilgili eylem gerçekleşecekse o kukla seçilir ve gerçekleştirilmesi istenen eylemler yazılır.

Sıra Sizde: Girilen iki sayının toplamını ve çarpımını bulduran scratch uygulamasını yapınız.

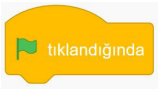
Örnek 9: Yeşil bayrağa tıklandığında sayı tahmin oyununu başlatan scratch uygulamasını yapınız.

Not: Aklınızdan bir sayı tutup arkadaşlarınızdan bu sayıyı tahmin etmelerini isteyiniz. Tutulan sayı 1 ile 50 arasında olmalı ve herkesin sadece beş kez tahmin hakkı bulunmalıdır.

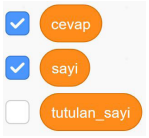
Görsel 2.23: Örnek 9 görseli

Yönergeler

http://kitap.eba.gov.tr/KodSor.php?KOD=22321

- "Sayı tahmini" adında yeni bir proje oluşturunuz.
- Resimdeki dekoru sahneye ekleyiniz.
- Kedi kuklasını silip "Dee" kuklasını ekleyiniz.
- Yeşil bayrağa tıklandığı zaman olay gerçekleşeceğinden Olaylar bloğundan 

kod bloğunu kodlama alanına sürükleyiniz.
- Bilgisayarın rastgele oluşturacağı sayı için "tutulan_sayi", tahmin sayınızı tutmak için "sayi" ve yaptığınız tahminleri tutmak için ise "cevap" adında üç değişken oluşturunuz.
- Değişkenler bloğundaki değişkenleri aşağıdaki gibi düzenleyiniz. Böylelikle bilgisayarın tuttuğu sayının ekranda görünmesi engellenmiş olacaktır.

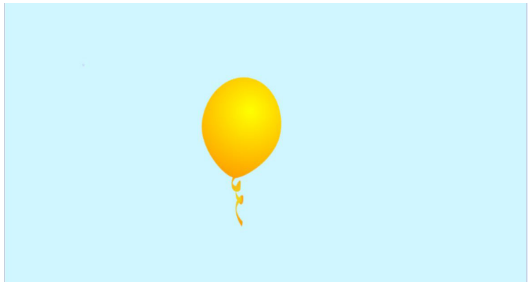


- Sayi1 ve cevap değişkenlerini 0 (sıfır) yapınız.
- Bilgisayarın 1 ile 50 arasında rastgele sayı üretmesini sağlamak için Operatörler bloğundan "1 ile 10 arasında rastgele sayı seç" komutunu seçip düzenleyiniz.
- Değişkenler bloğunu kullanarak "Tutulan_sayi" değişkenini 

yapınız.
- Kuklanın ekrana mesaj vermesini sağlayınız.
- Sayi1 sayacını 1 arttırınız.
- Bilgisayarın tuttuğu sayı ile verilen cevap eşit olana kadar ve tahmin sayısı 5'i geçmediği sürece tutulan_sayi değişkeni ile verilen cevabı karşılaştırınız. Beş adet sayı girildiği hâlde girilen sayı tahmin edilememişse bilemediniz mesajını verip tutulan sayının ne olduğunu ekrana yazdırınız.

Örnek 10: Balon yakalama oyunu

Yeşil bayrağa tıklandığında sahnede rastgele uçan balonun, üzerine her tıklandığında renginin değişip ses çıkmasını ve puanın artmasını sağlayan uygulamayı yapınız.



Görsel 2.24: Örnek 10 görseli

Yönergeler

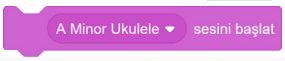
http://kitap.eba.gov.tr/KodSor.php?KOD=22321

- "Balon yakalama" adında yeni bir proje oluşturunuz.
- Sahneye bir dekor ekleyiniz.
- Sahneye balon kuklası ekleyiniz.
- Öncelikle balonun oyun durdurulana kadar sahnede rastgele gezmesini sağlamak için aşağıdaki komutları yazınız.

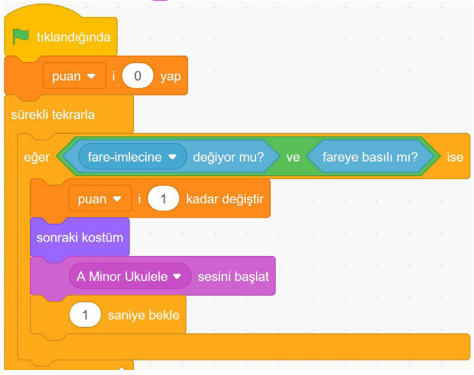
- "0.5 saniye bekle" komutu ile balon hızlı hareket edecektir. Balonun daha yavaş gezinmesi istenirse bu rakamı arttırabilir, çok daha hızlı hareket etmesi için değer azaltılabilir.
- Balona her tıklandığında bir puan alınmasını sağlamak için "Puan" isminde bir değişken oluşturunuz.
- Puanı 0 yapmak için Değişkenler bloğunu kullanınız.
- Oyunda puan alabilmek için hem imlecin balon üzerine değmesi hem de farenin tıklanması gibi iki koşulun aynı anda gerçekleşmesi için Operatörler bloğundan "ve" operatörünü kullanınız.
- Farenin kuklayla etkileşimi ve fareye tıklanıp tıklanmadığının kontrolünü yapmak için Algılama bloğunu kullanınız.

- **Her iki şartın aynı anda gerçekleşmesi hâlinde Eğer kontrol bloğu içinde yer alan komutlar çalışacaktır.
- Balona tıklandığı zaman puanı 1 sayı arttırmak için Değişkenler bloğundan 

komutunu kullanınız.
- Balonun renginin değişmesi için "sonraki kostüm" komutunu uygulayınız.
- Sesler sekmesini kullanarak 

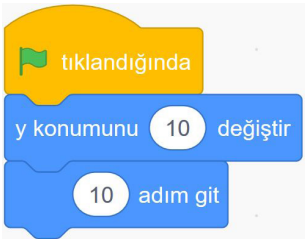
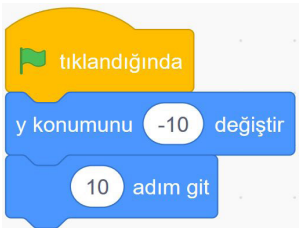
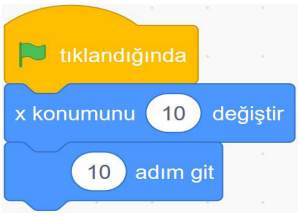
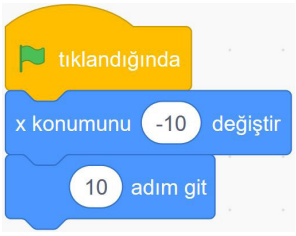
sesini seçiniz. Her puan alındığında bu sesin de çalması için Sesler bloğundan 

komutunu diğer blokların altına yerleştiriniz.



ÖLÇME VE DEĞERLENDİRME 2

Aşağıdaki tablodaki sol sütunda scratch kodları, sağ sütunda ise kodlar çalıştığında gerçekleşen olaylar gösterilmiştir.

1. 	a) Nesneyi sola doğru 10 adım hareket ettirir.
2. 	b) Nesneyi sağa doğru 10 adım hareket ettirir.
3. 	c) Nesneyi aşağıya doğru 10 adım hareket ettirir.
4. 	d) Nesneyi yukarıya doğru 10 adım hareket ettirir.

1. Buna göre sütunların eşleşmesi aşağıdakilerden hangisinde doğru gösterilmiştir?

   A) 1-a, 2-b, 3-c, 4-d B) 1-b, 2-c, 3-a, 4-d C) 1-a, 2-d, 3-b , 4-c

   D) 1-d, 2-a, 3-c, 4-b E) 1-d, 2-c, 3-b, 4-a

2. Blok tabanlı programa kaydolurken aşağıdaki bilgilerden hangisi istenmez?

   A) Yaşanılan ülke adı B) Cinsiyet C) Baba adı

   D) Doğum tarihi E) E-posta adresi

3. Seçilen karakterin boyutunu değiştirmek için kullanılan komut hangi kod bloğunda yer alır?

   A) Görünüm B) Kontrol C) Olaylar D) Algılama E) Hareket

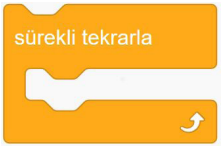
4. Scratch programında hazırlanmış olan projelerin ön izlemesinin yapıldığı bölüm aşağıdakilerden hangisidir?

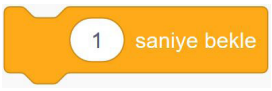
   A) Menüler B) Bloklar C) Kodlama alanı

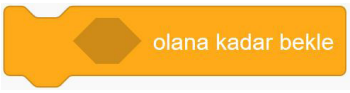
   D) Sahne E) Kuklalar

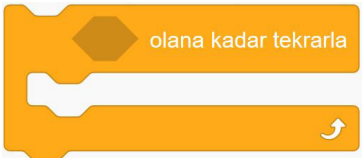
5. Scratch programında verilen bir işlemin 5 kez tekrar etmesi isteniyorsa kullanılması gereken komut aşağıdakilerden hangisi olmalıdır?

A) 

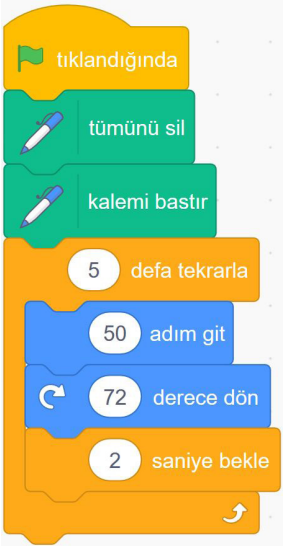
B) 

C) 

D) 

E) 

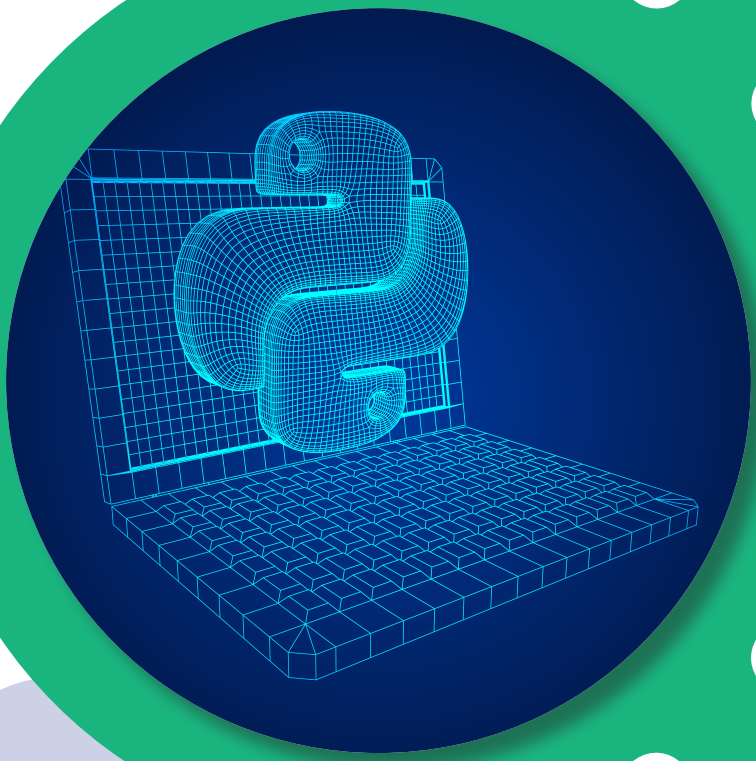
Aşağıda scratch programında yazılmış kod blokları verilmiştir. Buna göre;



6. Kodlar çalıştırıldığı zaman aşağıdaki şekillerden hangisi çizilmiş olur?

A) Daire B) Üçgen C) Kare D) Dikdörtgen E) Çokgen

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.



ÖĞRENME BİRİMİ 3

PROGRAMLAMA DİLİ TEMELLERİ

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Program, yazılım ve programlama dili kavramlarını açıklayabilecek,
- Yorumlama ve derleme kavramlarını tanımlayabilecek,
- Python programlama dilinin avantajlarını sıralayabilecek,
- Python ile yapılabilecek proje fikirleri geliştirebilecek,
- Python kurulumu yapabilecek,
- IDLE üzerinde kod çalıştırabilecek,
- Python için gerekli araçları belirleyip kurulumlarını yapabileceksiniz.

Anahtar Kelimeler:

Program, yazılım, programcı, yorumlama, derleme, IDLE, IDE.

Hazırlık Çalışmaları

1. Son yıllarda dünya genelinde yazılımcılar tarafından en çok hangi programlama dillerinin tercih edildiğini araştırınız ve sebeplerini tartışınız.
2. Açık kaynak ve kapalı kaynak kod kavramlarını araştırınız.

3. PROGRAMLAMA DİLİ TEMELLERİ

3.1. Program ve Yazılım

Görsel 3.1: Program ve yazılım

Program, herhangi bir elektronik cihaza işlem yaptırmak için yazılan komutlar dizisidir.

Elektronik cihazlara; bilgisayar, cep telefonu, akıllı saat, akıllı televizyonlar vb. örnek olarak verilebilir. Program yazımı denilince akla ilk gelen cihazlar bilgisayarlar olsa da günümüzde pek çok elektronik cihaza kod yazılabilir.

Çoğu zaman "program" kelimesi ile "yazılım" kelimesi birbirinin yerine kullanılmaktadır. Ancak bu doğru değildir. Yazılım, programa göre daha geniş kapsamlıdır. Programlar tek başlarına yazılım olarak adlandırılamaz. Yazılım denildiğinde; programın yanında belgeleme, yardım dosyaları, veri kaynakları ve yardımcı programlar akla gelmektedir. Örnek olarak, bilgisayarınızda bulunan "Hesap makinesi" bir yazılımdır. "Hesap makinesi" yazılımını kullanabilmek için çalıştırılan "calc.exe" ise programdır.

3.2. Programlama Dili

Görsel 3.2: Programlama dili örneği

Tüm elektronik sistemler "1" ve "0"lardan oluşan kod bloklarını çalıştırır. "1" ve "0"lardan oluşan kodlama diline "makine dili" denir. Aşağıda makine dilinde yazılmış bir kod parçası görülmektedir.

```
11010011101000000
00000000000010001
11011000001100111
00110001100111010
11001111001010111
```

Bu kod parçası bilgisayarda bulunan "Merkezi İşlem Birimi (CPU)" için pek çok şey ifade etse de bizler için çok anlamlı gelmemektedir.

Bundan dolayı makine dili bir adım ileriye taşınarak "Assembly dilleri" oluşturulmuştur. Assembly dili, daha okunaklı olarak yazılan kodları makine diline dönüştürür. Aşağıdaki Assembly kod bloğu örneğini inceleyebilirsiniz.

```
mov r0,#1
mov r1,#1
l:
add r2,r0,r1
str r2,[r3]
add r3,#4
mov r0,r1
mov r1,r2
b l
```

Görüldüğü üzere, Assembly dilinde yazılan programlar da kolay anlaşılır değildir.

İşte bu noktada kolay yazılabilir ve anlaşılabilir kodlar oluşturmak üzere "Programlama Dilleri" geliştirilmiştir. Günlük hayatta kullanılan İngilizce ile programlar yazılmakta ve bu dilin makine diline çevrimi, arka planda otomatik olarak gerçekleşmektedir.

```
a = 5
b = 8
if a + b > 10:
print "toplam, 10 dan büyüktür."
else:
print "toplam, 10 dan büyük değildir."
```

Programlama dilini hiç bilmeyen biri bile bu kod parçasının ne amaçla yazıldığını tahmin edebilir. İşte bu yüzden program geliştirmek için "Programlama dilleri" kullanılmaktadır.

Programlama dili, belirli bir algoritmaya dayalı olarak bir yazılım programı oluşturmak için kullanılan, sıkı kurallara sahip, bir dizi komut ve talimatlar bütünüdür. Günümüzde, kullanıma sunulmuş onlarca programlama dili vardır. Her bir problem / algoritma için hangi programlama dilinin kullanılacağı tamamen programcıya kalmıştır.

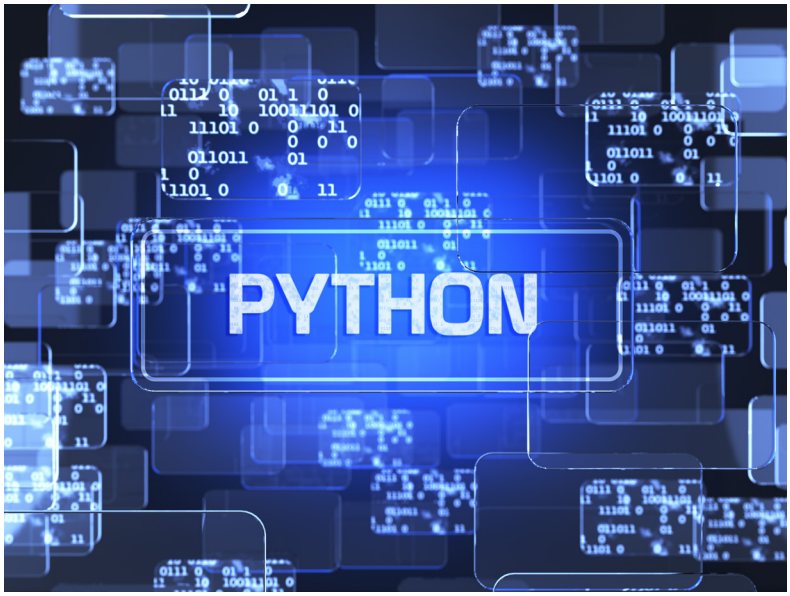
Teorik olarak şu söylenebilir, herhangi bir "A" noktasından "B" noktasına gidilmek istendiğini düşününüz. Bu işlem yürüyerek, bisikletle, otomobille veya uçakla gerçekleştirilebilir. Sonuçta A noktasından B noktasına varılmış olur. Ancak maliyet, zaman, konfor gibi bazı parametreler açısından her bir yöntemin artıları ve eksileri vardır. Yürüyerek veya bisikletle gidildiğinde maliyet sıfıra yakın olurken zaman ve konfor konusunda ciddi sıkıntı yaşanır. Diğer taraftan otomobille daha kısa sürede varılabilir ancak bu sefer de maliyet yükselecektir. Uçak kullanıldığında süre daha çok kısaltılabilir ve yüksek konfor sağlanabilir fakat maliyet daha çok yükselecektir.

Özetle; en iyi denilebilecek programlama dili yoktur. Önemli olan önümüzde duran problem veya algoritma için en uygun programlama dilini seçmektir.

Python'un Kısa Hikâyesi

Python, son zamanlarda popüler bir dil hâline gelse de aslında yeni bir dil değildir. 1990'lı yılların başında "Guido van Rossum" isimli Hollandalı bir yazılımcı tarafından geliştirilmeye başlanmıştır. Çoğu kişi Python dilinin adını piton yılanından aldığını düşünür. Ancak gerçek böyle değildir. Python dilini geliştiren Guido van Rossum bu dili, "The Monty Python" adlı bir komedi grubunun "Monty Python's Flying Circus" adlı gösterisinden esinlenerek isimlendirmiştir. Hâl böyle olsa da pek çok Python kitabının kapağında çeşitli piton yılanı figürlerini görmek artık sıradan bir durumdur.

3.3. Neden Python?



Görsel 3.3: Python'a giriş

Python programlama dilinin basit ve temiz bir söz dizimi vardır. Bu özelliğinden dolayı program yazmak, yazılan programı okumak ve anlamak diğer dillere nazaran daha kolaydır.

Python'un önemli bir özelliği de pek çok dilin aksine "yorumlanan" bir dil olmasıdır. Bu dilde yazılan kodlar derlenmeden direkt çalıştırılır. Python, bu özelliği ile teknik olarak bir programlama dili değil, bir betik (script) dilidir. Python'da hızlı bir şekilde program geliştirilebilir. Bu noktada yorumlama ve derleme olaylarını kısaca açıklamak faydalı olacaktır.

Yorumlama (Interpretation) işlemi, yazılan kodun satır satır okunup bilgisayarın işlemcisine özel makine diline anında çevrilmesi işlemidir. Program her çalıştırıldığında yorumlama işlemi tekrardan yapılır. Bundan dolayı derlenen yazılımlara göre bir nebze yavaş çalışacağını söylemek mümkündür. Bağımsız platform desteği sağlanması sayesinde hazırlanmış olan program, desteklenen her ortamda kolaylıkla çalıştırılabilir. Bu sayede yazılan programın boyutu da küçük olmaktadır.

Yorumlanarak çalıştırılan yüksek seviyeli diller doğrudan yorumlanmaz. Genellikle bir ara forma (Opcode, Bytecode vs.) dönüştürülür ve bu kodlar yorumlanarak yerel makine diline çevrildikten sonra işletilir. Java, PHP, Python gibi yorumlanan diller aslında yorumlama aşamasına geçilmeden önce en az bir kere derlenir.

Derleme (Compilation) işlemi; yüksek seviyeli bir dilde yazılan programın bir başka hedef dile veya makine diline çevrilmesi işlemidir. Programı çalıştırmak için bir kere derleme işleminden geçirmek yeterlidir. Program her çalıştırılışında tekrardan derleme olayı gerçekleşmez.

Direkt makine dili veya çevrilme işleminde, platform bağımlılığından bahsedilmesi gerekir. Program her işletim sisteminde veya işlemcide çalıştırılmak isteniyorsa her işletim sisteminde veya işlemcide ayrı ayrı derleme işleminden geçirmek gerekir. Örnek: C, C++… vb.

Python'da ve oldukça eski, popüler bir dil olan C++'ta birer klasik "Merhaba Dünya" uygulaması yazarak aralarındaki farkı inceleyebilirsiniz.

C++:

```
#include <iostream>;
using namespace std;
int main(){
    cout << "Merhaba Dünya!" << endl;
    return 0;
}
```

Aynı programı Python'da aşağıdaki şekilde yazmak mümkündür.

```
print("Merhaba Dünya!")
```

Her iki program da ekrana "Merhaba Dünya!" yazdırmaktadır. Ancak birinde 6 satır, diğerinde sadece 1 satır kod yazılmıştır. Bu örneklere bakıldığında Python'un son derece basit ve hızlı bir şekilde kod geliştirilebilen bir programlama dili olduğu görülmektedir. Python, birçok işletim sisteminde sıkıntısız bir şekilde çalıştırılabilmektedir: Windows, Linux / Unix, MacOS X ve daha fazlası.

Python'u farklı kılan bazı özellikler şu şekilde listelenebilir:

- Açık kaynak kodlu olması
- Ücretsiz olması
- Hızlı ve kolay kurulabilmesi
- Sade ve kolay okunabilen kod yapısı
- Toplu ve düzenli kod yapısı
- Öğrenme ve adapte olma kolaylığı
- Kolay anlaşılır nesne tabanlı programlama özellikleri
- Güçlü ifade yeteneği
- Son derece esnek modüler yapısı
- "Exception" tabanlı hata yönetimi
- Yüksek seviye dinamik veri yapıları
- Oldukça geniş standart kütüphanelerinin olması
- Otomatik hafıza temizliği
- C, C++, Java ile kolay entegre edilebilmesi
- Hemen her tür platformda sıkıntısız çalışması
- Az kod / çok iş anlayışı

3.4. Python ile Neler Yapılabilir?

Python dili, yukarıda sayılan avantajları sayesinde büyük küçük pek çok şirketin kullandığı bir dildir. Dev sosyal medya şirketleri her zaman Python programcılarına ihtiyaç duymaktadır. Bu nedenle Python dilinin popülaritesinin son yıllarda arttığı söylenebilir. Bazı internet sitelerinde yıllara göre programlama dilleri popülarite indeksi yayınlanmaktadır. Bu sitelerden birisi incelendiğinde aşağıdaki gibi bir liste oluştuğu görülür:

May 2020	May 2019	Change	Programming Language
1	2	^	C
2	1	v	Java
3	4	^	Python
4	3	v	C++
5	6	^	C#
6	5	v	Visual Basic
7	7		JavaScript
8	9	^	PHP
9	8	v	SQL
10	21	^^	R

Görsel 3.4: Mayıs 2019 ve Mayıs 2020 ayları itibariyle en çok tercih edilen programlama

Liste incelendiğinde Python'un Mayıs 2020 itibariyle dünyada en çok kullanılan 3. programlama dili konumunda olduğu görülür.

Aşağıda Python dilinin kullanıldığı alanlar listelenmiştir. Bu liste Python'un neden bu kadar popüler olduğunu açıklamaktadır.

Yapay zekâ ve makine öğrenmesi
Web uygulamaları
Bilimsel hesaplamalar
Veri analizi
Masaüstü uygulama geliştirme
Ağ ve soket programlama
Nesnelerin interneti
Kriptoloji
Sistem yönetimi
Oyun geliştirme vb.

Görüldüğü üzere, Python ile çok çeşitli alanlarda uygulama geliştirmek mümkündür. Sonuç olarak denilebilir ki Python, öğrenilmesi ve kullanılması kolay, neredeyse her iş için bir kütüphanesi olan güçlü bir programlama dilidir.

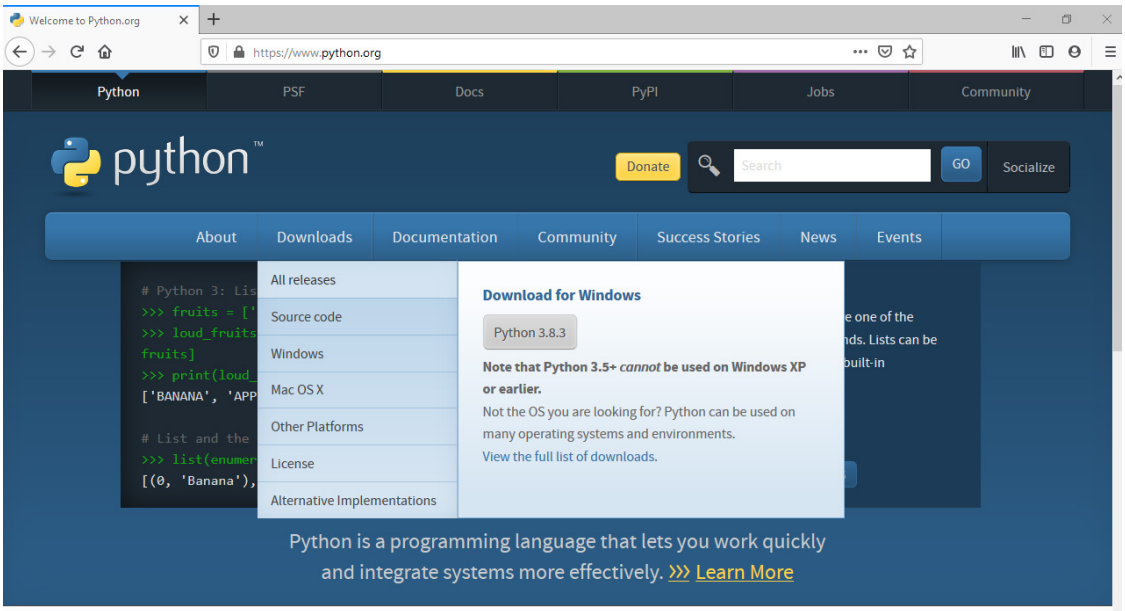
3.5. Python Kurulumu

http://kitap.eba.gov.tr/KodSor.php?KOD=22324

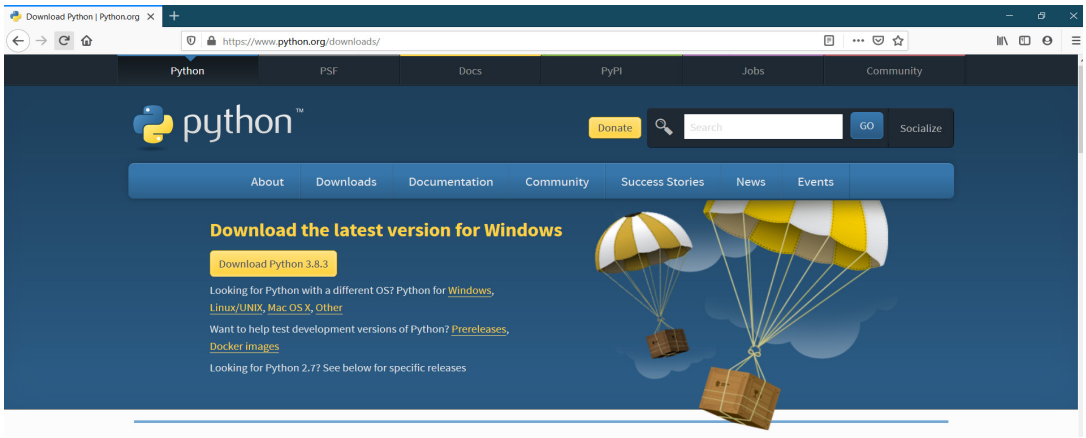
Python dilini bilgisayara kurarak program yazılabilir. Daha önce de belirtildiği gibi Python dilini bilgisayara kurmak oldukça hızlı ve kolay bir işlemdir.

Bunun için Python resmî web sitesi olan python.org adresi ziyaret edilmelidir. Siteye girildikten sonra;

• Üst tarafta bulunan "Downloads" menüsü altında yer alan "Python x.y.z" butonuna veya "python.org/downloads/" adresinde yer alan "Download Python x.y.z" butonuna tıklamak yeterlidir.

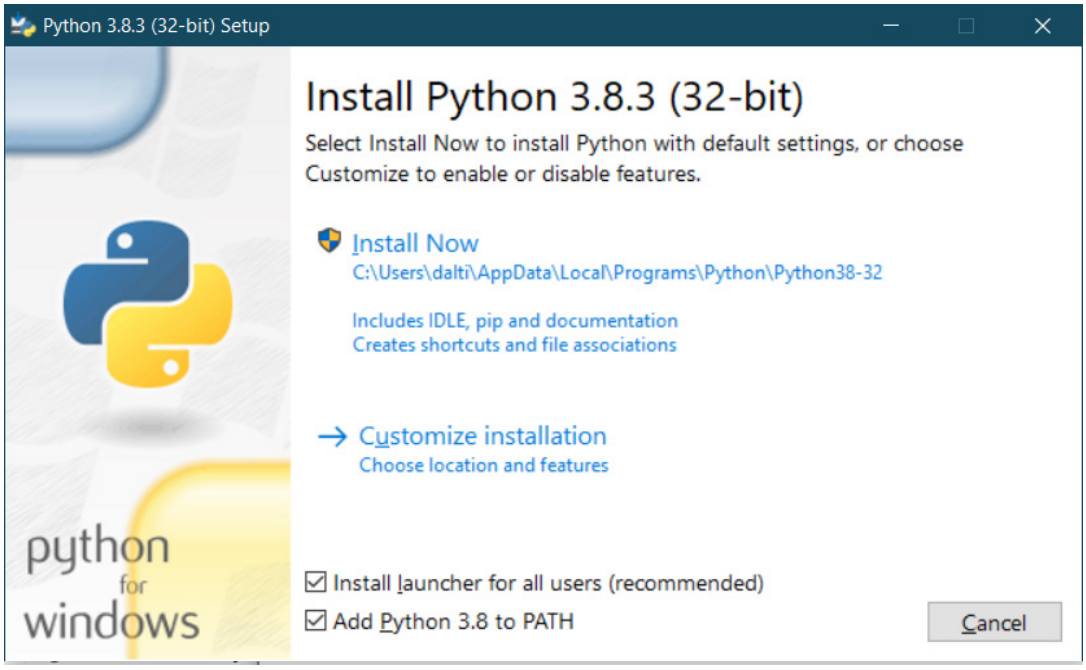


Görsel 3.5: Python kurulumu



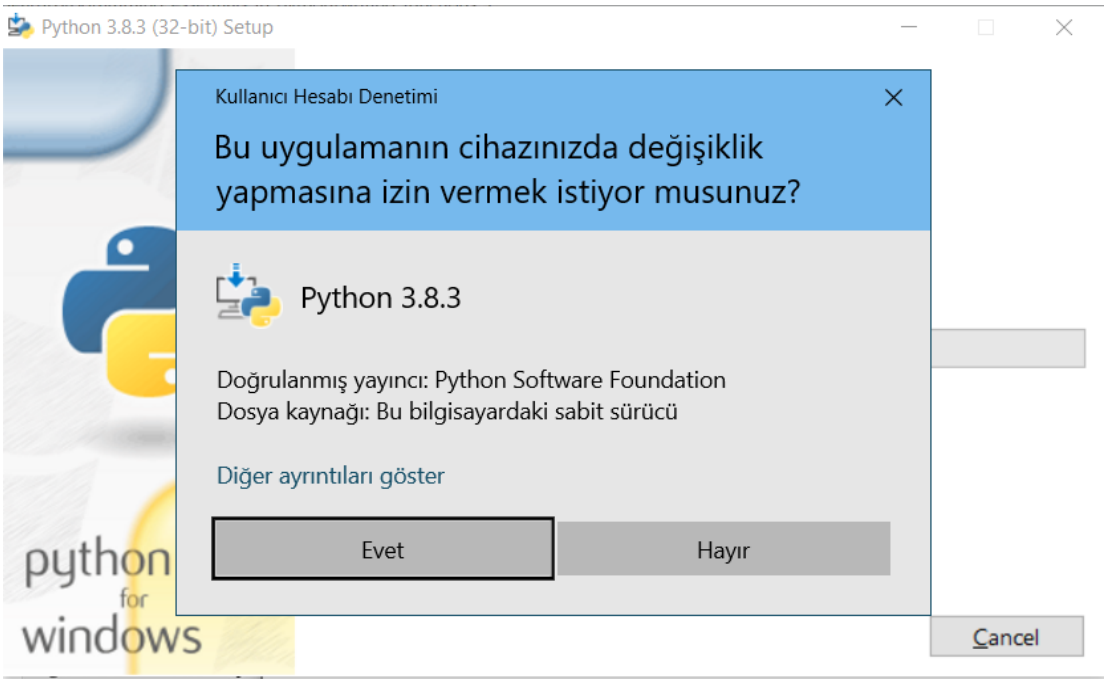
Görsel 3.6: Python kurulumu

Butona tıkladığınızda "Python-x.y.z.exe" dosyası bilgisayarınıza indirilecektir. İndirme işleminin ardından .exe uzantılı dosyayı çalıştırdığınızda kurulum aşaması başlayacaktır. Karşınıza çıkan ilk ekranda "Add Python x.y to PATH" seçeneğini işaretlemeniz tavsiye edilir. Ardından "Install Now" butonuna tıklanıp bir sonraki ekrana geçiş sağlanır.



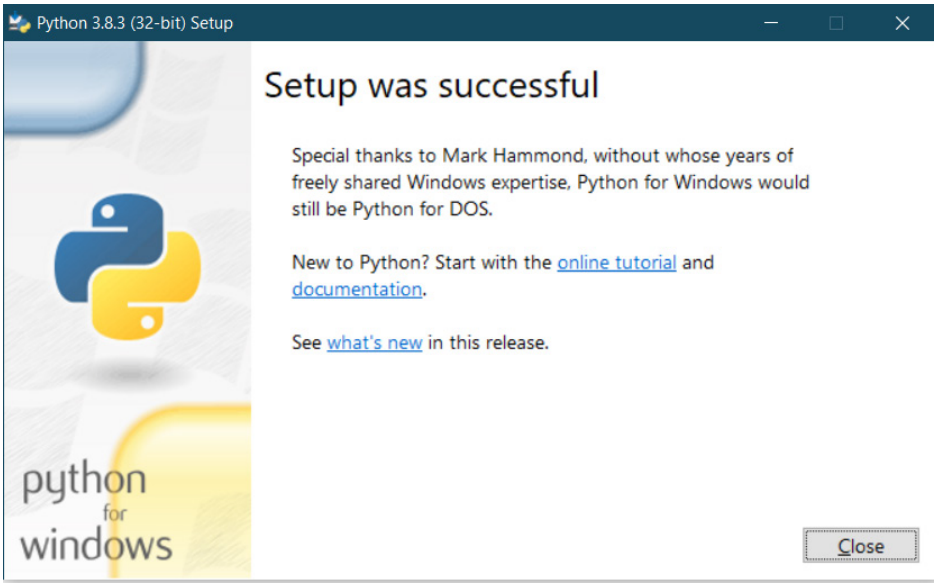
Görsel 3.7: Python kurulumu

İşletim sistemi "Kullanıcı Hesabı Denetimi" onayı isterse "Evet" seçeneğine tıklanarak kuruluma devam edilir.



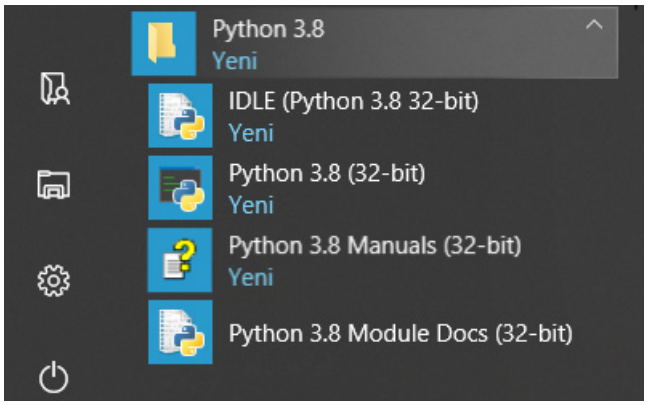
Görsel 3.8: Python kurulumu

Kısa bir süre sonra "Setup was successful" yazısı görüldüğünde kurulum işlemi tamamlanmış olacaktır. Artık Python dili bilgisayarda kullanılabilir hâle gelmiş demektir. "Close" butonu tıklanarak pencere kapatılır.



Görsel 3.9: Python kurulumu

İlk komutu yazmak için kurulumun ardından "Başlat" menüsü içinde yer alan "Python 3.8" menüsündeki "Python 3.8 (32-bit)" programını çalıştırmak gerekir. Sürüm bilgisinin (3.8) kurulum yapılan sürüme göre değişiklik gösterebileceği unutulmamalıdır.

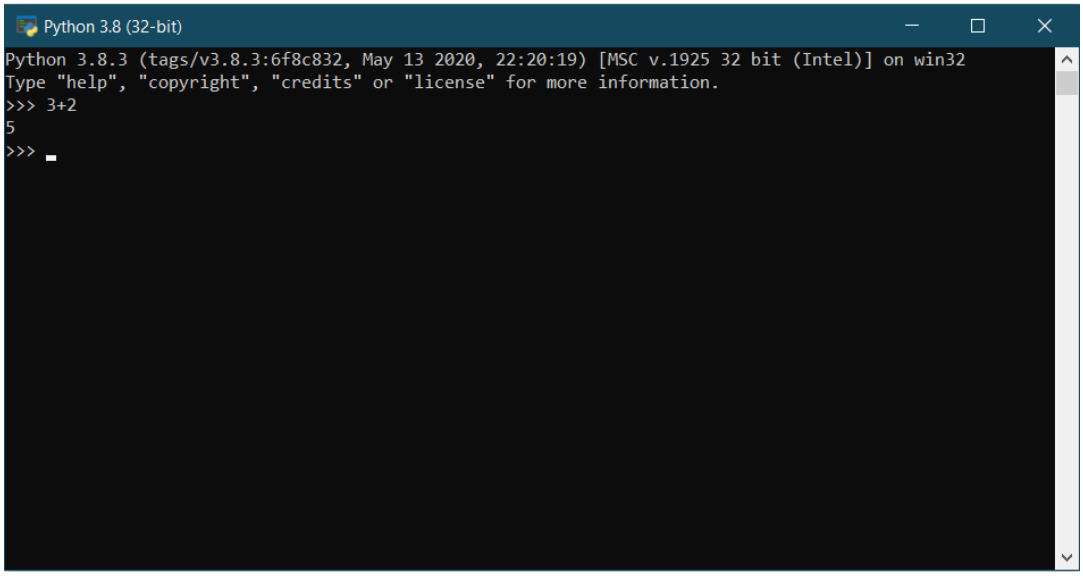


Görsel 3.10: Python programlama dilini başlatma

Ardından ">>>" karakterlerinin yanında yanıp sönen imlecin olduğu yere ilk komut yazılabilir. Bu ekran, "konsol ekranı" olarak adlandırılır.

```
3 + 2
```

Kod Örneği 3.1



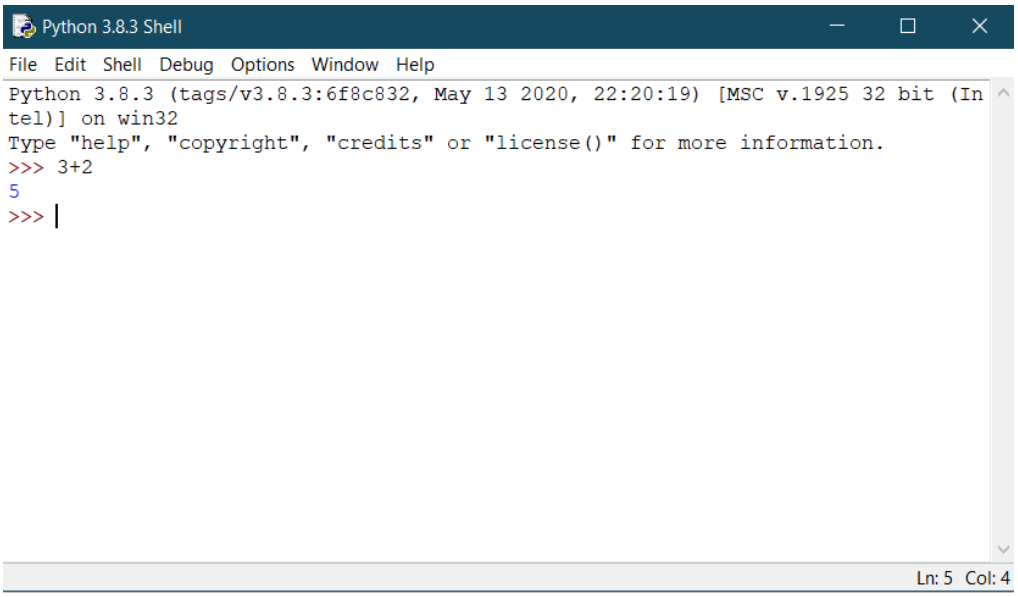
Görsel 3.11: Python konsol ekranı

Python dili, bu ekranda komutları çalıştırabilir. Ancak kısıtlı imkânların bulunduğu bir program olduğu bilinmelidir. Bu nedenle komutların daha rahat yazılabileceği bir arayüz de kurulum ile gelmektedir.

Bu sefer, "Python 3.8" menüsündeki "IDLE (Python 3.8 32-bit)" programı çalıştırılır (IDLE: Integrated Development and Learning Environment / Bütünleşik Geliştirme ve Öğrenme Ortamı).

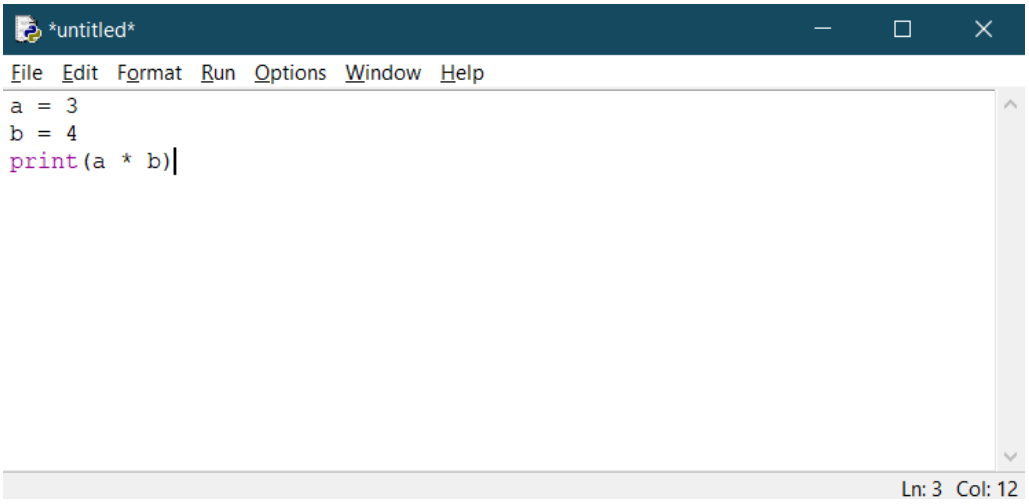
Karşınıza aşağıdaki pencere gelecektir. Bu ekrana "Shell" adı verilir.

Aynı komut buraya yazıldığında aynı sonucun hesaplandığı görülür.



Görsel 3.12: Python Shell ekranı

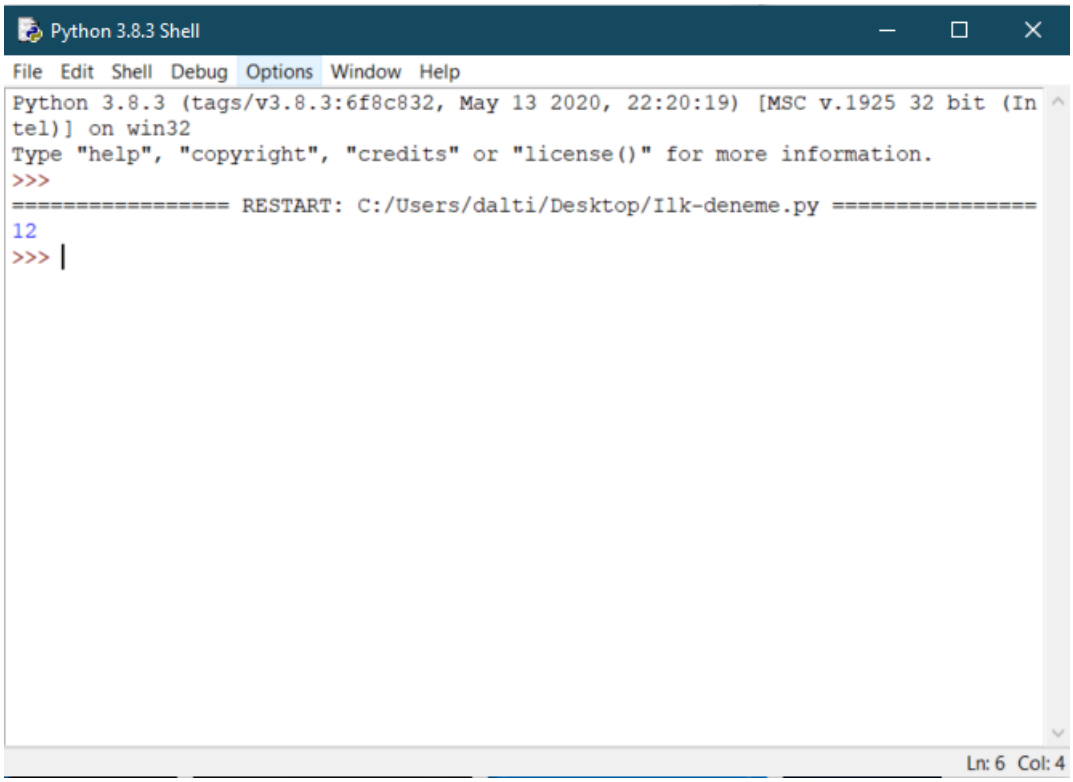
Tek bir satır komut çalıştırmak değil de birden fazla kod satırı çalıştırılmak istenirse "File -> New File" menü seçeneğinden yeni bir dosya oluşturulur ve kodlar bu dosyaya yazılıp çalıştırılır.



Görsel 3.13: Python kod dosyası

Başlık çubuğunda "untitled" yazdığı görülüyorsa henüz dosyanın kaydedilmediği anlaşılır. Hemen "File -> Save" menü seçeneğinden dosya istenen bir klasör altına kaydedilir. Dosya kaydedildikten sonra başlık çubuğundaki metnin değiştiği görülür (Dikkat edilirse dosya uzantısı ".py" olarak kaydedilmiştir.).

"Run -> Run Module" menü seçeneği ile yazılan program çalıştırılır. Programın çıktısı ayrı bir pencerede sunulacaktır.



Görsel 3.14: Python Shell ekranı

Görüldüğü üzere sadece birkaç dakika içinde sıfırdan Python dili bilgisayara kurularak ilk denemeler gerçekleştirilebilir. Bundan sonra diğer Python dili komutlarını ve özelliklerini öğrenecek ve kendi programlarınızı yazabilir hâle geleceksiniz.

3.6. Python için Gerekli Araçlar

3.6.1. Editör Kurulumu

Python'da kod yazıp çalıştırmak için önceki bölümde bahsedilen kurulumlar yeterlidir. Ancak kod satır sayısı arttıkça birden fazla dosya ile çalışmaya başlandığında şu ana kadar kullanılan IDLE (Integrated Development and Learning Environment / Bütünleşik Geliştirme ve Öğrenme Ortamı)'nin yetersiz kalacağı da bilinmelidir.

Editör, çok fazla özelliği olmayan bir kelime işlemci olarak düşünebilir (Notepad gibi). Program yazımlarını kolaylaştırması ve hızlandırması için daha gelişmiş editörler mevcuttur. Bu yazılımlara IDE (Integrated Development Environment / Bütünleşik Geliştirme Ortamı) adı verilmektedir. Şu an kullanıma sunulmuş pek çok IDE mevcuttur.

IDE'ler size kod yazmak için sadece editör imkânı sağlamaz. Aynı zamanda;

- Dosyalarınız arasında kolayca gezinme,
- Gelişmiş dosya işlemleri (bul, değiştir, satıra git vb.),
- Editör ortamını özelleştirme (renk, font, yerleşim vb.),
- Otomatik kod tamamlama,
- Kod renklendirme,
- Hazır geliştirme araçları,
- Gelişmiş hata ayıklama aracı,
- Kütüphanelere erişim kolaylığı vb. daha pek çok kullanım kolaylığı da sağlar.

Dolayısıyla Python ile gelen IDLE yerine, aşağıda bazıları sıralanan IDE'lerden kullanılması tavsiye edilir:

- Pycharm
- Spyder
- Eclipse + Pydev

Sublime Text
Visual Studio Code
Vim
Atom
Jupyter vb.

IDE tercihinizi, internetten kısa bir araştırma ile belirleyebilirsiniz.

3.6.2. Kütüphane Kullanımı

Kütüphane kavramını, önceden yazılmış ve çok sık kullanılan kod parçacıklarını programın içine dâhil ederek o kod parçacıklarını kendiniz yazmışsınız gibi kullanabildiğiniz bir yapı olarak düşünebilirsiniz. Bazı kaynaklarda "kütüphane" kavramı "paket" veya "modül" olarak da geçmektedir.

Python ile gelen pek çok hazır kütüphane "Standart kütüphaneler" olarak isimlendirilir. Programlar içinde doğrudan kullanılabileceği gibi internetten veya çeşitli araçlarla rahatlıkla bulunabilecek, daha özelleşmiş fonksiyonları barındıran diğer kütüphaneler de kullanılabilir.

Aşağıdaki örnekte Python ile gelen örnek bir kütüphane programa dâhil edilip kullanıldığında 1-100 arası rastgele bir sayı oluşturulmaktadır.

```
import random
x = random.randint(1, 100)
print(x)
```

İlk satırdaki import random komutu ile programa random kütüphanesi eklenmiş olunur. Ardından random.randint(1,100) komutu ile 1-100 arası rastgele bir tam sayı oluşturulması sağlanır ve ekrana yazdırılır. "Random" kütüphanesi Python kurulumu ile geldiği için ekstra bir kurulum gerektirmez.

Örnek 1: 2 üzeri 3'ü hesaplayan bir program kodu yazınız.

```
import math
x = math.pow(2, 3)
print(x)
```

http://kitap.eba.gov.tr/KodSor.php?KOD=22325

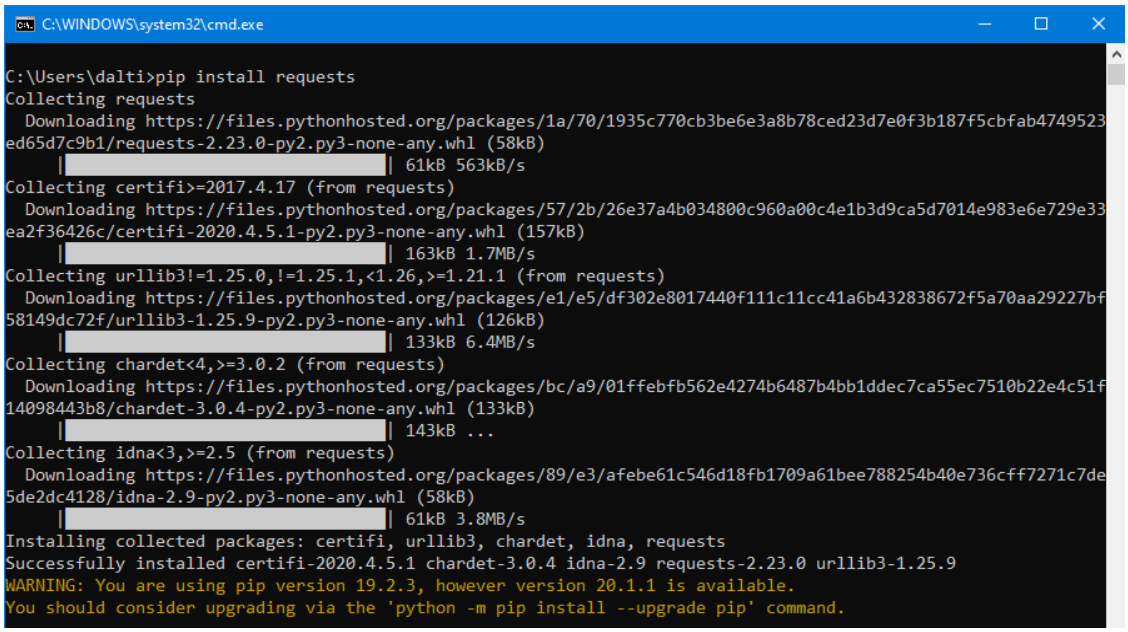
İlk satırda bu sefer math kütüphanesinin eklendiği görülmektedir. "math" kütüphanesi de Python kurulumu ile bilgisayara otomatik yüklenen kütüphanelerden biridir. math.pow(2, 3) komutu ile 2 üzeri 3'ün hesaplanması gerçekleştirilmiştir.

Python ile gelen standart kütüphanelerin tam listesi https://docs.python.org/3/library/index.html adresinden incelenebilir.

Öte yandan; Python kurulumu ile gelmeyen bir kütüphaneyi programda kullanabilmek için öncelikle ilgili kütüphane bilgisayara yüklenmelidir. Python kurulumu ile gelen "PIP" (Package Installer for Python / Python için paket yükleyicisi) programı kullanılarak bilgisayara Python kütüphaneleri yüklenebilir.

Örnek bir paket yüklemek için konsol ekranı kullanılarak aşağıdaki komut yazılır.

```
pip install requests
```



Görsel 3.15: Kütüphane kurulumu

"Successfully installed …" yazan satır, istenilen kütüphanenin doğru bir şekilde bilgisayara yüklendiğini ifade etmektedir. Artık "requests" kütüphanesi programda kullanılabilir.

```
import requests


icerik = requests.get('https://www.w3schools.com/xml/note.xml')
print(icerik.text)
```

Örnek program, belirtilen adresteki içeriği okuyup ekrana yazdırmaktadır.

Kullanışlı bazı Python kütüphane listelerine https://wiki.python.org/moin/UsefulModules adresinden erişilebilir.

Kütüphaneler, 6. öğrenme biriminde daha geniş bir şekilde işlenecektir.

ÖLÇME VE DEĞERLENDİRME 3

1. Python programlama dili ile "işletim sistemi" yazılabilir mi? Araştırınız.
2. Bir yazılımın niçin en son versiyonunun kullanılması gerektiğini açıklayınız.
3. Python programlama dilini kullanabileceğimiz IDE'lerini (Integrated Development Environment / Bütünleşik Geliştirme Ortamı) inceleyiniz. Hangilerinin tercih edilebileceğini açıklayınız.
4. Shell ekranında aşağıdaki komutları çalıştırınız ve bu komutların ne iş yaptıklarını karşılarına yazınız.
   i. 10 + 20 : .....
   ii. 4 * 30 : .....
   iii. 2 ** 1000 : .....
   iv. print("merhaba") : .....
   v. 36 / 4 * (3 + 2) * 4 + 2 : .....
5. [ ] Python yorumlanan bir dildir. (Evet/Hayır)

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.

ÖĞRENME BİRİMİ 4

VERİ YAPILARI

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Değişken ve sabit kavramlarını açıklayabilecek,
- Değişken tanımlayarak programlarınızda kullanabilecek,
- Operatörleri ve veri tiplerini anlayabilecek ve kullanabileceksiniz.

Anahtar Kelimeler:

Değişken, sabit, operatör, veri tipi.

Hazırlık Çalışmaları

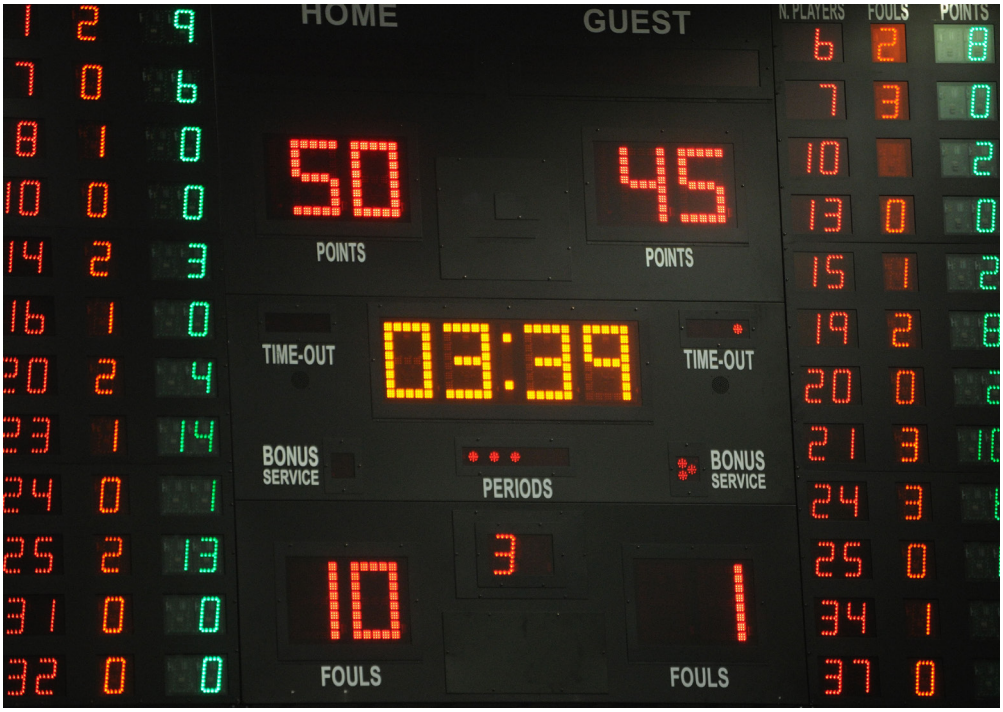
1. Değişken kavramını araştırıp günlük hayattan örnekler vererek arkadaşlarınızla tartışınız.
2. Python programlama dilinde kullanılan operatörleri araştırarak matematik dersinde kullanılan operatörlerle karşılaştırınız.

4. VERİ YAPILARI

4.1. Değişken ve Sabit Kavramları

Programlama dillerinde ihtiyaç olduğu an ulaşılacak veri tutuculara değişken adı verilir. Değişkenler kutular olarak düşünülebilir. Kutular açılarak içinde ne olduğuna bakılabileceği gibi kutuların içine yeni bir şeyler de koyulabilir. Adı üzerinde değişkenler, program içinde değeri değişebilen tutuculardır.

Değişken kavramını anlayabilmek için günlük hayattan örnekler verilebilir. 12 Dev Adam'ın bir maçını düşünün. Maçın o anki skoru 47-45 12 Dev Adam lehine olsun. Burada 47 sayısı değişkenin o anki değeridir. Cedi Osman'ın attığı üç sayılık isabetli atıştan sonra değişkenin yeni değeri artık 50 sayısıdır.



Görsel 4.1: Değişken mantığı

Değişkenler sadece tam sayıları değil; ondalıklı sayıları, metinleri, doğru ya da yanlış gibi ifadeleri de hafızada tutabilir.

Her değişkenin bir adı ve değeri vardır. Tekrar kutu örneğini hatırlayınız. Kutunun bir değişken olduğu varsayıldığında burada görünen "paket" ifadesi değişken adı, kutu açıldığında karşınıza çıkan "kitap" ise değişkenin değeridir.

Görsel 4.2: Değişken adı ve değeri

Sabit kavramı ise uygulama çalıştığı sürece değeri değişmeyen veriler olarak ifade edilebilir. Örneğin bir inç 2,54 cm'dir. Bu gibi değeri değişmemesi gereken veriler sabit olarak tanımlanabilir.

Sıra Sizde: Sabit olarak kullanılabilecek verilere birkaç örnek düşününüz.

http://kitap.eba.gov.tr/KodSor.php?KOD=22327

4.1.1. Değişken Tanımlama

Değişken tanımlamak için her programlama dilinde önceden belirlenmiş bazı kurallar bulunmaktadır. Python programlama dilinde değişken tanımlarken önce değişken adı yazılır. Değişken adı yazıldıktan sonra = (eşittir) işareti konulur ve değişkenin değeri yazılır. Ancak burada değişken isimlendirme kurallarına dikkat edilmelidir.

Değişken isimlendirirken hata mesajı ile karşılaşmamak için uyulması gereken kurallar şunlardır:

a) Değişken isimleri case sensitive yani büyük küçük harf duyarlıdır. Örneğin; değişken isminin adres ya da Adres olması bu değişkenlerin farklı iki değişken olduğunu gösterir.

b) Değişken isimlerinin anlaşılır olması işinizi kolaylaştırır. Örneğin; kullanıcıdan elektronik posta bilgisi alınacağı zaman bunu e-posta gibi anlaşılır bir değişken ismi ile ifade edebilirsiniz.

c) Değişken isimlendirilirken farklı standartlar kullanılmaktadır. Python'da genel kabul gören standart Snake Case standardıdır. Bu kitapta Lower Snake Case kullanılmıştır. Snake Case standardında değişken isimleri iki farklı kelimeden oluşuyorsa alt tire (_) ile birleştirilir. Lower Snake Case ise tüm harflerin küçük harf olacağı anlamına gelir. Örneğin: ev_adresi, kimlik_numarasi vs.

ç) Değişken isimlendirilirken hem harfler hem de sayılar kullanılabilir. Ancak sayılar başa gelmez. Örneğin sayi1 doğru bir isimlendirmeyken 1sayi doğru bir isimlendirme değildir.

d) Değişken isimlendirilirken alt tire (_) kullanılabilir. Ancak boşluk ve diğer özel karakterler (?,%,!, ., + vb.) kullanılmaz. Örneğin ev adresi ya da kimlik%no gibi değişken isimleri kurallara aykırı olduğundan hataya neden olacaktır.

e) Değişken isimlendirilirken özel kullanım için ayrılmış olan if, for, true vb. ifadeler hata vermemesine rağmen özellikle kodların daha anlaşılır olması amacıyla kullanılmamalıdır.

f) Bazı programlama dillerinde Türkçe karakterlerin (ç,ğ,ı,ö,ş,ü) kullanımı kabul edilirken bazılarında kabul edilmez. Python'da Türkçe karakterler kullanılması hataya neden olmaz. Ancak farklı programlama dillerinde problem yaşanmaması için değişken tanımlarken Türkçe karakter kullanılmaması önerilmektedir.

Bu kurallar çerçevesinde aşağıda doğru tanımlanmış bazı değişken örnekleri görülmektedir:

```
yasadigi_sehir="Ankara"

sinav_notu=72

faiz_orani=5.7
```

Sıra Sizde: Aşağıdaki değişken tanımlamalarını doğru ya da yanlış olarak değerlendiriniz. Yanlış olanların neden yanlış olduğunu açıklama bölümüne yazınız.

Değişken İsmi	Doğru	Yanlış	Açıklama
yükseklik			
uzun kenar			
3not			
ortalama			
x			

Örnek : okul_no isimli ve değeri 1923 olan bir değişken tanımlayarak bu değeri ekranda yazdırınız.

```
okul_no=1923
print(okul_no)
```

Bu örnekte ilk satırda bir değişken tanımlandı. İkinci satırda ise print fonksiyonu ile okul_no değişkeni ekrana yazdırıldı.

Örnek : Kısa kenarı 3 cm, uzun kenarı 5 cm olan dikdörtgenin alanını hesaplayınız.

```
kisa_kenar=3
uzun_kenar=5
alan=kisa_kenar *uzun_kenar
print(alan)
```

Bu örnekte kısa ve uzun kenar değişkenleri tanımlandıktan sonra alan isminde yeni bir değişken tanımlandı. Dikdörtgenin alanı hesaplanarak (kisa_kenar *uzun_kenar) yeni tanımlanan alan değişkenine atandı. Print fonksiyonu ile alan değişkenin değeri (15) ekrana yazdırıldı.

4.2. Operatörler

Veriler üzerinde işlem yaparak yeni değerler üretilmesini sağlayan programlama dili sembollerine operatör adı verilir. Python programlama diline yeni başlayanlar için aritmetiksel, atama, karşılaştırma, mantıksal ve kimlik operatörleri öğrenmek son derece önemlidir.

4.2.1. Aritmetiksel Operatörler

Operatör	Tanımı	Örnek
+	Toplama	a+b
-	Çıkarma	a-b
*	Çarpma	a*b
/	Bölme	a/b
%	Mod alma (Bir sayının diğer sayıya bölümünden kalan)	a%b
**	Kuvvet alma (ab)	a**b
//	Tam sayı bölme (Bölme işleminde sadece tam kısım alınır.)	a//b

Örnek : (4+3)**2 işleminin sonucunu bulunuz.

Kitabın ilk bölümünde işlem önceliği konusunu öğrenmiştiniz. İşlem önceliğine göre önce parantez içindeki işlem yapılır. Parantez içindeki işlemin sonucu 7 olduğundan bu örnek artık 7**2 şekline dönüşmüştür. ** operatörü üs almak için kullanıldığından 7 sayısının 2. kuvveti (yani karesi) alınmalıdır. Bu kod çalıştırıldığında 49 çıktısı ile karşılaşılır.

Önemli Not: Matematikte çarpma işlemi genel olarak çarpı (x) ya da nokta (.) ile ifade edilir. Ancak çoğu programlama dilinde çarpma işleminin * ile ifade edildiği unutulmamalıdır.

Sıra Sizde: Aşağıdaki örneklerin sonuçlarını da siz yazınız.

2**5	
5//2	
4+3	
11%3	
5/2	
2*5	
3-5	
(4-1)**2	
(7//3)/2	

4.2.2. Atama Operatörleri

Operatör	Örnek	Açıklama
=	a=2	a değişkenine 2 değeri atanmıştır.
+=	a+=2	a değişkenine 2 değerini ekleyerek yine a değişkenine atanmıştır. Başka bir ifadeyle a=a+2 anlamına gelmektedir.
-=	a-=2	a değişkeninden 2 değeri çıkarılarak yine a değişkenine atanmıştır. Başka bir ifadeyle a=a-2 anlamına gelmektedir.
*=	a*=2	a değişkeni 2 ile çarpılarak yine a değişkenine atanmıştır. Başka bir ifadeyle a=a*2 anlamına gelmektedir.
/=	a/=2	a değişkeni 2 değerine bölünerek yine a değişkenine atanmıştır. Başka bir ifadeyle a=a/2 anlamına gelmektedir.
%=	a%=2	a değişkenin 2 değeri ile modu alınarak yine a değişkenine atanmıştır. Başka bir ifadeyle a=a%2 anlamına gelmektedir.
**=	a**=2	a değişkeninin ikinci kuvveti (a2) alınarak yine a değişkenine atanmıştır. Başka bir ifadeyle a=a**2 anlamına gelmektedir.
//=	a//=2	a değişkeni 2 değerine tam bölünmüş (kalan dikkate alınmadan) ve çıkan değer yine a değişkenine atanmıştır. Başka bir ifadeyle a=a//2 anlamına gelmektedir.

Örnek :

http://kitap.eba.gov.tr/KodSor.php?KOD=22330

```
a=8
a/=2
print(a)
```

a değişkeninin ilk değeri 8 olarak tanımlanmıştır. a/=2 ifadesi a=a/2 anlamına da gelen a değişkenini 2 değerine böl demektir. Print fonksiyonu ile a değişkeninin son değeri olan 4.0 (8/2) ekrana yazdırılmıştır.

Aşağıdaki kodlar çalıştırıldığında çıktılarının ne olduğunu yazınız.

a=3 a+=2 print(a)	
a=6 a*=3 print(a)	
a=5 a**=3 print(a)	
a=11 a%=3 a**=3 print(a)	

4.2.3. Karşılaştırma Operatörleri

Operatör	Tanımı	Örnek
==	Eşittir	a==b
!=	Eşit değildir	a!=b
<	Küçüktür	a<b
>	Büyüktür	a>b
<=	Küçük eşittir	a<=b
>=	Büyük eşittir	a>=b

Programlama dilinde gerçekleştirilen karşılaştırmalar doğru ise True yanlış ise False değerlerini döndürür. Konuyu anlamak için aşağıdaki örneği inceleyiniz.

```
a=6
b=4
print (a<b)
```

a değişkenine 6, b değişkenine 4 değeri verilmiştir. Print fonksiyonuyla da a<b karşılaştırmasının sonucu ekrana yazdırılacaktır. 6 sayısı 4 sayısından küçük olmadığı için bu kod çalıştırıldığında False çıktısını üretir. Eğer üçüncü satır print (a>b) ya da print (b<a) şeklinde değiştirilirse çıktı da True değerini üretecektir.

Aşağıdaki kodlar çalıştırıldığında çıktılarının True mu yoksa False mu olduğunu yazınız.

a=3 b=4 print (a==b)	
a=6 b=6 print (a==b)	
a=2 b=1 print (a!=b)	
a=6 b=11 print (a<b)	
a=9 b=7 print (a>b)	
a=5 b=5 print (a>=b)	

4.2.4. Mantıksal Operatörler

Operatör	Örnek	Açıklama
and	a<3 and b>=5	İki veya daha fazla şartın tamamının doğru olması durumunda True değerini döndürür. Buradaki örnekte a değişkeni 3'ten küçük ve b değişkeni 5'e eşit ya da 5'ten büyük olursa True değeri döndürülür.
or	a<3 or b>4	İki veya daha fazla şartın en az birinin doğru olması durumunda True değerini döndürür. Buradaki örnekte a değişkeninin 3'ten küçük olması ya da b değişkenin 4'ten büyük olması True değeri döndürmek için yeterlidir.
not	not(a<3)	Durumu tersine çevirmek (True ise False; False ise True) için kullanılır. Buradaki örnekte parantez içindeki mantıksal sınamanın sonucu tersine çevrilir. İfadenin not komutu olmadan yazıldığında true döndüreceği varsayıldığında bu haliyle false döndürecektir.

Örnek :

http://kitap.eba.gov.tr/KodSor.php?KOD=22332

```
a=5
b=3
print (a>b and b<2)
```

a değişkenine 5, b değişkenine ise 3 değeri atanmıştır. Print fonksiyonuyla (a>b and b<2) mantıksal sınaması yapılarak sonuç ekrana yazdırılacaktır. and tüm şartların doğru olması durumunda True değerini döndürür. Burada iki şart bulunmaktadır. Birinci şart olan a>b şartına göre True değeri üretilir. İkinci şart olan b<2 şartında ise 3 değeri 2 değerinden küçük olmadığı için False değeri döndürülür. İki şart birlikte değerlendirildiğinde biri doğru, diğeri yanlıştır. Bu nedenle sonuç da yanlış yani False olarak karşımıza çıkacaktır.

Aşağıdaki kodlar çalıştırıldığında çıktılarının True mu yoksa False mu olacağını yazınız.

a=5 b=3 print (a>b and b>2)	
a=6 b=6 print (a==b and a<10)	
a=2 b=4 print (a==b or a>b)	
a=2 b=4 print (a!=b or a>8)	
a=9 b=7 print (not(a>b))	
a=5 b=5 print (not(a>=b and a<1))	

4.2.5. Kimlik Operatörleri

Python programlama dilinde karşımıza çıkan operatörlerden biri de kimlik operatörleridir. Kimlik operatörleri değişken değerlerini karşılaştırmak yerine, değişken ya da nesne adreslerini karşılaştırarak sonuç üretir. Python'da her nesnenin bellek adresini ifade eden kimlik numarası (identity) vardır. Değişken ya da nesnelerin bellek adresleri yani kimlik numaraları id() fonksiyonu ile öğrenilebilir. Kimlik operatörlerini kullanmak için is ve is not ifadeleri kullanılır.

Örnek :

```
a=100
b=101
print(id(a))
print(id(b))
a+=1
print(id(a))
print (id(b))
```

Ekran Çıktısı:

```
140729385594352
140729385594384
140729385594384
140729385594384
```

Dikkat edilecek olursa a ve b değişkenlerinin bellek adresleri başta farklıyken a değişkeninin değeri 1 artırıldığında (a+=1) bellek adresleri aynı olmuştur. Bu durumda yukarıdaki değişkenler için is operatörü kullanıldığında aşağıdaki sonuçlar ortaya çıkacaktır.

print (a is b) => a ve b değişkenlerinin bellek adresleri aynı olduğundan True döndürür. Bu işlemi yukarıdaki kodun en alt satırına print (a is b) satırını ekleyerek deneyebilirsiniz.

Sıra Sizde: Aşağıdaki kod nasıl bir çıktı üretir?

```
a=10
b=11
c=12
print (a is c)
print (a is not b)
a+=1
print (a is b)
a+=1
print(a is c)
```

4.3. Veri Tipleri

Python'da genel olarak string (metinsel), numbers (sayısal), list (liste), tuple (demet), dictionary (sözlük) ve set (küme) veri tipleri bulunmaktadır.

4.3.1. String (Metinsel) Veri Tipi

Tek ya da çift tırnak içlerine yazılan karakter dizileridir. Burada karakter harf (t,c), rakam (1,9,2,3) ya da özel semboller (&,/) olabilir. String veri tipleri tek ya da çift tırnak içinde yazılır.

Örneğin aşağıdaki iki ifade birbirinin aynısıdır.

```
print ("Bütün ümidim gençliktedir.")
print ('Bütün ümidim gençliktedir.')
```

Her iki kod satırı çalıştırıldığında aynı çıktı üretilir. Bu kitapta çift tırnak kullanılacaktır.

String bir değişken tanımlama işlemi aşağıdaki gibi yapılır:

yasadiginiz_sehir="Ankara"

Sıra Sizde: Değişken adı para birimi, değeri Türk lirası olan bir değişken tanımlayınız ve bu değişkeni ekrana yazdırınız.

Python'da type() fonksiyonu kullanılarak veri tipi öğrenilebilir.

Örnek :

okul_turu="Meslek Lisesi"

print(type(okul_turu))

Bu kod parçası çalıştırıldığında <class 'str'> çıktısı üretilir. Bu çıktının anlamı kullanılan veri tipinin string olduğudur.

String ifadeleri birbirlerine bağlayabilirsiniz.

ifade1="Merhaba"

ifade2="Dünya"

ifade3=ifade1+ifade2

print(ifade3)

Bu örnekte bir programlama ritüeli olan Merhaba Dünya satırı birleştirilerek yazıldı. Burada matematikte de bilinen toplama işlemi kullanıldı. Başka bir deyişle daha önce öğrenilen aritmetiksel operatörlerden biri olan toplama işlemi string ifadelere uygulandı. Bu kod çalıştırıldığında "MerhabaDünya" çıktısı üretilir. Bu çıktıyı biraz daha düzenlemek için iki kelime arasına boşluk bırakılabilir. Bunun için ilk yöntem ilk değişkeni ifade1="Merhaba " olarak değiştirmektir. İkinci bir yöntem olarak da bosluk="_" değeri bir karakterlik boşluk olan bir değişken tanımlanarak ifade3=ifade1+boşluk+ifade2 yazılabilir. Her iki yöntem de aynı sonucu üreteceğinden sizin için kolay olanı tercih edebilirsiniz.

Sıra Sizde: Değerleri sırasıyla Millî, Eğitim, Bakanlığı olan üç değişken tanımlayınız. Bu üç değişkeni birleştirerek ekrana yazdırınız. Kelimeler arasında birer tane boşluk olması için gerekli işlemleri yapınız.

String ifadeleri tekrarlamak için toplama işlemi yerine çarpma işlemi kullanılabilir. Daha önce belirtildiği üzere programlama dilinde çarpma işlemi * operatörü ile ifade edilmektedir.

Örnek :

TR ifadesini 5 kez tekrarlayan kodu yazınız.

print("TR"*5)

Burada TR ifadesini ekrana 5 kez yazdırmak için tekrar ettirme işlemi çarpma operatörü olan * ile yapılmıştır.

Örnek : : "Python" ifadesini 10 kez yazdırınız.

ifade="Python"

ifade2=ifade *10

print(ifade2)

Bu örnekte tekrar ettirme işlemi değişken tanımlanarak yapıldı. ifade2 değişkeni ifade değişkeninin 10 ile çarpılması şeklinde tanımlandı ve ekrana yazdırıldı.

Sıra Sizde: Değeri "Merhaba" olan bir değişken tanımlayınız ve 3 kez yan yana yazdırınız.

4.3.2. Numbers (Sayısal) Veri Tipleri

Sayısal verileri tutan veri tiplerine verilen addır. Python'da sayısal veri tipleri genel olarak int, float ve complex veri tipleridir. Bu kitapta int ve float veri tiplerinden bahsedilecektir. Int veri tipi tam sayı değerleri tutarken; float veri tipi ondalıklı değerleri tutar. Bu noktada tüm tam sayıların da ondalıklı olarak ifade edilebileceğini unutmayınız. Örneğin 3 tam sayısı (normalde int) 3.00 şeklinde ifade edildiğinde float olarak da tanımlanabilir.

Örnek : sayi isminde değeri 1919 olan bir değişken tanımlayarak ekrana yazdırınız.

```
sayi=1919
print(sayi)
```

Örnek : pi_degeri isminde değeri 3.14 olan bir değişken tanımlayarak ekrana yazdırınız.

```
pi_degeri=3.14
print(pi_degeri)
```

Her iki örnekte de int ya da float şeklinde bir tanımlama yapılmadı. Birçok programlama dilinin aksine Python veri tiplerini belirleme yeteneğine sahiptir. Daha önce değinilen type() fonksiyonu kullanılarak veri tipleri kontrol edilebilir.

Sıra Sizde: Her iki örnekte de üçüncü bir satır oluşturarak veri tipini ekrana yazdırınız.

İpucu: BHString konusuna dönerek type() kullanımını hatırlayınız.

Örnek : Bir öğrencinin matematik dersinden aldığı notlar sırasıyla 64, 86 ve 70'tir. Bu öğrencinin not ortalamasını hesaplayınız.

http://kitap.eba.gov.tr/KodSor.php?KOD=22334

```
not1=64
not2=86
not3=70
ortalama=(not1+not2+not3)/3
print(ortalama)
```

Çıktı: 73.33333333333333

Bu örnekte öğrencinin aldığı notlar üç farklı değişkene aktarıldı. Ortalama isimli bir değişken tanımlanarak formül yazıldı. Daha sonra print fonksiyonu ile sonuç ekrana yazdırıldı. Bu örnekte aranızdan 173.33333333333334 sonucunu bulanların da olması muhtemeldir.

Kitabın ilk bölümünde işlenen işlem önceliği konusunu hatırlamanız gerekir. Eğer üç notun toplamını parantez içine almadan 3'e bölerseniz aslında sadece üçüncü sınavı bölmüş olursunuz ve sonuç yanlış çıkacaktır. Bu nedenle işlem önceliğine dikkat etmelisiniz.

Bu örnekte dikkat edilmesi gereken bir başka husus da girilen notların int veri tipinde olmasına rağmen sonucun float olarak üretilmesidir.

Önemli Not: Python'da kullanılan bir diğer veri tipi de bool veri tipidir. Kod yazarken bazı ifadelerin doğru ya da yanlış olarak değerlendirilmesi istenebilir. Bu durumlarda yalnızca True (doğru) ve False (yanlış) değerlerini döndüren bool veri tipi kullanılır.

Örnek :

```
print(1 > 2)
print(2 == 2)
print(2 < -5)
```

Ekran Çıktısı:

```
False
True
False
```

Bu örnekte üç farklı print satırında bazı ifadeler doğru ya da yanlış olarak değerlendirilerek True ya da False sonuçlarını üretmiştir.

Bool veri tipine ismi verilen George Boole'nin bilişim alanına katkılarını araştırınız.

input() fonksiyonu ile kullanıcıdan veri alma

Programlamada bazı değerlerin kullanıcılar tarafından girilmesi gerekebilir. Kullanıcıdan değer almak için input() fonksiyonu kullanılır.

Örnek : Kullanıcıya yaşını sorunuz ve girilen yaşı ekrana yazdırınız.

```
yas=int(input("Yaşınızı girin: "))
print(yas)
```

Çıktı:

```
Yaşınızı girin: 16
16
```

Bu örnekte input fonksiyonu ile kullanıcıya yaşı soruldu. Yaş sorusuna verilen cevap tam sayı olacağı için tam sayıya dönüştürülerek (casting) yaş değişkenine atanmıştır. İkinci satırda ise yas değişkeni ekrana yazdırıldı. Kod parçası çalıştırıldığında "Yaşınızı girin: " uyarısı ile karşılaşılır ve buraya kullanıcının bir değer girmesi beklenir. Alt satırda ise girilen değer ekrana yazdırılmıştır.

Bu örnek aşağıdaki şekilde değiştirilirse;

```
yas=int(input("Yaşınızı girin: "))
print("Yaşınız ",yas)
```

Çıktı:

```
Yaşınızı girin: 16
Yaşınız 16
```

Print ile başlayan satırda yas değişkeninin hemen önünde string bir ifade olduğu görülmektedir. String olduğu çift tırnak içinde yazıldığından anlaşılabilir. Bu ifade ile yas değişkeni arasına da iki değeri birleştirmek için virgül (,) eklendi. Bu şekilde çıktıları daha anlaşılır hâle getirebilirsiniz. Son olarak Yaşınız ifadesi ile 16 arasına bir karakter boşluk bırakıldığına dikkat ediniz.

Sıra Sizde:

1. Daha önce yapılan girilen 3 notun ortalaması örneğini hatırlayınız. Örnekte notlar sizin tarafınızdan verilmişti. Şimdi bu notların kullanıcılar tarafından girilmesini sağlayarak kullanıcının girdiği 3 sınav notuna göre ortalamayı hesaplayan ve ekrana yazdıran kodu yazınız.
2. Kullanıcının girdiği kısa ve uzun kenar değerlerine göre dikdörtgenin alanını ve çevresini hesaplayınız. Daha sonra Dikdörtgenin Alanı: ….. Çevresi:……. şeklinde bir çıktı üretiniz. Burada noktalar kullanıcının gireceği değerlere göre değişecektir.
3. Girilen sayının karesini ekrana yazdırınız. Ekran çıktısı aşağıdaki gibi olsun.

   ….. sayısının karesi ……'dır.
4. Kullanıcıya adını ve doğum tarihi sorunuz. Girilen doğum tarihine göre yaşını hesaplayınız. Aşağıdaki gibi bir ekran çıktısı üretiniz.

   Merhaba …(ad)……., yaşınız ……'dır.
5. Kullanıcıya adını ve bu yıl kaç kitap okuduğunu sorunuz. Aşağıdaki gibi ekran çıktısı üretiniz.

   …(ad)….., bu yıl …… kitap okudu.

Veri Tipi Dönüşümleri

Python'da bir değişkenin ya da değerin tipini başka bir veri tipine dönüştürmeniz gerekebilir. Örneğin float olarak tanımlanan bir değişkeni (örneğin 3.14) programın herhangi bir yerinde tam sayı olarak (3) kullanmanız gerekebilir. Bu durumda float olan bu değeri int'e çevirmeniz gerekmektedir. Şimdi örneklerle veri tipi dönüşümlerini inceleyiniz.

Örnek : int(3.14) ifadesi ile float olan bir değer int türüne dönüştürülür. int tam sayı olduğu için tip dönüşümü sonrası yeni değer 3 olacaktır.

Sıra Sizde:

Kod	Tip Dönüşümü Sonrası Değer
int(2.54)	
float(6)	
str(1920)	
int("500")	

Aşağıdaki kodu yazınız ve çalıştırınız.

```
sayi1=(input("Birinci sayı: "))
sayi2=(input("İkinci sayı: "))
toplam=sayi1+sayi2
print(toplam)
```

Ekran Çıktısı:

```
Birinci sayı: 10
İkinci sayı: 20
1020
```

Görüldüğü üzere kullanıcıdan iki sayı girilmesi istendi. Kullanıcının 10 ve 20 sayılarını girdiği varsayılsın. Toplama (+) işlemi sonucu 10 ve 20 değerlerinin toplanmayıp birleştirildiği görülmektedir. Eğer type() fonksiyonu kullanılarak sayi1 ve sayi2 değişkenlerinin veri tiplerine bakılırsa her ikisinin de string olduğu görülür. Bu nedenle sayi1 ve sayi2 değişkenleri int tipine dönüştürülmelidir.

Aşağıda tür dönüşümü iki farklı yolla yapılmıştır. Her ikisi de çalıştırıldığında 30 değeri ekran çıktısı olarak görülecektir.

Birinci yol	İkinci yol
sayi1=int(input("Birinci sayı:"))	sayi1=(input("Birinci sayı:"))
sayi2=int(input("İkinci sayı:"))	sayi2=(input("İkinci sayı:"))
toplam=sayi1+sayi2	toplam=int(sayi1)+int(sayi2)
print(toplam)	print(toplam)

Yorum satırları: Programlamada bazen kod satırına açıklama yapmak ya da yorum yazmak gerekebilir. Python'da yorum satırları için # işareti kullanılır.

Örnek :

```
faiz_orani=1.24 # float türünde bir değişken tanımlandı.
```

Bu örnekte # sonrasında bir açıklama yapıldı. Program çalıştırıldığında # sonrası dikkate alınmayacaktır.

4.3.3. List (Listeler)

Farklı verilerin bir dizi hâlinde tutulduğu koleksiyonlara liste adı verilir. Daha önce int, float, string gibi veri türlerini öğrenmiştiniz. Bu veri tiplerini kullanarak tek bir veriyi tutabilirsiniz. Birden fazla veriyi sıralı ve değiştirilebilen bir yapıda tutmak için listeler kullanılır. Listeler ile farklı veri tiplerini tutabilirsiniz. Python programlama dilinde listeler iki köşeli parantez ile tanımlanmaktadır.

Örnek :

```
ilk_liste=["Ankara", 312, 0.6]
print(ilk_liste)
```

Liste veri tipi için tanımlanan ilk liste incelendiğinde sırasıyla string, integer ve float tiplerinin bir arada kullanıldığı görülmektedir. Üç elemanı olan bu listeyi yazdırmak için daha önce kullanılan print() fonksiyonunu kullanmanız gerekmektedir.

Bu örneğin liste veri tipinde olduğunu doğrulamak için type fonksiyonu kullanılabilir. type(ilk_liste) kod satırı çalıştırıldığında ekran çıktısı <class 'list'> olacaktır.

Sıra Sizde:

1. Elemanları haftanın günleri olan bir liste oluşturunuz ve ekrana yazdırınız.
2. En sevdiğiniz 3 meyveyi liste hâline getirerek ekrana yazdırınız.
3. Sırasıyla pi sayısı, inç biriminin cm olarak karşılığı, mikroişlemcilerin kısaltması, kullandığınız işletim sisteminin adı ve 48 bitin byte olarak karşılığını bir liste hâline getirerek ekrana yazdırınız.

İndeks kullanımı

Liste içindeki elemanlara erişmek için ilgili elemanın indeksi kullanılır. Bazı kaynaklarda indis olarak da karşınıza çıkabilir. İlk elemanın indisi her zaman 0 (sıfır) olarak kabul edilir.

```
sehirler=["Ankara", "Bursa", "Çanakkale", "Denizli", "Eskişehir"]
```

Şehirler isimli listenin ilk elemanı olan "Ankara", indeksi sıfır olan elemandır. Aşağıdaki tabloda indeksleri ve değerleri bir arada görebilirsiniz.

İndeksi	0	1	2	3	4
Değeri	Ankara	Bursa	Çanakkale	Denizli	Eskişehir

Örnek : İndeksi 2 olan elemanı ekrana yazdırınız.

http://kitap.eba.gov.tr/KodSor.php?KOD=22336

```
sehirler=["Ankara", "Bursa", "Çanakkale", "Denizli", "Eskişehir"]
print(sehirler[2])
```

Bu örnekte indeksi 2 olan eleman Çanakkale değeridir. İlk elemanın indeksinin 0(sıfır) olduğunu unutmayınız.

Sıra Sizde:

1. Haftanın günlerinden Pazartesi ile başlayan ve Cuma ile biten bir liste oluşturunuz. Oluşturduğunuz listenin indeksi 4 olan elemanını ekrana yazdırınız.
2. Aşağıdaki kodun çıktısını yazınız (Python'da tek karakterden oluşan değerleri tek tırnak (') içinde tanımlayabilirsiniz.).

```
ders=['K','O','D','L','A','M','A']
```

Kod satırı	Çıktı
print(ders[0])	
print(ders[2])	
print(ders[5])	

İndeksler negatif olarak da yazılabilir. Örneğin -1 indeksi sondaki elemanı gösterirken -2 indeksi sondan bir öncekini gösterir.

```
sehirler=["Ankara", "Bursa", "Çanakkale", "Denizli", "Eskişehir"]
print(sehirler[-2])
```

Yukarıdaki kodun çıktısı Denizli olacaktır.

Sıra Sizde:

```
ders=['K','O','D','L','A','M','A']
```

Kod satırı	Çıktı
print(ders[-1])	
print(ders[-4])	
print(ders[-3])	

Listelerde indekslerle birlikte iki nokta (:) operatörü kullanılarak istenilen elemanlara ulaşılabilir. Bu işlem için liste[başlangıç indeksi:bitiş indeksi] yapısı kullanılır.

Örnek :

```
asal_sayilar=[2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[1:4])
```

Bu örnekte indeksi 1 olan elemandan başlayarak indeksi 4 olan elemana (4 dâhil değil) kadar ekrana yazdırır. Dolayısıyla ekran çıktısı [3, 5, 7] olacaktır.

Örnek :

```
asal_sayilar=[2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[5:])
```

Ekran Çıktısı:

```
[13, 17, 19, 23]
```

Buradaki kullanımda dikkat edilirse başlangıç olarak 5 indeksi verilip bitiş indeksi ise verilmemiştir. Bu kullanımda indeksi 5 olan elemandan başlayarak son elemana kadar yazılır.

Örnek :

```
asal_sayilar=[2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[:5])
```

Ekran Çıktısı:

```
[2, 3, 5, 7, 11]
```

Bu kullanımda da başlangıç indeksi verilmemiş bitiş indeksi olarak 5 verilmiştir. Başlangıç indeksinin verilmediği durumda indeksi 0 (sıfır) olan elemandan başlayarak yazdırılır.

Örnek :

```
asal_sayilar=[2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[0:6:2])
```

Ekran Çıktısı:

```
[2, 5, 11]
```

Bu kullanımda ise sırasıyla başlangıç indeksi, bitiş indeksi ve atlama değeri verilmiştir. Yani 0. indeksten başlayarak 6. indekse kadar ikişer artarak ekrana yazdırılır.

Sıra Sizde: Aşağıdaki kod nasıl bir çıktı üretir?

```
asal_sayilar=[2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[::2])
```

Aşağıdaki işlemlerin sonucunu yazınız.

```
tek_sayilar=[3, 5, 7, 9, 11, 13, 15, 17, 19]
```

print(tek_sayilar[0:6])	
print(tek_sayilar[2:5])	
print(tek_sayilar[3:8])	
print(tek_sayilar[:5])	
print(tek_sayilar[3:])	
print(tek_sayilar[0:8:2])	
print(tek_sayilar[::3])	

Liste elemanını değiştirme

Liste veri tipindeki bir elemanın indeksi kullanılarak yeni değer atanabilir.

Örnek :

```
ornek=['Y','A','N','I','T']  #YANIT kelimesinin harflerinden bir liste oluşturuldu.
print(ornek)  #Liste ekrana yazdırıldı.
ornek[0]='K'  #Listenin ilk elemanı (indeksi sıfır) K olarak değiştirildi.
print(ornek)  #Listenin yeni değeri KANIT kelimesinin harflerine dönüştü.
```

Sıra Sizde:

1. Elemanları sırasıyla sanat, sanat, içindir olan listeyi sanat, toplum, içindir şeklinde değiştiriniz.
2. Değerleri sırasıyla 3,1,2 olan listeyi 1,1,2 olarak değiştiriniz.

Listenin Uzunluğu

Listelerin eleman sayısına ulaşmak için İngilizce uzunluk anlamına gelen length kelimesinin kısaltması olan len() fonksiyonu kullanılır.

Örnek :

```
sayi=[20, 40, 60, 80]
print(len(sayi))
```

Yukarıdaki kodlar çalıştırıldığında dizinin eleman sayısı bulunur. Dizide 4 eleman bulunmaktadır.

in operatörü: Bir elemanın listede olup olmadığını kontrol eder. Eleman listede var ise True yok ise False çıktısını üretir.

Örnek :

```
renkler=["mavi", "yeşil", "kırmızı", "mor"]
print("mavi" in renkler)
```

Ekran Çıktısı:

```
True
```

Örnek :

```
renkler=["mavi", "yeşil", "kırmızı", "mor"]
print("beyaz" in renkler)
```

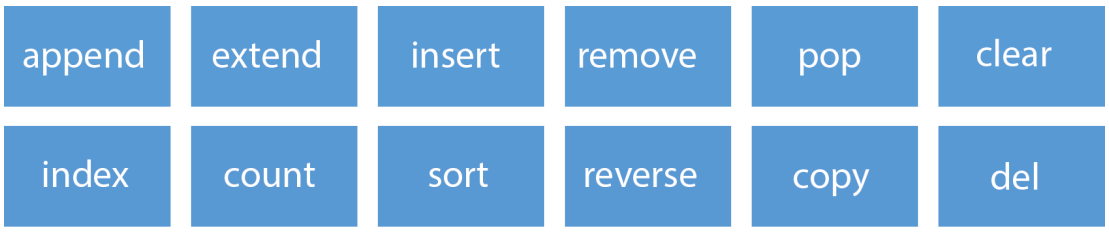
Ekran Çıktısı:

```
False
```

Sıra Sizde: hafta_ici isimli bir liste oluşturarak haftanın günlerini ekleyiniz. Daha sonra sırasıyla Cuma ve Cumartesi günlerinin listede olup olmadığını kontrol ediniz.

Listelerin Fonksiyonları

Liste veri tipinde kullanılabilecek bir dizi fonksiyon bulunmaktadır (Şekil 4.1).



Şekil 4.1: Listelerin fonksiyonları

Fonksiyonları denemek için aşağıdaki gibi bir liste oluşturunuz.

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
print(donanim)
```

1. Append: Listenin sonuna eleman eklemek için kullanılır.

Örnek : Listenin sonuna "bellek" elemanını ekleyiniz.

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
donanim.append("bellek")  #burada append fonksiyonu ile eleman eklenmiştir.
print(donanim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'işlemci', 'bellek', 'sabit disk', 'bellek']

2. Extend: Listeleri birleştirmek için kullanılır. Kullanımı aşağıdaki gibidir:

Örnek : donanim isimli liste ile yazilim isimli listeyi birleştiriniz.

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
yazilim=["işletim sistemi","web tarayıcı"]
donanim.extend(yazilim)  #burada extend fonksiyonu ile listeler birleştirilmiştir.
print(donanim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'işlemci', 'bellek', 'sabit disk', 'işletim sistemi', 'web tarayıcı']

Önemli Not: Birleştirmek için extend fonksiyonu gibi + operatörü de kullanılabilir. Aşağıdaki örnek çalıştırıldığında aynı çıktıyı elde edebilirsiniz.

Örnek :

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
yazilim=["işletim sistemi","web tarayıcı"]
print(donanim+yazilim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'işlemci', 'bellek', 'sabit disk', 'işletim sistemi', 'web tarayıcı']

3. Insert: Listenin belirtilen konumuna (indeksine) eleman eklemek için kullanılır.

Örnek : Listede indeksi 2 olan konuma tarayıcı değerini ekleyiniz.

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
donanim.insert(2,"tarayıcı") #indeksi 2 olan konuma tarayıcı eklenmiştir.
print(donanim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'tarayıcı', 'işlemci', 'bellek', 'sabit disk']

Hatırlatma: Liste konumunu belirleyen indeks 0'dan başlar. Bu nedenle 0-yazıcı, 1-klavye, 2-işlemci'dir. Dolayısıyla tarayıcı değeri 2-işlemci değerinin hemen önüne eklenmiştir.

4. Remove: Listenin içindeki değeri verilen elemanı siler.

Örnek : Listedeki klavye elemanını siliniz.

```
donanim=["yazıcı","klavye","işlemci","bellek","sabit disk"]
donanim.remove("klavye")   #değeri klavye olan eleman silinmiştir.
print(donanim)
```

Ekran Çıktısı: ['yazıcı', 'işlemci', 'bellek', 'sabit disk']

Önemli Not: Liste içindeki herhangi bir eleman indis numarasına göre de silinebilir. Yukarıdaki örnekte klavye elemanını indis kullanarak siliniz. Örnek çalıştırıldığında aynı çıktıyı elde edebilirsiniz.

Örnek : donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]

donanim.remove(donanim[1]) #indis numarası 1 olan eleman "klavye" silinmiştir.

print(donanim)Ekran çıktısı: ['yazıcı', 'işlemci', 'bellek', 'sabit disk']

Ekran Çıktısı: ['yazıcı', 'işlemci', 'bellek', 'sabit disk']

5. Pop: Listede belirtilen konumdaki (indeks) elemanı siler.

Örnek : indeksi 3 olan elemanı siliniz.

donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]

donanim.pop(3)

print(donanim)

Ekran Çıktısı: ['yazıcı', 'klavye', 'işlemci', 'sabit disk']

Önemli Not: pop fonksiyonu ile indeks belirtilmezse son eleman silinir. donanim.pop() yazılırsa son eleman olan sabit disk silinir.

6. Clear: Listenin tüm elemanlarını siler ve boş bir liste ortaya çıkarır.

Örnek : Listenin tüm elemanlarını siliniz.

donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]

donanim.clear()

print(donanim)

Ekran Çıktısı: [ ]

7. Index: Bir elemanın listedeki konumunu bulur.

Örnek : Listedeki "sabit disk" elemanının indeksini bulunuz.

donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]

print(donanim.index("sabit disk"))

Ekran Çıktısı: 4

8. Count: Listede belirtilen elemandan kaç adet olduğunu bulur.

Örnek : Listenin en sonuna bir tane daha klavye elemanı ekleyiniz ve count ile kaç tane klavye elemanı olduğunu bulunuz.

donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk", "klavye"]

say=donanim.count("klavye")

print(say)

Ekran Çıktısı: 2

Bu örnekte say isimli bir değişken tanımlanmış ve count fonksiyonu ile kaç adet klavye kelimesi olduğu bulunmuştur.

9. Sort: Listenin içindeki elemanları sıralar. Burada liste elemanlarının string, int vb. veri tiplerine uygun olarak sıralanacağı unutulmamalıdır.

Örnek : donanim listesini sıralayınız.

```
donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]
donanim.sort()
print(donanim)
```

Ekran Çıktısı: ['bellek', 'işlemci', 'klavye', 'sabit disk', 'yazıcı']

Sıralamanın küçükten büyüğe değil de tam tersi olması için reverse=True parametresi verilebilir. İlgili kod satırını donanim.sort(reverse=True) şeklinde değiştirerek deneyiniz.

10. Reverse: Listeyi sondan başa doğru yani ters yazar.

Örnek : donanim listesini ters bir şekilde yazdırınız.

```
donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]
donanim.reverse()
print(donanim)
```

Ekran Çıktısı: ['sabit disk', 'bellek', 'işlemci', 'klavye', 'yazıcı']

11. Copy: Listeyi yeni bir liste olarak kopyalar.

Örnek : donanim listesini yeni_donanim listesine kopyalayarak ekrana yazdırınız.

```
donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]
yeni_donanim=donanim.copy()
print(yeni_donanim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'işlemci', 'bellek', 'sabit disk']

12. Del: İndeksi verilen elemanı siler. Pop fonksiyonuna benzer bir fonksiyon olmasına rağmen kullanımı farklıdır.

Örnek : indeksi 2 olan elemanı silerek listeyi ekrana yazdırınız.

```
donanim=["yazıcı", "klavye", "işlemci", "bellek", "sabit disk"]
del donanim[2]
print(donanim)
```

Ekran Çıktısı: ['yazıcı', 'klavye', 'bellek', 'sabit disk']

Önemli Not: "pop", "remove" ve "del" fonksiyonları silme işlemi yapar. **remove** fonksiyonunda verilen değer silinirken **pop** ve **del** fonksiyonlarında verilen indekse göre silme işlemi yapılır. pop ve del fonksiyonlarının yazılışı farklıdır.

Sıra Sizde:

1. Adı ders, elemanları sırasıyla B,İ,L,İ,Ş,İ,M olan bir liste oluşturarak aşağıdaki işlemleri yapınız.

   a) Listeyi alfabetik olarak sıralayınız.

   b) Listeyi tersten yazdırınız.

   c) Listede kaç tane İ elemanı olduğunu bulunuz.

   ç) Gerekli harfleri silerek listeyi B,İ,L,İ,M hâline getiriniz.

   d) ders listesini alan listesine kopyalayarak ekrana alan listesini yazdırınız.

   e) Listenin tüm elemanlarını siliniz.

   f) L elemanının indeksini bulunuz.

2. Adı sayilar, elemanları sırasıyla 35, 26, 81, 64 olan bir liste oluşturarak aşağıdaki işlemleri yapınız.

   a) Listeyi büyükten küçüğe doğru sıralayınız.

   b) Listeyi tersten yazdırınız.

   c) Listede kaç tane 26 elemanı olduğunu bulunuz.

   ç) Listedeki 81 sayısını siliniz.

   d) Listenin tüm elemanlarını siliniz.

   e) 64 elemanının indeksini bulunuz.

   f) Listeyi ondalikli_sayilar isimli, elemanları 1.4, 6.8 olan liste ile birleştiriniz.

İç İçe Liste Oluşturma

Python programlama dilinde iç içe liste adı verilen yapı ile bir liste içinde başka listeler de tutulabilir.

Örnek :

```
meyveler=["elma","çilek","armut"]
alisveris_listesi=["süt","peynir",meyveler]
print(alisveris_listesi)
```

Ekran Çıktısı:

```
['süt','peynir',['elma','çilek','armut']]
```

Bu örnekte iki farklı liste bulunmaktadır. Görüldüğü üzere alisveris_listesi içinde meyveler listesi de kullanılmıştır. Ekran çıktısına bakıldığında iki listesinin iç içe kullanıldığı görülmektedir.

Örnek :

```
bellekler=["RAM","ROM"]
ekran_kartlari=["Paylaşımlı","Paylaşımsız"]
sabit_diskler=["SSD"]
birimler=bellekler, ekran_kartlari, sabit_diskler
print(birimler)
```

Ekran Çıktısı:

```
['RAM','ROM','Paylaşımlı','Paylaşımsız','SSD']
```

Bu örnekte üç farklı liste oluşturulmuş ve birleştirilerek birimler listesine eklenmiştir.

```
bellekler=["RAM","ROM"]
ekran_kartlari=["Paylaşımlı""Paylaşımsız"]
sabit_diskler=["SSD"]
birimler=bellekler,ekran_kartlari,sabit_diskler
print(birimler[0][1])
```

Bu örnekte de birleştirilen listelerden indeksi 0 olan (bellekler) listenin, indeksi 1 olan elemanı (ROM) ekrana yazdırılmıştır.

```
bellekler=["RAM", "ROM"]
ekran_kartlari=["Paylaşımlı""Paylaşımsız"]
sabit_diskler=["SSD"]
birimler=bellekler,ekran_kartlari,sabit_diskler
print(birimler[0][1], birimler[2][0])
```

Bu değişiklikle bir önceki çıktının yanına indeksi 2 olan listenin (sabit_diskler) 0. indeksli elemanı olan SSD yazdırılır.

Sıra Sizde: 5 ile 15 (15 dâhil) arasındaki tek sayıları bir listeye alınız. 6 ile 16 (16 dâhil) arasındaki çift sayıları da başka bir listeye alınız.

a) Oluşturduğunuz tek sayılar listesine çift sayıları ekleyerek iç içe bir liste hazırlayınız.

b) Ekran çıktısı olarak 7 14 üreten kodu yazınız.

c) Ekrana sırasıyla çift sayılar listesinden 10 ve 12; tek sayılar listesinden 13 yazdırınız.

4.3.4. Tuple (Demet) Veri Tipi

Listeler konusunda oluşturulan listeler üzerinden daha sonra değişiklikler yapılabildiğini gördünüz. Tuple veri tipi de listelere oldukça benzemektedir. Aralarındaki temel fark ise tuple veri tipinin tanımlandıktan sonra değişikliğe yani eleman ekleme ya da silmeye izin vermemesidir. Tuple veri tipi ile yapılabilecek işlemler şu şekildedir:

1. Tuple oluşturma: Tuple tanımlaması yapılırken listelerden farklı olarak parantezler kullanılır.

Örnek :

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler)
```

Ekran Çıktısı: ('bit', 'inç', 'byte', 'hertz', 'piksel')

Listelere benzer şekilde tuple oluşturulur ve ekrana yazdırılır.

2. Tuple elemanlarına ulaşma: Listelerdeki gibi indeks kullanılır.

Örnek :

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler[3])
```

Ekran Çıktısı: hertz

Önemli Not: Listelerde olduğu gibi negatif indekslerde kullanılabilir. -1 en sondaki eleman anlamına gelirken -2 sondan iki önceki elemanı temsil eder.

Örnek :

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler[-3])
```

Ekran Çıktısı: byte

3. Indeks aralıklarına göre yazdırma: Listelerde olduğu gibi başlangıç ve bitiş indeksleri verilerek istenilen aralık yazdırılabilir.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
print(birimler[1:3])
```

Ekran Çıktısı: (‘inç’, ‘byte’)

Bu örnekte indeksi 1 olan inç değerinden başlanarak indeksi 3 olan hertz (dâhil değil) değerine kadar olan elemanlar ekrana yazdırılmıştır.

4. Tuple elemanlarını değiştirme: Tuple veri tipi tanımlanırken elemanların değiştirilemeyeceğinden bahsedildi. Eğer tuple veri tipi listeye çevrilirse elemanlar değiştirilebilir.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
birimler_liste=list(birimler)  #burada tuple listeye çevrildi.
birimler_liste[2]=”mega byte”  #listenin indeksi 2 olan elemanı değiştirildi.
print(birimler_liste)
```

Ekran Çıktısı: [‘bit’, ‘inç’, ‘mega byte’, ‘hertz’, ‘piksel’]

5. Elemanın olup olmadığını sorgulama: Tuple veri tipinde de listelerde olduğu gibi in operatörü ile bir elemanın listede olup olmadığı kontrol edilebilir. Eleman tuple’daysa True; yoksa False değerleri üretilir.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
print(“bit” in birimler)
```

Ekran Çıktısı: True

6. Tuple uzunluğunu bulma: len fonksiyonu ile tuple’ın eleman sayısı bulunur.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
print(len(birimler))
```

Ekran Çıktısı: 5

7. Tuple içinde bir elemanın sayısını bulma: Bu işlem için listelerde olduğu gibi count fonksiyonu kullanılır.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
say=birimler.count(“piksel”)
print(say)
```

Ekran Çıktısı: 1

8. Tuple içindeki elemanın indeksini bulma: Listelerde olduğu gibi index fonksiyonu kullanılır.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
print(birimler.index(“byte”))
```

Ekran Çıktısı: 2

9. Tuple birleştirme: Birden fazla tuple birleştirilerek tek bir tuple'da toplanabilir.

Örnek :

```
birimler = (“bit”, “inç”, “byte”, “hertz”, “piksel”)
degerler=(8,256,1024)
birlestir=birimler+degerler
print(birlestir)
```

Ekran Çıktısı: (‘bit’, ‘inç’, ‘byte’, ‘hertz’, ‘piksel’, 8, 256, 1024)

4.3.5. Dictionary (Sözlük) Veri Tipi

Python programlama dilinde sırasız, değiştirilebilir ve belirli bir konuma sahip koleksiyonlar sözlük olarak adlandırılır. Sözlükler süslü (ya da kırlangıç{}) parantezler arasına yazılır. Sözlük veri tipinde anahtarlar ve bu anahtarların değerleri vardır. Her anahtardan sonra iki nokta (:) kullanılır ve değer yazılır. Anahtar:değer (key:value) ikilileri virgülle birbirinden ayrılır.

Hatırlatma: Liste veri tipinde köşeli parantez [ ], demet veri tipinde normal parantez ( ), sözlük veri tipinde ise süslü parantez { } kullanılır.

Farklı şekillerde tanımlanabilen sözlük veri tipinin genel kullanımı şu şekildedir:

```
sozluk_adi={anahtar:deger}
```

Örnek :

```
sozluk = {“Mesleğiniz”:”Öğrenci”, “Alanınız”:”Bilişim”, “Yaşadığınız Yer”:”Ankara” }
print(sozluk)
```

Ekran Çıktısı: {‘Mesleğiniz’: ‘Öğrenci’, ‘Alanınız’: ‘Bilişim’, ‘Yaşadığınız Yer’: ‘Ankara’}

Sözlükte sadece anahtarları göstermek için keys ve values fonksiyonları kullanılır.

Örnek :

```
sozluk = {“Mesleğiniz”:”Öğrenci”, “Alanınız”:”Bilişim”, “Yaşadığınız Yer”:”Ankara” }
print(sozluk.keys())
print(sozluk.values())
```

Ekran Çıktısı: dict_keys([‘Mesleğiniz’, ‘Alanınız’, ‘Yaşadığınız Yer’])
dict_values([‘Öğrenci’, ‘Bilişim’, ‘Ankara’])

Sözlük veri tipi ile yapılabilecekler genel olarak şu şekildedir:

1. Sözlük elemanlarına erişim aşağıdaki şekilde yapılmaktadır.

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print(donanim["Türü"])
```

Ekran Çıktısı: RAM

2. Sözlük içindeki değerleri değiştirebilirsiniz. Aşağıda değer değişimine yönelik bir örnek bulunmaktadır.

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim["Kapasitesi"]="16 GB" #burada 8 GB değeri, 16 GB değeri ile değişti.
print(donanim)
```

Ekran Çıktısı: {'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '16 GB'}

3. Diğer veri tiplerinde olduğu gibi sözlüklerde de bir değerin olup olmadığına "in" operatörü ile bakılabilir.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print("Türü" in donanim)  #Sözlükte Türü anahtarının olup olmadığı kontrol edilmiştir.
```

Ekran Çıktısı: True

4. Sözlüklerde de uzunluk len fonksiyonu ile bulunur. Burada eleman sayısının anahtar-değer ikilileri olarak hesaplanacağını unutmayınız.

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print(len(donanim))
```

Ekran Çıktısı: 3

5. Sözlüğe daha sonra anahtar-değer ikilileri eklenebilir. Aşağıdaki örnekte ikinci satıra dikkat ediniz.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim["Hızı"]="2400 MHz"    #burada ekleme işlemi yapılmıştır.
print(donanim)
```

Ekran Çıktısı: {'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '8 GB', 'Hızı': '2400 MHz'}

6. Sözlük veri tipinde silme işlemi yapmak için pop fonksiyonu kullanılır.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim.pop("Kapasitesi") #burada silme işlemi yapılmıştır.
print(donanim)
```

Ekran Çıktısı: {'Türü': 'RAM', 'Tipi': 'DDR4'}

7. del fonksiyonu ile sözlük tamamen silinebilir.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
del donanim
print(donanim)
```

Ekran Çıktısı: name 'donanim' is not defined (donanım tanımlanmadı)

8. Sözlüğü silmek yerine içini boşaltmak için clear() fonksiyonu kullanılır.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim.clear()
print(donanim)
```

Ekran Çıktısı: {}

9. Sözlüğü kopyalamak için listelerde olduğu gibi copy() fonksiyonu kullanılır.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
yeni_donanim=donanim.copy()
print(yeni_donanim)
```

Ekran Çıktısı: {'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '8 GB'}

10. Bir sözlük kendi içinde birden fazla sözlük barındırabilir. Birden fazla sözlük yeni bir sözlükte birleştirilebilir.

Örnek :

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim2= {"Türü":"Sabit Disk", "Tipi":"SSD", "Kapasitesi":"1 TB"}
donanimlarim={"donanim":donanim, "donanim2":donanim2}
print(donanimlarim)
```

Ekran Çıktısı: {'donanim': {'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '8 GB'}, 'donanim2': {'Türü': 'Sabit Disk', 'Tipi': 'SSD', 'Kapasitesi': '1 TB'}}

Sıra Sizde: Aşağıdaki sözlükleri oluşturarak sizlerden istenen işlemleri yapınız.

sozluk = {"Bilim insanı":"Aziz Sancar", "Şair":"Mehmet Akif Ersoy", "Astronom":"Ali Kuşçu"}

a) sozluk isimli sözlüğü meslekler isimli başka bir sözlüğe kopyalayınız ve ekrana yazdırınız.

b) sozluk isimli sözlüğün değerlerini ekrana yazdırınız.

c) sozluk isimli sözlüğü içi boş bir sözlük hâline getiriniz.

ç) sozluk isimli sözlüğe Matematikçi: Cahit Arf ikilisini ekleyiniz.

d) sozluk isimli sözlüğün içinde sanatçı anahtarının olup olmadığını sorgulayınız.

e) sozluk isimli sözlüğün bilim insanı anahtarındaki değeri Canan Dağdeviren olarak değiştiriniz.

f) sozluk isimli sözlüğün şair anahtarı ile eşleşen değeri ekrana yazdırınız.

onemli_telefonlar = {"Acil Çağrı Merkezi":"112", "Polis İmdat":"155", "Milli Eğitim Bakanlığı İletişim Merkezi":"444 0 632"}

a) önemli_bilgiler isimli sözlüğün değerlerini ekrana yazdırınız.

b) önemli_bilgiler isimli sözlüğü siliniz.

c) önemli_bilgiler isimli sözlükten Acil Çağrı Merkezi anahtarını ve değerini siliniz.

ç) önemli_bilgiler isimli sözlükte Sağlık Bakanlığı İletişim Merkezi olup olmadığını sorgulayınız.

d) önemli_bilgiler isimli sözlüğü içi boş bir sözlük hâline getiriniz.

4.3.6. Set (Küme) Veri Tipi

Python programlama dilinde kullanılan veri tiplerinden biri de set (küme) veri tipidir. Sözlükler gibi süslü parantezlerin içine yazılan set veri tipi, sözlüklerden farklı olarak ikili anahtar yapısında değildir. Set veri tipinde elemanlar sırasızdır ve tekrar etmez. Türkçeye küme olarak çevrilen bu veri tipi bir dizi matematiksel işlemin kolaylaştırılmasını sağlar.

Set veri tipinin basit kullanımı şu şekildedir:

```
sayilar = {1, 2, 3, 4, 5}  #integer veri tipi tırnak içinde yazılmaz.
print(sayilar)
```

Ekran Çıktısı: {1, 2, 3, 4, 5}

Set veri tipinde de fonksiyonlar kullanılarak bir dizi işlem yapılabilir. Bu fonksiyonlar genel olarak liste, sözlük ve demet veri tipindeki fonksiyonlarla benzerdir. Aşağıda bu fonksiyonlara bazı örnekler verilmiştir:

1. Bir elemanın küme içinde olup olmadığı in fonksiyonu ile kontrol edilir.

```
sayilar = {1, 2, 3, 4, 5}
print(6 in sayilar)
```

Ekran Çıktısı: False

2. Küme veri tipinde eleman eklemek için add() fonksiyonu kullanılır.

Örnek :

```
sayilar = {1, 2, 3, 4, 5}
sayilar.add(6)
print(sayilar)
```

Ekran Çıktısı: {1, 2, 3, 4, 5, 6}

Tek bir eleman yerine birden fazla eleman eklemek için update() fonksiyonu kullanılır.

Örnek :

```
sayilar = {1, 2, 3, 4, 5}
sayilar.update([6,7,8])
print(sayilar)
```

Ekran Çıktısı: {1, 2, 3, 4, 5, 6, 7, 8}

3. Set içindeki bir elemanı silmek için remove() ya da discard() fonksiyonları kullanılır. Her iki fonksiyonunun kullanımı aynıdır.

Örnek :

```
sayilar = {1, 2, 3, 4, 5}
sayilar.discard(3)
print(sayilar)
```

Ekran Çıktısı: {1, 2, 4, 5}

Önemli Not: Verilen örneklerde sadece integer tipi kullanılmış olsa da set veri tipinde farklı veri tiplerini (aynı kümede integer, string veya float gibi) aynı anda kullanabilirsiniz.

Sıra Sizde: Set veri tipinde kullanılan diğer fonksiyonları araştırınız ve uygulayınız.

ÖLÇME VE DEĞERLENDİRME 4

1. Aşağıdaki değişken tanımlamalarından hangisi yanlıştır?

   A) adres B) 3gen C) pi_sayisi

   D) ortalama E) okulno

2. 3**3= işleminin sonucu hangisidir?

   A) 3 B) 6 C) 9 D) 18 E) 27

3. a==b ifadesinde hangi operatör kullanılmıştır?

   A) Atama B) Kimlik C) Karşılaştırma

   D) Mantıksal E) Aritmetiksel

4. Aşağıdaki fonksiyonların hangisi ile kullanılan veri tipi öğrenilebilir?

   A) type B) print C) str D) update E) len

5. Sadece True ve False değerlerini döndüren veri tipi hangisidir?

   A) int B) string C) float D) complex E) bool

```
kaynaklar=["Kitap", "Makale", "Tez", "Rapor", "Bildiri"]
print(kaynaklar[1])
```

6. Yukarıdaki kod nasıl bir çıktı üretir?

   A) Kitap B) Makale C) Tez D) Rapor E) Bildiri

```
sayilar=[1, 3, 5, 7, 9, 11, 13]
print(sayilar[1:6:2])
```

7. Yukarıdaki kod nasıl bir çıktı üretir?

   A) [1, 3, 7, 11]

   B) [3, 5, 7, 9, 11, 13]

   C) [3, 7, 11]

   D) [3, 7, 11, 13]

   E) [1, 3, 5, 7, 9, 11, 13]

```
donanim=["fare", "klavye", "hoparlör", "bellek", "ekran"]
donanim.pop(2)
print(donanim)
```

8. Yukarıdaki kod nasıl bir çıktı üretir?

   A) ['fare', 'klavye', 'bellek', 'ekran']
   B) ['fare', 'klavye', 'hoparlör']
   C) ['fare', 'klavye', 'hoparlör', 'bellek', 'ekran']
   D) ['fare', 'klavye']
   E) ['klavye', 'hoparlör', 'bellek', 'ekran']

```
egitim = {"Okul":"School", "Öğrenci":"Student", "Öğretmen":"Teacher" }
```

9. Yukarıdaki kod satırında hangi veri tipi kullanılmıştır?

   A) String B) Liste C) Set D) Sözlük E) Demet

```
birimler = ("bit", "inç", "byte", "hertz", "piksel", "bit", "byte")
say=birimler.count("bit")
print(say)
```

10. Yukarıdaki kod nasıl bir çıktı üretir?

    A) 5 B) 4 C) 3 D) 2 E) 1

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.

ÖĞRENME BİRİMİ 5

KARAR VE DÖNGÜ YAPILARI

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Karar yapısı kullanımlarını öğrenebilecek,
- If ve if-elif yapılarını kullanabilecek,
- Döngü mantığını anlayabilecek,
- Döngü türlerini kullanabileceksiniz.

Anahtar Kelimeler:

Karar yapısı, döngüler.

Hazırlık Çalışmaları

1. Karar yapılarını araştırarak günlük hayatta nasıl kullanıldığını tartışınız.
2. Programlama dillerinde kullanılan döngülerin sağladığı kolaylıkları araştırınız.

5. KARAR VE DÖNGÜ YAPILARI

5.1. Karar Yapıları

Günlük hayatta sık sık karar vermeyi gerektiren durumlarla karşılaşılmaktadır. Programlamada da benzer olarak karar yapıları kullanılmaktadır. Örneğin teneffüste çay ya da kahve arasında bir seçim yapma karar verme sürecidir. Karar verme sürecinde eldeki verilerle bir değerlendirme yapılmaktadır. Bir önceki teneffüste çay içilmesi bu teneffüsteki kararı etkiler ve belki de kahve tercihini daha cazip hâle getirir. Bu gibi örnekleri çoğaltmak mümkündür.

5.1.1. If-Else Yapısı

Python programlama dilinde (ve birçok diğer dilde) karar yapıları if (eğer) ile temsil edilmektedir. Bu yapıda bir durumun doğru (true) ya da yanlış (false) olma durumuna göre bazı eylemler icra edilmektedir. if yapısı tek başına kullanıldığı gibi else ile birlikte de kullanılabilir. Else anahtar sözcüğü tek başına kullanılmaz. Özetle; if "eğer" olarak, else ise "değilse" olarak düşünülebilir. Kullanıcının girdiği yaş 18 ve daha büyükse ekrana "ehliyet alabilir"; değilse ekrana "ehliyet alamaz" gibi uyarılar vermek bu yapının bir örneğidir. Bu örnek, programlama dilinde şu şekilde yazılır:

```
yas=int(input("Yaşınızı girin:"))
if yas>= 18:
   print("Ehliyet alabilirsiniz")
else:
   print("Ehliyet alamazsınız")
```

Örnekte int veri tipinde yas isimli bir değişken tanımlandı. input() fonksiyonu ile kullanıcıdan veri alındı. if satırında ise yas>=18 şartı sorgulandı. Bu şart doğru ise "Ehliyet alabilirsiniz" yanlış ise "Ehliyet alamazsınız" uyarılarının ekran çıktısı olması sağlandı. Burada else, doğru değilse anlamında kullanıldı. Ayrıca kod yapısı incelendiğinde farklı bir girinti yapısı da görülmektedir. Python programlama dilinde girinti yapısı örnekteki gibidir. if ve else satırının sonunda : (iki nokta) kullanıldığına dikkat edilmelidir. print satırlarında ise satırın içten başladığı görülmektedir. Klavyede bulunan tab tuşu ile bu girinti ayarlanabilir.

Örnek 1 : Kullanıcının girdiği sayı çift ise "Çift sayı"; değilse "Tek sayı" uyarılarını veren kodu yazınız.

```
girilen_sayi=int(input("Bir sayı girin:"))
if girilen_sayi%2==0:
   print("Çift sayı")
else:
   print("Tek sayı")
```

Bu örnekte kullanıcıdan bir sayı alınmıştır. Bir sayının çift olması, o sayının ikiye kalansız bölünmesi ile açıklanır. Bu nedenle bir mod alma işlemi yapılmıştır. girilen_sayi%2==0 şartı sayının 2'ye bölündüğünde kalanın 0 (sıfır) olması anlamına gelmektedir.

Önemli Not: Bu örnekteki == karşılaştırma operatörüdür. Atama operatörü ile karıştırılmaması gerekir.

Sıra Sizde:

a) Kullanıcıdan iki sınav ve bir performans notu girmesini isteyiniz. Girilen 3 notun ortalaması 50 ve daha büyükse "Başarılı"; değilse "Başarısız" çıktıları veren kodu yazınız.

b) Bir üçgenin iç açıları toplamı 180 derecedir. Kullanıcının girdiği üç açı değerine göre "Bu bir üçgendir." ya da "Bu bir üçgen değildir." çıktıları veren kodu yazınız.

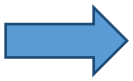
c) Bir hava yolu firması en fazla 20 kilogram bagaj hakkı vermektedir. 20 kilogramdan sonraki her kilogram için 10 TL ek ücret almaktadır. Buna göre bagajı 20 kg ya da daha az olan yolculara "Herhangi bir ücret ödemeniz gerekmiyor."; 20 kg'den fazla olanlar için de ne kadar ek ücret ödeneceğini hesaplayarak "Fazla bagaj için ….. TL ödemelisiniz." çıktılarını veren kodu yazınız.

Not: Bu soruda kilogram hesabında sadece tam sayıları dikkate alınız. Örneğin 28,70 kilogram olan bagaj için sadece 8 kg için ek ücret ödenmesi yeterlidir.

ç) Kullanıcının girdiği iki ürünün toplam fiyatı 200 TL ve altıysa "Ödenecek miktar=…. TL"; 200 TL'yi geçerse %25 indirim yaparak "Ödenecek miktar, indirimden sonra ….. TL'dir." çıktılarını veren kodu yazınız.

Önemli Not: Tek satırlık ifadeler, Python'da tek satırlık kodların if ifadesinin yanına yazılmasını desteklemektedir. Aşağıdaki örnekte solda verilen kod bloğu sağdaki gibi tek satır hâlinde de yazılabilir.

```
a = 2
b = 330
if a>b:
   print("A")
else:
   print("B")
```



```
a = 2
b = 330
print("A") if a > b else print("B")
```

Örnek 2: cikis_birimleri isimli bir liste oluşturarak yazıcı, hoparlör ve ekran elemanları eklensin. If yapısı kullanılarak ekran elemanı listede varsa "Eleman bulundu."; yoksa "Eleman bulunamadı." çıktıları veren kodu yazınız.

```
cikis_birimleri=["yazıcı","hoparlör","ekran"]
if "ekran" in cikis_birimleri:
   print("Eleman bulundu.")
else:
   print("Eleman bulunamadı.")
```

Ekran Çıktısı: Eleman bulundu.

Sıra Sizde: haftaici isimli bir liste oluşturarak Pazartesi, Salı, Çarşamba, Perşembe ve Cuma elemanları eklensin. If yapısı kullanılarak Cumartesi elemanı listede varsa "Listede bulundu."; yoksa "Listede bulunamadı." çıktıları veren kodu yazınız.

Örnek 3 : Kullanıcıya yabancı dil ve ofis programlarını bilip bilmediği sorulsun. Her iki soruya da "Evet" cevabı verilirse "İşe alındınız."; diğer durumlarda ise "İşe alınmadınız." çıktıları veren programı yazınız.

```
yabanci_dil=input("Yabancı dil biliyor musunuz? (Evet/Hayır): ")
ofis_programlari=input("Ofis programlarını biliyor musunuz? (Evet/Hayır): ")
if yabanci_dil=="Evet" and ofis_programlari=="Evet":
   print("İşe alındınız.")
```

```
else:
  print("İşe alınmadınız.")
```

Bu örnekte iki değişken tanımlanmıştır. Her iki değişkene verilen cevabın da "Evet" olması durumunda koşullar sağlanarak "İşe alındınız." çıktısı verilmiştir. Burada and operatörünün kullanıldığı görülmektedir. Bu operatör tüm şartların doğru olmasını gerektirir. Dikkat edilmesi gereken başka bir nokta da string ifadeler karşılaştırılırken tırnak işaretinin kullanılmasıdır.

Sıra Sizde:

a) Kullanıcıdan kullanıcı adı ve şifre girilmesi istensin. Kullanıcı adı "Türkiye"; şifre 1923 ise "Giriş başarılı"; değilse "Kullanıcı adı ya da şifre yanlış" çıktıları veren kodu yazınız.

b) Girilen sayı hem 3 hem de 5'e tam bölünüyorsa "15'e tam bölünür."; bölünmüyorsa "15'e tam bölünmez." çıktıları veren kodu yazınız.

Örnek 4 : Bir mülakatta katılımcının başarılı olabilmesi için İngilizce ya da Fransızcadan birini bilmesi ve yaşının 40'tan küçük olması gerekmektedir. Katılımcıya yukarıdaki bilgileri, adını ve soyadını sorarak mülakat sonucunu "Başarılı" ya da "Başarısız" çıktıları ile gösteriniz.

```
ad_soyad=input("Adınız-Soyadınız: ")
yabanci_dil=input("Bildiğiniz yabancı dil: ")
yas=int(input("Yaşınız: "))
if ((yabanci_dil=="İngilizce" or yabanci_dil=="Fransızca") and yas<40):
  print("Sayın "+ad_soyad+", sonuç başarılı")
else:
  print("Sayın "+ad_soyad+", sonuç başarısız")
```

Ekran Çıktısı:

```
Adınız-Soyadınız: Canan Yılmaz
Bildiğiniz yabancı dil: Fransızca
Yaşınız: 32
Sayın Canan Yılmaz, sonuç başarılı
```

Bu örnekte mantıksal operatörlerden and ve or bir arada kullanılmıştır. or operatöründe şartlardan birinin doğru olması yeterliyken and operatöründe tüm şartların doğru olması gerekmektedir. if satırına bakıldığında yabanci_dil değişkeninin İngilizce ya da Fransızca olması; ayrıca yas değişkeninde de 40 değerinden küçük olması istenmektedir. print satırında ise + operatörü ifadeleri birleştirmek amacıyla kullanılmıştır.

Sıra Sizde: Bir programın bilgisayara kurulması için i7 işlemci ya da en az 8 GB RAM belleğe ihtiyaç duyulmaktadır. Şartlar sağlanıyorsa "Kurulum uygun"; sağlanmıyorsa "Kurulum uygun değil" çıktıları veren programı yazınız.

5.1.2. If-Elif-Else Yapısı

Daha önce yapılan karar yapısı örneklerinde eğer-değilse yapısı kullanıldı. Başka bir ifadeyle şart doğru ise bir durum, yanlış ise başka bir durum vardı. Bazen tek bir şartın değil de daha fazla şartın olduğu durumlar da ortaya çıkmaktadır. Bu gibi durumlarda if-elif-else yapısı kullanılır. Bu yapıda ilk şart if; aradaki şartlar elif; değilse kısmında da else sıralaması bulunmaktadır. Örneğin bir sayının pozitif olup olmadığının öğrenilmeye çalışıldığı bir durumda if-else yapısı yetersiz kalacaktır. Çünkü sayı 0'dan büyükse "Pozitif", küçükse "Negatif" olacağı gibi sayı sıfıra eşit de olabilir. Örnekte bu durumun if-elif-else yapısı ile kodlanması görülmektedir:

```
sayi=int(input("Bir sayı girin: "))
if sayi>0:
```

```
    print("Pozitif")
elif sayi<0:
    print("Negatif")
else:
    print("Sayı sıfıra eşittir")
```

Ekran Çıktısı:

```
Bir sayı girin: -5
Negatif
```

Bu örnekte birden fazla şart olduğu için if-elif-else yapısı kullanılmıştır. Bu yapıda ilk şart olan sayi>0 if satırına; sayi<0 ise elif satırına yazılmıştır. Örnekte üç durum yaşanabilir. Sayı 0'dan büyük, 0'dan küçük ya da 0'a eşit olabilir. Bu nedenle ilk iki şartı yazdıktan sonra başka bir ihtimal olmayacağı için sayi==0 yazmak yerine else anahtar sözcüğünü yazmak yeterlidir.

Sıra Sizde: Girilen plaka kodu 06 ise ekrana Ankara, 07 ise Antalya, 08 ise Artvin, bunların dışında girilen tüm değerlerde ise Türkiye çıktısı veren kodu yazınız.

Örnek 5: Girilen iki sayıya ve operatöre (+,-,*,/) göre toplama, çıkarma, çarpma ya da bölme işlemlerini yapan; bu operatörler dışında bir değer girildiğinde "Yanlış işlem girdiniz." uyarısı veren kodu yazınız.

```
sayi1=int(input("Birinci sayıyı girin:"))
sayi2=int(input("İkinci sayıyı girin:"))
islem=input("İşlem seçin (+,-,*,/):")
if islem=="+":
    sonuc=sayi1+sayi2
    print(sonuc)
elif islem=="-":
    sonuc=sayi1-sayi2
    print(sonuc)
elif islem=="*":
    sonuc=sayi1*sayi2
    print(sonuc)
elif islem=="/":
    sonuc=sayi1/sayi2
    print(sonuc)
else:
    print("Yanlış işlem girdiniz")
```

Ekran Çıktısı:

```
Birinci sayıyı girin: 10
İkinci sayıyı girin: 20
İşlem seçin (+,-,*,/): -
-10
```

http://kitap.eba.gov.tr/KodSor.php?KOD=22356

Örnek 6: Yaşam süresinin artmasından sonra yaş grupları aşağıdaki gibi değerlendirilmeye başlanmıştır. Girilen doğum tarihine göre kişinin yaş grubunu ekrana yazdıran kodu yazınız.

```
0-17 yaş arası: Çocuk
18-65 yaş arası: Genç
66-79 yaş arası: Orta Yaşlı
80 yaş ve üstü: Yaşlı
dogum_tarihi=int(input("Doğum yılınızı girin: "))
yas=2020-dogum_tarihi
if yas>=0 and yas<=17:
  print("Çocuk")
elif yas>=18 and yas<=65:
  print("Genç")
elif yas>=66 and yas<=79:
  print("Orta Yaşlı")
elif yas>=80:
  print("Yaşlı")
else:
  print("Yanlış değer girdiniz")
```

Ekran Çıktısı:

```
Doğum yılınızı girin: 1965
Genç
```

Sıra Sizde:

a) Kullanıcı tarafından girilen hava sıcaklığı 5 °C ve altındaysa "Soğuk"; 6-14 °C arasındaysa "Ilık"; 15 °C ve daha fazlaysa "Sıcak" çıktılarını veren kodu yazınız.

b) Bir otoparkın ücret tarifesi aşağıdaki gibidir:

1 saate kadar: 5 TL

1-5 saat arası: Saat başı 4 TL

5 saatten fazla: Saat başı 3 TL

Buna göre kullanıcının girdiği otoparkta kalınan saat süresine göre ödenecek miktarı bularak ekrana yazdırınız.

c) Üçgenler kenar uzunluklarına göre üçe ayrılmaktadır: Eşkenar, İkizkenar ve Çeşitkenar. Kullanıcının girdiği 3 kenar uzunluğuna göre üçgenin türünü ekrana yazdırınız.

ç) Kullanıcının girdiği boy ve ağırlık değerlerine göre vücut kitle indeksini (VKİ=ağırlık/(boy*boy), boy metre cinsinden verilmeli) hesaplayınız.

VKİ 18 ile < 25 aralığındaysa normal,

VKİ 25 ile <30 aralığındaysa kilolu,

VKİ 30 ve daha yüksekse obez,

VKİ 35 ve daha fazlaysa ciddi obez olarak kabul edilir.

VKİ'ni hesaplayarak kişinin durumunu yazdırınız.

d) Kullanıcıdan adını, maaşını ve çalışma yılını girmesini isteyiniz. 0-5 yıl arası çalışanlara %10; 6-10 yıl arası

çalışanlara %15; 11 ve daha fazla yıl çalışanlara %25 zam yapılmaktadır. Buna göre "Sayın ……..., zamlı maaşınız …….. TL" çıktısı veren kodu yazınız.

e) Girilen üç sayıdan en büyüğünü bulan kodu yazınız.

5.1.3. İç İçe İfadeler

Önceki konuda mantıksal operatörleri kullanarak birden fazla durumun kontrolü sağlanmıştır. İç içe ifadeler de birden fazla durumun kontrol edilmesi gerektiğinde kullanılır.

Örnek 7: Bir firma işe alımlarda 40 yaş altı kişileri tercih etmektedir. Bu şartı sağlayan kişilerde de sürücü belgesi olan üniversite mezunlarını tercih etmektedir. Buna göre kullanıcıya önce yaşı sorulsun. Yaşı 40 altı olmayanlara "Üzgünüz, kriterlerimize uymuyorsunuz." uyarısı verilerek programdan çıkılırken; yaş şartı uyanlara diğer iki soruyu sorarak işe alınıp alınmadıklarını çıktı olarak veren kodu yazınız.

```
yas=int(input("Yaşınızı girin: "))
if yas<40:
    mezuniyet=input ("Üniversite mezunu musunuz? (E/H):")
    surucu_belgesi=input ("Sürücü belgeniz var mı? (E/H):")
    if mezuniyet=="E" and surucu_belgesi=="E":
        print("Tebrikler, işe alındınız")
    else:
        print("İşe alınmadınız")
else:
    print("Üzgünüz, kriterlerimize uymuyorsunuz")
```

Ekran Çıktısı 1 :

```
Yaşınızı girin: 46
Üzgünüz, kriterlerimize uymuyorsunuz.
```

Ekran Çıktısı 2 :

```
Yaşınızı girin: 35
Üniversite mezunu musunuz? (E/H):E
Sürücü belgeniz var mı? (E/H):E
Tebrikler, işe alındınız.
```

Bu örnekte birden fazla durumun kontrolü iç içe if yapısı ile sağlanmıştır. Kullanıcıya sorulan yaş bilgisine göre; yaşı 40 altı olmayanlara herhangi bir soru sormadan programdan çıkılırken diğer adaylara ek sorular sorulmuştur.

Sıra Sizde: Kullanıcıya sinema ya da tiyatro tercihi sorulsun. Sinema izlemek için 15 TL, tiyatro için 10 TL ödenmesi gerekmedir. Öğrencilere %50 indirim yapıldığı düşünülerek öğrenci ise indirim yapılan; öğrenci değilse indirimsiz tutarı hesaplayarak ekrana yazdıran kodu yazınız.

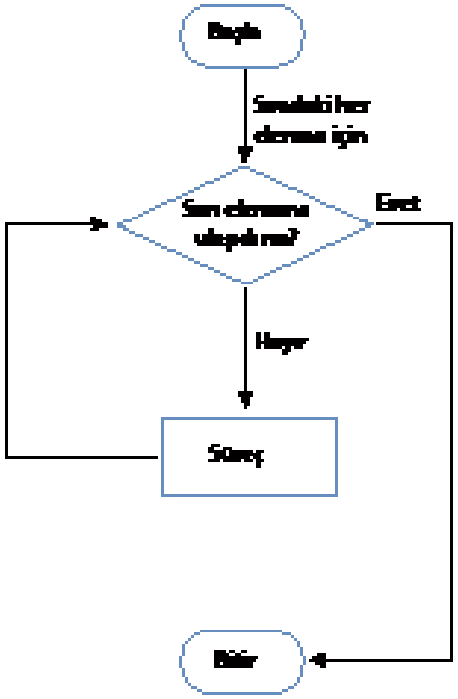
5.2. Döngüler

Programlama dillerinde karar yapıları gibi sık kullanılan başka bir yapı da döngülerdir. Program içinde kod bloklarının istenen sayıda tekrar etmesini sağlayan yapılara döngü adı verilir. Python programlama dilinde for ve while döngüleri bulunmaktadır.

Görsel 5.1: Döngüler

5.2.1. For Döngüsü

Şart doğru olduğu sürece işlemlerin tekrarını sağlayan döngü yapısıdır. For döngüsü belirli bir şart sağlanana kadar belirlenen kod bloklarını tekrarlar. For döngüsünün yapısı aşağıdaki gibidir:



Şekil 5.1: For döngüsünün yapısı

5. 2. 1. 1. Range Kullanımı

Döngünün başlangıç ve bitiş değeri belli olan durumlarda kullanılan fonksiyondur. Varsayılan olarak 0'dan (sıfır) başlayarak birer birer artar. Range fonksiyonunun bitiş değeri döngü dışında kabul edilir.

Örnek 8:

```
for sayilar in range(10):
    print(sayilar)
```

Ekran Çıktısı:

```
0
1
2
3
4
5
6
7
8
9
```

Bu örnekte bir başlangıç değeri verilmediği için döngü 0'dan (sıfır) başlar ve 10'a kadar devam eder (10 hariç). sayilar ismiyle oluşturulan değişken print fonksiyonu ile ekrana yazdırılmıştır.

```
for sayilar in range(5,10):
  print(sayilar)
```

Ekran Çıktısı:

```
5
6
7
8
9
```

Bu örnekte başlangıç ve bitiş değerleri birlikte verilmiştir. Ekran çıktısına bakıldığında başlangıç değeri olan 5'ten başlayarak bitiş değerine kadar olan sayılar (bitiş değeri dâhil değil) ekrana yazdırılmıştır.

Örnek 9:

```
for sayilar in range(5,20,3):
  print(sayilar)
```

Ekran Çıktısı:

```
5
8
11
14
17
```

Bu örnekte başlangıç ve bitiş değerleri ile artış değeri de verilmiştir. Başka bir ifadeyle döngünün 5'ten başlayarak 20'ye kadar 3'er 3'er artması sağlanmıştır. Burada yine dikkat edilmesi gereken nokta 17 sayısından sonra 20 sayısının son değer olduğu için çıktıda görülmemesidir.

Örnek 10 :

```
for sayilar in range(20,5,-3):
    print(sayilar)
```

Ekran Çıktısı:

```
20
17
14
11
8
```

Bu örnekte 20'den başlayarak 5'e kadar (5 dâhil değil) 3'er azalan sırada sayılar yazdırılmıştır.

Sıra Sizde:

a) 0-20 arası çift sayıları for döngüsü ile ekrana yazdırınız.

b) 1-30 arası tek sayıları for döngüsü ile ekrana yazdırınız.

c) 3'ten başlayarak 41'e kadar olan sayıları 5'er arttırarak for döngüsü ile ekrana yazdırınız.

ç) 50'den 20'ye kadar olan sayıları 3'er azaltarak for döngüsü ile ekrana yazdırınız.

Range fonksiyonu ile else anahtar sözcüğü de kullanılarak döngü sonunda bir mesaj verilebilir. Kullanımı şu şekildedir:

```
for sayilar in range(10):
    print(sayilar)
else:
    print("Döngü bitti")
```

Ekran Çıktısı:

```
0
1
2
3
4
5
6
7
8
9
Döngü bitti
```

Önemli Not: Toplama işleminde, 0 etkisiz eleman olduğundan toplam değişkenine başlangıçta 0 (sıfır) atanır. Çarpma işleminde ise 1 etkisiz eleman olduğundan çarpım değişkenine başlangıçta 1 atanır.

Örnek 11 : for döngüsü ile 1'den 10'a kadar olan sayıların toplamını bularak ekrana yazdırınız.

http://kitap.eba.gov.tr/KodSor.php?KOD=22357

```
toplam=0
for sayilar in range(11):
    toplam=toplam+sayilar
print("Sayıların toplamı= ",toplam)
```

Ekran Çıktısı:

```
Sayıların toplamı=  55
```

Bu örnekte toplama 0 ilk değeri atandı. 1-10 arasındaki sayılar toplanacağı için range değeri 11 olarak verildi ve sayilar isimli bir değişken oluşturuldu. Döngü her döndüğünde sayilar değişkeni toplam değişkenine eklendi. Print fonksiyonu ile toplam ekrana yazdırıldı.

Önemli Not: Python programlama dilinin girinti yapısından bahsedilmişti. Yukarıdaki kodun son satırı bir önceki satırla aynı hizada yazılırsa for döngüsü içinde kabul edilir. Bu durumda toplam, döngü her başa döndüğünde adım adım ekrana yazılır. Lütfen deneyerek sonucunu görünüz.

Sıra Sizde:

a) Girilen iki sayı arasındaki sayıların toplamını bularak ekrana yazdırınız.

b) Girilen iki sayının arasındaki sayıların ortalamasını bularak ekrana yazdırınız.

c) Girilen sayının faktöriyelini bularak ekrana yazdırınız.

ç) Elemanları sırasıyla 4, 12, 18, 33 olan sayılar ile sayilar isminde bir liste oluşturunuz. Listenin elemanlarını for döngüsü kullanarak toplayınız ve ekrana yazdırınız.

5. 2. 1. 2. In Kullanımı:

in operatörü bir elemanın listede olup olmadığını kontrol eder. For döngüsü ile kullanımı şu şekildedir:

```
meyveler=["çilek","muz","şeftali"]
for meyve in meyveler:
    print(meyve)
```

Ekran Çıktısı:

```
çilek
muz
şeftali
```

Bu örnekte meyveler listesi içinde meyve isimli bir değişken oluşturularak ekrana yazdırılmıştır.

in operatörü metinsel (string) ifadeleri de harf harf ekrana yazdırabilir. Kullanımı şu şekildedir:

```
for harfler in "Döngü":
    print(harfler)
```

Ekran Çıktısı:

D

ö

n

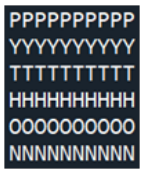
g

ü

Sıra Sizde:

a) Yukarıdaki örnekte ikinci satırı print(harfler*10) şeklinde değiştirerek çalıştırınız.

b) Çıktısı aşağıdaki gibi olan kodu yazınız.



Örnek 12 : 10-20 arası sayılardan oluşan sayilar isimli bir liste oluşturarak liste içinde 3'e tam bölünen sayıları ekrana yazdırınız.

```
sayilar=[10,11,12,13,14,15,16,17,18,19,20]
for sayi in sayilar:
  if sayi%3==0:
    print(sayi)
```

Ekran Çıktısı:

12

15

18

Sıra Sizde:

a) Yukarıdaki listede bulunan çift sayıları ekrana yazdırınız.

b) Yukarıdaki liste ile sayilar2=[21,22,23,24,25] listesini birleştirerek 4'e tam bölünen sayıları ekrana yazdırınız.

Örnek 13 : alan_adi isimli, değeri bilişim olan bir değişken tanımlayarak içinde kaç adet "i" harfi olduğunu bulup ekrana yazdırınız.

```
alan_adi="bilişim"
toplam=0
for aranan in alan_adi:
  if aranan=="i":
    toplam=toplam+1
print("Bu metinde toplam ",toplam," adet i vardır")
```

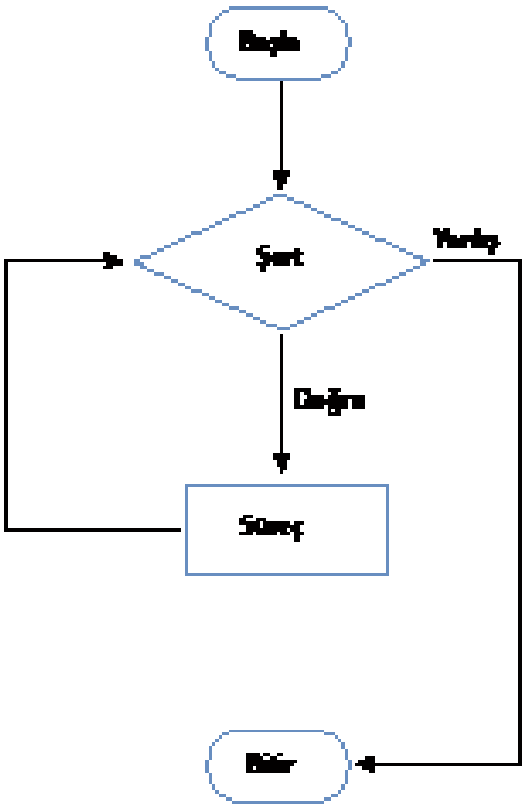
Ekran Çıktısı:

Bu metinde toplam 3 adet "i" vardır.

Sıra Sizde: Kullanıcıdan bir ifade ve aranacak harf girmesini isteyiniz. Girilen ifadede kaç tane "a" harfi olduğunu bularak ekrana yazdırınız.

5.2.2. While Döngüsü

While döngüsü hemen hemen tüm programlama dillerinde bulunmaktadır. Test edilen ifade doğru (true) olduğu sürece kodları tekrarlamaktadır. While yapısı genellikle kod bloğunun kaç kez tekrar edileceğinin bilinmediği durumlarda kullanılmaktadır. While döngüsünün yapısı aşağıdaki gibidir:



Şekil 5.2: While döngüsünün yapısı

Örnek 14 :

```
i=0
while (i<5):
  print("Kodlama")
  i=i+1
```

Ekran Çıktısı:

```
Kodlama
Kodlama
Kodlama
Kodlama
Kodlama
```

Bu örnekte i değişkenine ilk değer olarak 0 (sıfır) atanmıştır. while (i<5): satırı ile i değeri 5 olana kadar (5 dâhil değil) döngü devam eder. Her adımda bir kez "Kodlama" ifadesi ekrana yazdırılır ve i değeri 1 arttırılır. i değeri sırasıyla 0, 1, 2, 3 ve 4 olur. Yani döngü 5 kez döner ve program sonlanır.

Sıra Sizde: Yukarıdaki kodda while satırındaki şartı i<=5 yaparak çalıştırınız ve çıktılar arasındaki farkı sınıfta tartışınız.

Örnek 15 :

```
i=0
while (i<=20):
  print(i)
  i=i+2
print("Döngü sonu")
```

Ekran Çıktısı:

```
0
2
4
6
8
10
12
14
16
18
20
Döngü sonu
```

Bu örnekte i değeri 0'dan başlayarak 20'ye kadar 2'şer artarak (i=i+2) ekrana yazdırılmıştır.

Sonsuz döngü: Programlama dillerinde döngü oluştururken yapılacak bir mantık hatası sonsuz döngüye neden olabilir. Sonsuz döngüde program sürekli çalışacaktır. Sonsuz döngüden çıkmak için Crtl+C tuş kombinasyonu kullanılabilir.

Örnek 16 :

```
i=15
while (i<20):
  print(i)
  i=i-1
```

Bu örnekte i değişkeni 15'ten başlayarak 1'er azalır (i=i-1). while (i<20) şartına göre i sürekli azalarak devam ettiği için sonsuza doğru gitmektedir ve bu döngü hiçbir zaman bitmeyecektir.

Örnek 17 :

```
while True:
  print("Sonsuz döngüye girildi")
```

Bu örnekte de while True satırı aksi belirtilmediği sürece devam edilmesi anlamı taşır ve bu nedenle kullanıcı döngüden çıkana kadar print satırında bulunan ifadeyi yazar.

Sıra Sizde:

a) 1-30 (30 dâhil) arasındaki tek sayıları while döngüsü ile ekrana yazdırınız.

b) 60-30 (30 dâhil değil) arasındaki çift sayıları azalan sırada while döngüsü ile ekrana yazdırınız.

c) 0-100 (100 dâhil) arasındaki sayılardan 5'e tam bölünenleri while döngüsü ile ekrana yazdırınız.

ç) Ekran çıktısı aşağıdaki gibi olan kodu while döngüsü ile yazınız.

1 . sınıf

2 . sınıf

3 . sınıf

4 . sınıf

5 . sınıf

6 . sınıf

7 . sınıf

8 . sınıf

9 . sınıf

10 . sınıf

11 . sınıf

12 . sınıf

Örnek 18 :

http://kitap.eba.gov.tr/KodSor.php?KOD=22359

```
i=1
sonuc=1
faktoriyel=int(input("Faktoriyeli hesaplanacak sayıyı giriniz: "))
while (i<=faktoriyel):
  sonuc=i*sonuc
  i=i+1
print("Sonuc=",sonuc)
```

Ekran Çıktısı:

```
Faktoriyeli hesaplanacak sayıyı giriniz: 5
Sonuc= 120
```

Bu örnekte sonuç değişkenine ilk değer olarak 1 atanmıştır. Faktöriyel hesaplama bir çarpma işlemi olduğundan, çarpmadaki etkisiz eleman olan 1 verilmiştir. Faktöriyel hesaplama 1 sayısından başlayacağı için i değişkenine de ilk değer olarak 1 atandı. Faktöriyel kendinden önceki sayıların çarpımı olduğundan sonuc=i*sonuç ifadesi eklendi.

Sıra Sizde:

a) 1 ile 20 arasındaki (20 dâhil) sayıların toplamını bulan programı while döngüsü ile yazınız.

b) Girilen iki sayı arasındaki sayıları toplayan programı while döngüsü ile yazınız.

c) Girilen iki sayı arasındaki sayıların ortalamasını bulan programı while döngüsü ile yazınız.

ç) 20 ile 50 arasındaki (50 dâhil) çift sayıların toplamını bulan programı while döngüsü ile yazınız.

Örnek 19 : Girilen sayı 0 (sıfır) olana kadar girilen tüm sayıları toplayan ve ekranda gösteren programı yazınız.

```
toplam=0
sayi=1
while (sayi!=0):
  sayi=int(input("Bir sayı giriniz: "))
  toplam=toplam+sayi
print("Sonuc=",toplam)
```

Ekran Çıktısı:

```
Bir sayı giriniz: 5
Bir sayı giriniz: 6
Bir sayı giriniz: 7
Bir sayı giriniz: 0
Sonuc= 18
```

Bu örnekte toplam değişkenine ilk değer olarak 0 atandı. *sayi* değişkenine de 1 değeri verilmiştir (0 dışında herhangi bir değer verilebilir). sayi!=0 ifadesi ile sayı 0 olmadığı sürece while bloğunun çalışması sağlandı. Girilen her sayı toplama eklendi ve 0 girildiğinde döngüden çıkılarak sonuç ekrana yazdırıldı.

Sıra Sizde:

a) Klavyeden 1 girilene kadar girilen sayıların ortalamasını alan kodu yazınız.

b) Girilen şifre "Python" olana kadar "Tekrar deneyiniz" uyarısı veren, "Python" girildiğinde "Giriş başarılı" uyarısı veren kodu yazınız.

Programlama dillerinde döngüler iç içe de kullanılabilir. Örneğin 3*3'lük matris ve adreslerini oluşturmak için iç içe döngü yapısı kullanılır.

	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

Örnek 20 :

```
for i in range(0,3):
  for j in range(0,3):
    print([i,j])
```

Ekran Çıktısı:

```
[0, 0]
[0, 1]
[0, 2]
[1, 0]
[1, 1]
[1, 2]
[2, 0]
[2, 1]
[2, 2]
```

Bu örnekte 0 ile 3 aralığında iki farklı döngü oluşturularak iç içe kullanılmıştır. Birinci for döngüsündeki i değeri 0 olduğunda; içindeki döngüde j değeri sırayla 0,1 ve 2 olmaktadır.

Sıra Sizde: Çıktısı aşağıdaki gibi devam eden çarpım tablosunu iç içe döngü kurarak kodlayınız.

```
------------------
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
------------------
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
...
```

5.2.3. Break ve Continue Deyimleri

Break komutu döngüleri sonlandırır. Programlamada öngörülemeyen bir durum gerçekleştiğinde break komutu döngüden çıkılmasına imkân tanır. Döngüden çıkıldıktan sonra ise döngü sonrasındaki ilk satırdan kod çalıştırılmaya devam eder.

Örnek 21 :

```
i=1
while True:
  if (i==5):
    print("Döngüden çıkıldı")
    break
  print(i)
  i=i+1
```

Ekran Çıktısı:

```
1
2
3
4
Döngüden çıkıldı
```

Örnekte while True: ifadesi görülmektedir. Aksi belirtilmediği sürece döngünün devam etmesi anlamına gelen bu ifade sonsuz döngü anlatılırken de örnek olarak verilmişti. Burada kullanılan break ifadesi sayesinde while True: satırı ile sürekli devam etmesi istenen döngüden çıkılmıştır. Görüldüğü üzere i değişkeni 1 değerini alır. i değişkeni 5 değerine eşit olduğunda ise döngüden çıkılır. break ifadesi print(i) satırından önce olduğu için 5 değeri ekran çıktısında görülmez.

Sıra Sizde:

a) Kullanıcıdan 1 ile 5 arasında bir sayı girmesini isteyiniz. Kullanıcı 3 sayısını girdiğinde break komutu ile döngüden çıkılarak "3 sayısı girildi ve döngü sona erdi" çıktısı veren kodu yazınız.

b) Kullanıcıdan 8 karakterlik bir şifre girmesini isteyiniz. Kullanıcı 8'den az ya da daha fazla karakter içeren bir şifre girdiğinde "Şifreniz 8 karakter olmalıdır." şeklinde uyarı verdiriniz. Kullanıcı şartlara uygun bir şifre girdiğinde de "Şifreniz kaydedildi." uyarısı verdiriniz.

Örnek 22 :

http://kitap.eba.gov.tr/KodSor.php?KOD=22360

```
sayi = int(input("Bir sayı girin: "))
for i in range(1, 10):
  if i == sayi:
    break
  print(i)
print("Döngü sona erdi")
```

Ekran Çıktısı:

```
Bir sayı girin: 8
1
2
```

3

4

5

6

7

Döngü sona erdi

Bu örnekte kullanıcı tarafından bir sayı girildi. For komutu kullanılarak 1 ile 10 aralığında bir döngü oluşturuldu. Kullanıcı 1-10 arası bir sayı girerse döngü kullanıcının girdiği sayıya kadar devam eder. Örneğin 8 sayısı girildiğinde döngü 8'e kadar çalıştıktan sonra durdurulur. 8 dâhil olmadığından ekranda en son 7 sayısı görülür.

Sıra Sizde: Yukarıdaki kodu çalıştırarak 1-10 aralığı dışında bir sayı giriniz ve ekran çıktısını sınıfta tartışınız.

Örnek 23 :

```
metin = "Ankara"
for i in metin:
  if i == 'r':
    break
  print(i)
```

Ekran Çıktısı:

A

n

k

a

Bu örnekte "Ankara" metni for döngüsü ile ekran çıktısı olacakken r harfinde break ile döngü sonlandırılmıştır.

Örnek 24 :

```
sayilar=[10, 11, 12, 13, 14, 15, 16]
for aranan in sayilar:
  print(aranan)
  if(aranan==14):
    print("14 sayısı bulundu")
    break
```

Ekran Çıktısı:

10

11

12

13

14

14 sayısı bulundu

Bu örnekte, oluşturulan dizide bulunan 14 sayısı bulunduğunda döngüden çıkılmıştır.

Sıra Sizde: Elemanları alfabedeki ilk 8 harf olan bir liste oluşturarak "e" harfine gelindiğinde döngüden çıkan kodu yazınız.

Örnek 25 :

```
import random
while True:
  n = random.randint(1, 20)
  print("Rastgele seçilen ", n)
  if n % 2 == 0:
    print("Çift sayı seçildi, döngü bitti")
    break
```

Ekran Çıktısı:

```
Rastgele seçilen  5
Rastgele seçilen  13
Rastgele seçilen  15
Rastgele seçilen  8
Çift sayı seçildi, döngü bitti.
```

Bu örnekte kullanılan random komutu ile 1-20 arasında rastgele bir sayının hafızaya alınması sağlanmıştır. Seçilen bu rastgele sayı çift bir sayı olana kadar döngü devam eder. Çift sayı tutulduğunda ise döngü sona erer.

Sıra Sizde:

a) 1 ile 100 arasında rastgele 6 sayı seçerek ekrana yazdırınız.

b) Random metodu ile 0-20 arası bir sayı seçerek kullanıcının bu sayıyı tahmin etmesini isteyiniz. Kullanıcının tahminine göre arttır ve azalt şeklinde uyarılar verdirerek doğru sonuca ulaşılmasını sağlayınız.

Continue komutu döngüyü başa döndürerek continue sonrasında yazılan kod bloğunun göz ardı edilmesini sağlar. Başka bir ifadeyle döngünün o anki adımını atlayarak kaldığı yerden devam eder.

Örnek 26 :

```
i = 0
while i < 10:
 i=i+1
 if i == 5:
  continue
 print(i)
```

Ekran Çıktısı:

```
1
2
3
4
6
7
```

8

9

10

Bu örnekte i değişkenin değeri 5 olduğunda continue ifadesi ile döngü başa dönerek 5 değerini yazmadan devam etmiştir.

Sıra Sizde: Aşağıdaki kod nasıl bir çıktı üretir? Yazınız.

```
i = 0
while i < 50:
 i=i+1
 if i>10 and i<45:
  continue
 print(i)
```

Örnek 27 :

```
while True:
   sifre = input("Bir şifre giriniz: ")
   if len(sifre) < 4 or len(sifre) > 4:
      print("4 karakterden oluşan bir şifre girmelisiniz.")
      continue
   else:
      print("Şifreniz oluşturuldu: ", sifre)
      break;
print("Şifrenizi while döngüsü içinde oluşturdunuz. ")
```

Ekran Çıktısı:

```
Bir şifre giriniz: 45
4 karakterden oluşan bir şifre girmelisiniz.
Bir şifre giriniz: 459876
4 karakterden oluşan bir şifre girmelisiniz.
Bir şifre giriniz: 4554
Şifreniz oluşturuldu:  4554
Şifrenizi while döngüsü içinde oluşturdunuz.
```

Bu örnekte kullanıcıdan bir şifre girmesi istendi. Girilen şifre 4 karakterden az ya da fazla olduğu sürece döngü, continue komutu ile devam etmektedir. 4 karakter şifre girildiğinde ise "Şifre oluşturuldu" mesajı verilerek break komutu ile döngüden çıkılmaktadır.

Önemli Not: Telefon numarası, T.C. kimlik numarası gibi sayısal ifadeler üzerinde matematiksel işlemler yapılmayacağı için string olarak tanımlanabilir.

Örnek 28 :

http://kitap.eba.gov.tr/KodSor.php?KOD=22362

```
sayilar=[20,23,79,88,111,65]
for sayi in sayilar:
    if sayi%2 == 0:
        continue
    print(sayi)
```

Ekran Çıktısı:

```
23
79
111
65
```

Bu örnekte listedeki sayılardan çift olanlar continue kullanıldığı için atlanmış ve ekran çıktısında listelenmemiştir.

Sıra Sizde: 1-30 arasındaki sayıları bir liste hâline getirerek continue ile sadece tek olanları ekrana yazdırınız.

ÖLÇME VE DEĞERLENDİRME 5

A) Aşağıdaki boşluk doldurma sorularını cevaplayınız.

```
if sayi%3==0:
  print("A")
else:
  print("B")
```

1. Yukarıdaki kod çalıştırıldığında ve sayi değişkeni 7 olarak girildiğinde …… çıktısı üretilir.
2. if ((3>5 and 5<1) or 5==5) satırı …… değerini döndürür.
3. Döngüleri sonlandırmak için ……. komutu kullanılır.
4. Rastgele sayı seçmek için ……. komutu kullanılır.
5. ……., döngünün başlangıç ve bitiş değeri belli olan durumlarda kullanılan, varsayılan olarak 0'dan (sıfır) başlayarak birer birer artan fonksiyondur.

B) Aşağıdaki çoktan seçmeli soruları cevaplayınız.

```
if yas>18 and ehliyet=="var":
  print("Başarılı")
```

6. Yukarıdaki kod bloğu ne anlama gelmektedir?

   A) yas değişkeni 18 ve ehliyet değişkenin değeri var ise ekrana Başarılı yazar.
   B) yas değişkeni 18'den küçük ve ehliyet değişkenin değeri var ise ekrana Başarılı yazar.
   C) yas değişkeni 18'den büyük veya ehliyet değişkenin değeri var ise ekrana Başarılı yazar.
   D) yas değişkeni 18'den büyük ve ehliyet değişkenin değeri var ise ekrana Başarılı yazar.
   E) yas değişkeni 18'den küçük veya ehliyet değişkenin değeri var ise ekrana Başarılı yazar.

```
for sayilar in range(15):
  print(sayilar)
```

7. Yukarıdaki kod çalıştırıldığında üretilecek çıktının son değeri hangisidir?

   A) 15 B) 14 C) 10 D) 1 E) 0

```
i=3
while (i<7):
  print("MEB")
  i=i+1
```

8. Yukarıdaki kod çalıştırıldığında ekrana kaç kez MEB yazar?

   A) 2 B) 3 C) 4 D) 6 E) 7

```
for sayilar in range(10,2,-3):
    print(sayilar)
```

9. Yukarıdaki kod çalıştırıldığında sırasıyla hangi çıktıları üretir?

A) 10 7 4

B) 10 2 3

C) 2 5 8

D) 10 8 6

E) 2 4 6

```
toplam=0
for sayilar in range(5):
    toplam=toplam+sayilar
print(toplam)
```

10. Yukarıdaki kod çalıştırıldığında nasıl bir çıktı üretir?

A) 5 B) 15 C) 6 D) 8 E) 10

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.

ÖĞRENME BİRİMİ 6

FONKSİYONLAR

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Fonksiyon mantığını kavrayacak,
- Gömülü fonksiyonları programlarınızda kullanabilecek,
- Kendi fonksiyonlarınızı yazabilecek, parametre gönderebilecek ve kullanabileceksiniz.

Anahtar Kelimeler:

Fonksiyon, parametre, veri döndürme, özyineleme.

Hazırlık Çalışmaları

1. İnternet üzerinden "Nesne Yönelimli Programlama" konusunu araştırınız.
2. Fonksiyonel programlama ve işlevsel programlama konularını araştırıp aralarındaki temel farkı yazınız.

6. FONKSİYONLAR

6.1. Fonksiyon

Bir bilgisayar programı yazılırken bazı işlemlerin programın farklı yerlerinde sürekli tekrarlanması gerekebilir. Örneğin arazi hesapları ile ilgili bir program yazılıyorsa sık sık geometrik şekillerin alanı hesaplanmak zorunda kalınabilir. Her gerektiğinde alan hesabı işlemini yerine getiren kodları yazmak hem programcının iş yükünü hem de hata yapma olasılığını arttırır. Bu nedenle programcılar, sık tekrarlanan işler için aynı kodu defalarca yazmak yerine, işlemi yerine getiren kod bloğunu yazıp adlandırarak ihtiyaç hâlinde bu adla basit bir şekilde çağırıp kullanmayı tercih ederler.

İhtiyaç duyulduğunda çağrılıp çalıştırılabilen bu kod paketlerine fonksiyon adı verilir. Fonksiyon farklı programlama dillerinde prosedür veya yordam olarak da adlandırılabilir.

Sık tekrarlanan işlemleri gerektiğinde kullanılacak küçük kod parçalarına bölüp yazma yani "fonksiyon" yaklaşımı bütün programlama dillerinde kullanılabilecek bir yöntemdir.

Fonksiyonlar sayesinde;

Programcı aynı kodları defalarca yazma yükünden kurtulur.
Daha az kod yazılacağı için hata yapma olasılığı azalır.
Fonksiyon sadece çağrıldığında kullanılacağı için bilgisayarın bellek kullanımından tasarruf edilir.
Kod okunabilirliğini arttırır ve kod analizini daha kolay hâle getirir.
Karmaşık problemlerin daha basit küçük parçalara ayrılarak çözülmesini kolaylaştırır.

Fonksiyonlar çağrıldıklarında, barındırdıkları kod kümelerini işleyerek oluşan sonuçları döndürebilir. Ayrıca istenirse kendilerine parametre olarak gönderilen verileri işleyip ürettikleri sonucu da döndürebilir.

6.1.1. Fonksiyonların Kullanımı

Bu bölüme kadar yazılan örnek programlarda programlama dilinin bazı hazır fonksiyonları kullanıldı. Örneğin ekrana veri yazdırmak için kullandığınız print() bir fonksiyondur. Programlama dilleri yazılımcının gerektiğinde kullanabileceği birçok hazır fonksiyonla beraber gelir. Bunlara built-in (gömülü fonksiyonlar) denir.

Bir fonksiyonu çağırıp çalıştırmak için fonksiyona verilen ismi yazmak gerekir. Eğer fonksiyon parametre alıyorsa isminin yanına parantez içinde fonksiyona gönderilecek parametreleri de yazmak gerekir.

Örnek 1:

```
print("Merhaba, ben bir gömülü fonksiyonum!")
```

Yukarıdaki komut çalıştırıldığında programlama dili ile beraber gelen print isimli fonksiyon, ekrana getireceği metin fonksiyona parantez içinde parametre olarak gönderilerek çağrılmış olur.

Bu komut çalıştığında

```
Merhaba, ben bir gömülü fonksiyonum!
```

çıktısını verir ve girilen parametre ekrana metin olarak gelir.

Tanımladığımız veya programlama dili ile hazır gelen fonksiyonlar çağrılmadıkça o fonksiyon bloğu içinde yer alan kodlar çalıştırılmaz. Fonksiyonu başka bir fonksiyondan ya da doğrudan programdan ismi ile birlikte parantez içinde parametre bilgilerini yazarak çağırabilirsiniz.

6.1.2. Gömülü Fonksiyonların ve Modüllerin Kullanımı

Programlama dili ile temel işlemleri yerine getiren birçok fonksiyon hazır ve tanımlanmış olarak gelir. Şu ana kadar kullanılmış olan print, input, type, int, float, str gibi fonksiyonlar programlama dili içinde gömülüdür. Gömülü fonksiyonlar, geliştiricileri tarafından programlama dili içine gömülmüş ve tanımlamaya gerek kalmadan kullanılabilen fonksiyonlardır. Gömülü fonksiyonlarda tek yapılması gereken fonksiyonu çağırmak ve kullanmaktır.

Bu gömülü fonksiyonlar haricinde farklı işlevler için geliştirilmiş fonksiyon kütüphaneleri vardır. Örneğin matematik işlemlerinde ihtiyaç duyabileceğiniz tüm fonksiyonlar, hazır olarak programlama dili ve "Math" isimli bir kütüphane ile gelir. İhtiyaç duyduğunuzda makine öğrenmesi, oyun geliştirme, ağ işlemleri gibi alanlarda size gerekli işlevleri sağlayacak kütüphaneler programlama diline eklenip kullanılabilir.

Kendiniz de projenizde kullanmak için yazdığınız fonksiyonları bir kütüphane hâlinde toplayarak ihtiyacı olan programcılara dağıtabilirsiniz. Bu şekilde bir konuda belirli işlevleri yerine getiren fonksiyonların bir araya getirildiği Python dosyalarına modül denir. Hâlihazırda programlama dili kurulumu ile birlikte birçok modül bilgisayarınıza yüklenir. Bu modüller haricinde ihtiyaç duyabileceğiniz modüller de üçüncü parti sağlayıcılardan bulunabilir.

Programlama diline eklenmiş olan modülü ve içerdiği fonksiyonları kullanabilmek için önce yazılan kodun başına "import" komutu eklenerek modüle erişim sağlanır. Programlama dili kurulumu ile gelen, matematik fonksiyonlarını içeren "math.py" dosyasına yani Math modülüne erişmek için programın başına aşağıdaki gibi erişim ifadesi eklenmesi gerekir.

from modül_adi import fonksiyon_adi

Programın başlangıç kısmına bu ifade eklenerek Math modülünden istenilen fonksiyonlara erişilebilir.

Programa,

```
from math import sin
```

satırı eklendiğinde Math modülünden parametre olarak verilen sayının sinüs değerini veren fonksiyona erişim sağlanmış olur ve sin() tanımlı fonksiyon programda istediğiniz yerde kullanılabilir. Birden fazla fonksiyona erişim sağlamak isteniyorsa fonksiyonları aşağıdaki gibi beraber belirtmek yeterli olacaktır.

```
from math import sin, sqrt, cos, pow
```

Bu satırla programa modülden karekök bulan sgrt(), güç hesabı yapan pow() ve trigonometri işlemi yapan sin() ve cos() fonksiyonlarına erişim imkânı verilmiş olur. Bu fonksiyonlar aşağıdaki örnekle denenebilir.

Örnek 2:

```
from math import sin, sqrt, cos, pow # fonksiyonlara erişim sağlıyoruz.
print( sqrt(4) ) # 4 sayısının karekökünü buldurup ekrana yazdırıyoruz.
print( sin(30) ) # 30 sayısının sinüs değeri
print( cos(45) ) # 45 sayısının cosinüs değeri
print( pow(3,2) ) # 3'ün 2. kuvveti
```

Fonksiyonların yanında parantez içinde yer alan ifadeler parametredir. sqrt(), sin() ve cos() fonksiyonlarının tek parametre, pow() fonksiyonununsa iki parametre aldığına dikkat ediniz. Fonksiyonlar birden fazla parametre alabilir. Böyle durumlarda parametreler arasına virgül (,) konulur.

Çıktı:

```
2.0
0.9974949866040544
0.23523757330298942
9.0
```

Eğer aynı modülden çok sayıda fonksiyon kullanılması gerekiyorsa fonksiyon isimlerini spesifik olarak yazmak yerine programa import modül_adı satırı eklenerek modüldeki tüm fonksiyonlara erişim sağlanabilir.

```
import math
```

Aşağıdaki örnek programda Math modülündeki bütün fonksiyonlar erişime açılmış, bazıları kullanılmıştır.

Örnek 3:

```
import math
print( math.pow(3,12) )
print( math.sqrt(9) )
print( math.sin(math.pi/2) )
```

Çıktı:

```
531441.0
3.0
1.0
```

Fonksiyonlar isimleri ile erişime açılmadığı için programdan çağrılırken isimlerinin başlarına modül adları da eklendi. Bu sayede program hangi modülden hangi fonksiyona erişilmek istendiğini anlayarak fonksiyonun işlevini yerine getirmiştir.

6.2. Fonksiyon Tanımlama

Fonksiyonlar def komutu kullanılarak tanımlanabilir. Fonksiyon tanımlarken izlenecek yol aşağıdaki gibidir:

1. def komutu yazılarak yeni bir fonksiyon tanımlanacağı programlama diline bildirilir.
2. Anahtar sözcükten sonra fonksiyon çağrılırken kullanılacak olan isim Python'un isimlendirme kurallarına uygun olarak belirlenmelidir. Burada fonksiyonun işlevi ile ilintili bir isim vermek kod okunabilirliği açısından mantıklı olacaktır.
3. Parantezler arasına fonksiyona gönderilecek parametreler yazılır, eğer fonksiyonumuz parametre almıyorsa parantez araları boş bırakılır. Tanım sonuna iki nokta üst üste konarak alt satırdan itibaren kod bloğunun başladığı belirtilir.
4. Tanım ve isim satırının altında bir sekme (tab) boşluk bırakılarak fonksiyon çağrıldığında çalışacak kodlar yazılır.

```
def fonksiyon_adi  ( varsa parametre listesi ) :
    Kod_blogu
    Kod_blogu
```

Bu sıralamayı izleyerek çağrıldığında ekrana "Merhaba Arkadaşlar" yazan selamla isimli bir fonksiyon örneği aşağıdadır:

Örnek 4:

```
def selamla():
    print("Merhaba Arkadaşlar!")
```

Yukarıdaki kod bloğu çalıştırıldığında ekrana hiçbir şey gelmeyecektir. Çünkü fonksiyon tanımlanmasına rağmen henüz çağrılmadı. Tanımlanan fonksiyonlar sadece çağrıldıklarında çalışır. Programa bir satır daha ekleyerek yazılan fonksiyon çağrılabilir.

Örnek 5:

```
def selamla():
    print("Merhaba Arkadaşlar!")


selamla() #fonksiyonu çağırıyoruz.
```

Program çalıştırıldığında fonksiyonun çağrılıp yürütüldüğü ve ekrana "Merhaba Arkadaşlar!" yazısının geldiği görülür. İstenirse aynı fonksiyon birden fazla çağrılabilir.

Örnek 6:

```
def selamla():
    print("Merhaba Arkadaşlar!")


selamla()#fonksiyonu çağırıyoruz.
selamla()#fonksiyonu 2. defa çağırıyoruz.
selamla()#fonksiyonu 3. defa çağırıyoruz.
selamla()#fonksiyonu 4. defa çağırıyoruz.
```

Kod bloğunun ekran çıktısı aşağıdaki gibi olacaktır. Selamla() fonksiyonu her çağrıldığında çalışacak ve ekrana içeriğindeki kod bloğuna uygun davranarak "Merhaba Arkadaşlar!" metnini getirecektir.

Çıktı:

```
Merhaba Arkadaşlar!
Merhaba Arkadaşlar!
Merhaba Arkadaşlar!
Merhaba Arkadaşlar!
```

Bir programda fonksiyon kullanılmadan önce tanımlanmalıdır,
aksi hâlde programınız hata verecektir.

Önceki selamla() fonksiyonu bir isim verisi isteyecek şekilde düzenlendiğinde aşağıdaki kod örneği elde edilir.

Örnek 7:

```
def selamla():
  ad = str(input("Adınızı giriniz: "))
  if ad:
    print ("Merhaba " + str(ad))
  else:
    print("Merhaba Arkadaşlar!")



selamla()
```

Çıktı:

```
Adınızı giriniz: Çağan
Merhaba Çağan
Adınızı giriniz:
Merhaba Arkadaşlar!
```

Daha önceki bölümlerde döngüler konusu işlenmişti. Aşağıdaki örnekte while döngüsü bir fonksiyon içinde kullanılarak ekrana 1'den 10'a kadar sayılar yazdırıldı.

Örnek 8:

```
def onakadar_say():
    a = 0
    while a<10:
        a += 1
        print(a)


onakadar_say()#fonksiyonu çağırıyoruz.
```

Program çalıştırılıp fonksiyon çağrıldığında fonksiyon içindeki kodlar çalışacak ve ekrana 1'den 10'a kadar sayılar yazdırılacaktır. Örnekte gördüğünüz gibi Python'un sekmeli akış yapısı fonksiyon içinde de devam etmektedir.

Aşağıda aynı örneğin for döngüsü ile gerçekleştirilmiş hâli gösterilmektedir.

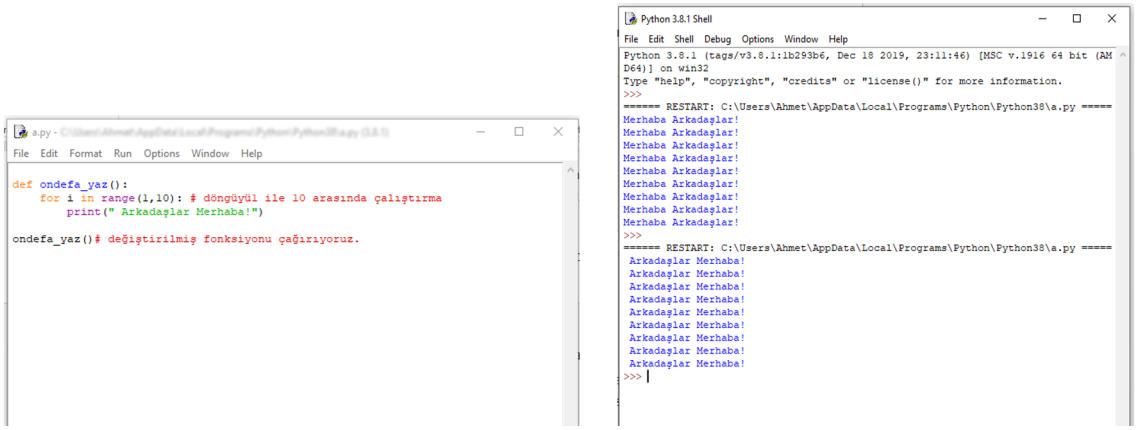
Örnek 9:

```
def ondefa_yaz():
    for i in range(1,10): # döngüyü1 ile 10 arasında çalıştırma
        print("Merhaba Arkadaşlar!")


ondefa_yaz()#fonksiyonu çağırıyoruz.
```

Kod derlendiğinde ekranda verilen metnin 10 defa yazdırıldığı görülür. Eğer ondefa_yaz() fonksiyonu bir daha çağrılırsa ekrana bir 10 satır daha eklenecektir.

6.2.1. Fonksiyon Düzenleme

Tanımlanan fonksiyonların işlevlerini değiştirmek için fonksiyon tanımı içindeki kod bloklarını yeniden düzenlemek yeterlidir. Örneğin yukarıdaki programda ekrana getirilen mesaj değiştirilmek istenirse içerikteki kodların düzenlenip yeniden çağrılması yeterli olacaktır.



Görsel 6.1: Fonksiyon düzenleme

Ekrana ilk olarak "Yıldız üçgen çiziliyor :" mesajını getiren, ardından her satıra bir döngü ile "*" işareti koyarak üçgen çizen fonksiyon örneği aşağıda gösterilmiştir.

Örnek 10:

```
def yildiz_ucgen_ciz():
    print("Yıldız üçgen çiziliyor:")
    for satir in range(1,10):
        print ("*" * satir)

yildiz_ucgen_ciz()# ücgen cizme fonksiyonu çağırılıyor.
```

Çıktı:

```
Yıldız üçgen çiziliyor:
*
**
***
****
*****
******
*******
********
*********
```

Sıra Sizde: Yukarıdaki örneği, aşağıdaki gibi bir ekran çıktısı verecek şekilde düzenleyiniz ve çalıştırınız.

```
Yıldız üçgen çiziliyor:
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
```

6.2.2. Parametre Kavramı ve Fonksiyonlar ile Parametre Kullanımı

Şu ana kadar fonksiyon tanımlarken parantez içleri hep boş bırakıldı. Bu şekilde parametre kullanmayan sadece içerdiği kod bloğunu işleyen fonksiyonlar yazıldı. Fonksiyonların en önemli işlevlerinden biri de verileri parametre olarak alıp işledikten sonra size sonucu bildirebilme yetenekleridir.

Parametre, yazılan fonksiyona işlemesi için gönderilen veridir. Metin, sayı, liste ve benzeri veriler fonksiyonlara işlenmeleri için parametre (bazen referans da denir) olarak gönderilebilir. Bunun için fonksiyon tanımlanırken parantez içine gönderilecek parametrenin hangi adla işleneceğinin belirtilmesi yeterlidir.

İlk yazılan selamla() isimli fonksiyon incelendiğinde parantez içine "ad" yazarak fonksiyona "ad" isimli bir parametre gönderileceği belirtilsin. Fonksiyon kodlarının da bu parametreyi ekrana yazdıracak şekilde düzenlenmesi gereklidir.

Örnek 11:

```
Merhaba Arda
Merhaba Ali
Merhaba Çağan
```

Aldığı parametre sayesinde fonksiyon daha işlevsel hâle gelmiştir. Fonksiyona her çalıştığında farklı parametre verilerek ekran çıktısı değiştirilmiş olur.

Çıktı:

```
def selamla(ad):
    print("Merhaba " + ad)


selamla("Arda")
selamla("Ali")
selamla("Çağan")
```

Eğer fonksiyon tanımlanırken parametre kullanacağı belirtilmiş ve parantez içine parametre tanımı yapılmışsa fonksiyon çağrılırken muhakkak parametre kullanılmalıdır. Aksi hâlde program çalışırken hata verecektir.

Programda fonksiyon ikinci defa çağrılırken parametre girilmesin.

```
def selamla(ad):
    print("Merhaba " + ad)

selamla("Arda")
selamla()
selamla("Çağan")
```

Program hatalı satıra kadar çalışacak ve hatalı satıra geldiğinde aşağıdaki gibi uyarı verecektir.

```
Merhaba Arda
Traceback (most recent call last):
  File "C:\Users\Ahmet\AppData\Local\Programs\Python\Python38\b6o1.py", line 5, in
<module>
    selamla()
TypeError: selamla() missing 1 required positional argument: 'ad'
```

Fonksiyon yazılırken alacağı parametrenin boş geçilmesi durumunda, parametre olarak varsayılan bir değer alacak şekilde düzenlenebilir.

Örnek 12:

```
def ulke_yaz(ulke = "Türkiye"): # varsayılan ülke değeri "Türkiye"
  print( ulke + " benim memleketim.")

ulke_yaz ("Türkiye")
ulke_yaz ("Azerbaycan")
ulke_yaz ()
ulke_yaz ("Almanya")
```

Çıktı:

```
Türkiye benim memleketim.
Azerbaycan benim memleketim.
Türkiye benim memleketim.
Almanya benim memleketim.
```

Parametre adlandırmaları biliniyorsa fonksiyon çağrılırken doğrudan parametrelere atama yapılabilir.

Örnek 13:

```
def en_kucuk_cocuk(cocuk3, cocuk2, cocuk1):
  print("En Genç olan cocuk " + cocuk3)

en_kucuk_cocuk (cocuk1= "Seyhan", cocuk2= "Ahmet", cocuk3= "Mehmet")
```

Çıktı:

```
En Genç olan cocuk Mehmet
```

Fonksiyona parametre olarak nesne, dizi ya da koleksiyon da gönderilebilir.

Örnek 14:

```
def mevsim_yaz(mevsim_dizisi):
  for x in mevsim_dizisi:
    print(x)

mevsimler = ["İlkbahar", "Yaz", "Sonbahar", "Kış"]

mevsim_yaz(mevsimler)
```

Çıktı:

```
İlkbahar
Yaz
Sonbahar
Kış
```

Aşağıda daha önce yazılan yildiz_ucgen_ciz() fonksiyonunun üçgenin satır sayısını parametre olarak alacak şekilde düzenlenmiş hâli gösterilmektedir.

Örnek 15:

```
def yildiz_ucgen_ciz(satirSayisi):
    print( str(satirSayisi) + " satırlık yıldız üçgen çiziliyor:")
    for satir in range(1,satirSayisi+1):
        print ("*" * satir)

yildiz_ucgen_ciz(6)# ücgen cizme fonksiyonu 6 satır parametresi ile çağrılıyor.
```

Satır sayısı dışarıdan fonksiyona "satirSayisi" adı gönderildi. Dışarıdan alınan bu değer str() gömülü fonksiyonu ile metne çevrilip ekrana yazdırıldı. Aynı değer döngüde de kullanıldı.

Düzenlenen yeni fonksiyonun ekran çıktısı aşağıdaki şekilde olacaktır.

```
6 satırlık yıldız üçgen çiziliyor:
*
**
***
****
*****
******
```

İstenirse bir fonksiyona virgül ile ayırarak birden fazla parametre gönderilebilir. Aşağıda bir dörtgenin alanını kenar uzunluklarını parametre alarak hesaplayan fonksiyon gösterilmiştir.

Örnek 16:

```
def dortgen_alan(kenar1, kenar2):
    alan = kenar1 * kenar2
    print("Verilen dörtgenin alanı = "+ str(alan) + " metre karedir.")



dortgen_alan(3,7)# 3 metreye 7 metre bir dörtgenin alanı içi fonksiyonu çağıralım.
```

Çıktı:

```
Verilen dörtgenin alanı = 21 metre karedir.
```

Gönderilen sayının tek ya da çift sayı mı olduğunu bulan fonksiyonu içeren program aşağıda gösterilmiştir.

Örnek 17:

http://kitap.eba.gov.tr/KodSor.php?KOD=22363

```
# girilen sayı çift mi tek mi bulan fonksiyon
def cift mi_tek mi( sayi ):
    if (sayi % 2 == 0):
        print("çift")
    else:
        print("tek")
# fonksiyonu deniyoruz
Cift mi_tek mi(5)
Cift mi_tek mi(8)
```

Çıktı:

```
tek
çift
```

Sıra Sizde: Yıldızlarla üçgen oluşturan örnekten faydalanarak satır sayısını dışarıdan parametre olarak alan ve aşağıdaki gibi bir desen çıktısı verecek fonksiyonu yazınız ve çalıştırınız.

```
*        *
**      **
***    ***
****  ****
**********
****  ****
***    ***
**      **
*        *
```

6.2.3. Değer Döndürme ve Return İfadesi

Daha önce yazılan fonksiyonlar, içerdikleri kodları işleyerek çalıştırıp aldıkları parametreleri kullanarak bazı işlemler yaptı. Çalışırken dışarıya herhangi bir veri göndermedi. Bu şekilde çalışıp dışarıya veri döndürmeyen fonksiyonlara void fonksiyon denir.

Örnek 18:

```
def ortalama_hesapla(sinav1,sinav2,sozlu):
    ortalama = (sinav1+sinav2+sozlu)/3.0  # ortalama hesaplanması
    print(ortalama) # ortalamanin yazdirilmasi
```

Yukarıdaki fonksiyonda sınav ve sözlü notlarını parametre olarak alan fonksiyon not ortalamasını bulup ekrana yazdırır. Bu fonksiyonu daha dinamik hâle getirmek için bulduğu ortalama değerini ekrana yazdırmak yerine veri olarak döndürmesi sağlanabilir.

Veri döndürme işlevi için return ifadesi kullanılır. Bulunan ortalama ekrana yazdırılmak yerine return ifadesi ile fonksiyon sonucu oluşan veri olarak programa geri döndürülebilir.

Ortalama sonucunu geri döndüren fonksiyon aşağıdaki örnekte gösterilmiştir.

```
def ortalama_hesapla(sinav1,sinav2,sozlu):
    ortalama = (sinav1+sinav2+sozlu)/3.0  # ortalama hesaplanması
    return ortalama # ortalamanin geriye değer olarak döndürülmesi
```

Örnek 19:

```
def ortalama_hesapla(sinav1,sinav2,sozlu):
    ortalama = (sinav1+sinav2+sozlu)/3.0  # ortalama hesaplanması
    return ortalama # ortalamanin geriye değer olarak döndürülmesi


if (ortalama_hesapla(35,60,40) < 50):
    print("Dersten kaldı")
else:
    print("Geçti")
```

Örnekte değerlendirilmek istenen notlar (35, 60 ve 40) karar cümlesi içinden parametre olarak fonksiyona gönderilmiştir. Fonksiyon bulduğu sonucu (Örneğe göre 45 sonucunu bulacaktır.) geri döndürecektir. Kodlardaki karar cümlesi de gelen 45 değerine göre karşılaştırma operatörünü işletecek ve ekrana "Dersten kaldı." ifadesini yazacaktır.

Aşağıdaki örnekte verilen sayının tam bölenlerini bulup liste hâlinde gönderen bir fonksiyon gösterilmiştir.

Örnek 20:

```
def tam_bolenleri_bul(sayi):
    tam_bolenler = []

    for i in range(2, sayi):
        if (sayi % i == 0):
            tam_bolenler.append(i)

    return tam_bolenler

print(tam_bolenleri_bul(15))
```

Çıktı:

```
[3, 5]
```

UYGULAMA FAALİYETİ 1

Öğrenilen fonksiyon ve değer döndürme kavramlarını kullanarak basit bir hesap makinesi programı yazınız.

```
def toplama(sayi1,sayi2):
    return sayi1+sayi2

 def cikarma(sayi1,sayi2):
    return sayi1-sayi2

 def carpma(sayi1,sayi2):
    return sayi1*sayi2

 def bolme(sayi1,sayi2):
    return sayi1/sayi2

print("Hesap Makinesi 1.0")
print("Toplama : 1, Çıkarma :2, Çarpma :3, Bölme :4, Çıkış :q")
while True:
    secim = input("İşleminiz :")
    if secim == "q":
        break

         sayi1 = int(input("Birinci sayıyı girin :"))
    sayi2 = int(input("İkinci sayıyı girin :"))
    if secim=="1":
        print("Sonuç :", toplama(sayi1, sayi2))
    elif secim =="2":
        print("Sonuç :", cikarma(sayi1,sayi2))
    elif secim =="3":
        print("Sonuc :",carpma(sayi1,sayi2))
    elif secim=="4":
        print("Sonuç :",bolme(sayi1,sayi2))
    else:
       print("Yanlış seçim lütfen tekrar deneyin")
        break
```

Çıktı:

```
Hesap Makinesi 1.0
Toplama : 1, Çıkarma :2, Çarpma :3, Bölme :4, Çıkış :q
İşleminiz :2
Birinci sayıyı girin :445
İkinci sayıyı girin :41
Sonuç : 404
İşleminiz :
```

Sıra Sizde: Yıldızlarla üçgen oluşturan örnekten faydalanarak satır sayısını dışarıdan parametre olarak alan ve aşağıdaki gibi bir desen çıktısı verecek fonksiyonu yazınız ve çalıştırınız.

Örnek kullanım:

```
#fonksiyon tanımını burada yapacaksınız.
sayi=int(input("sayi giriniz:"))
if (sayi_asalmi(sayi)):
    print("Sayı asal")
else:
    print("Sayı asal değil")
```

Örnek çıktı:

```
sayi giriniz:5
Sayı asal
```

6.3. Lambda Fonksiyonlar

Python programlama dilinde ihtiyaç hâlinde def ifadesi kullanmadan isimsiz, tek satırlık, küçük fonksiyonlar da tanımlanabilir. Bu fonksiyonlara anonim fonksiyonlar veya tanımlarken kullanılan lambda ifadesi nedeniyle lambda fonksiyonlar denir.

Lambda fonksiyon tanımı lambda [arg1 [,arg2,.....argn]]:ifade şeklinde yapılabilir.

Lambda ile tanımlanan fonksiyonlar herhangi bir sayıda argüman alabilir ancak ifade şeklinde tek bir değer döndürebilir. Ayrıca anonim olduklarından direk çağrılamaz, değişkene atanarak kullanılabilir.

Atandığında ürünün KDV'li fiyatını veren bir lambda fonksiyon tanımlanması ve kullanılması aşağıda gösterilmiştir.

Örnek 21:

```
kdvli_fiyat = lambda fiyat : fiyat * 1.18 # lambda fonksiyon tanımı

print(kdvli_fiyat(100)) #lambda fonksiyon kullanımı
```

Çıktı:

```
118.0
```

Birden fazla argüman alarak atandığında verilen üç sayıyı toplayan lambda fonksiyon aşağıda gösterilmiştir.

Örnek 22:

```
toplam = lambda a, b, c : a + b + c # lambda fonksiyon tanımı
print(toplam(4, 8, 15)) #lambda fonksiyon kullanımı
```

Çıktı:

```
27
```

Lambda fonksiyonu bu kullanımları ile pek de gerekli gözükmüyor olabilir ancak lambda fonksiyonlarının asıl gücü, başka bir fonksiyonun içinde isimsiz bir fonksiyon olarak kullanıldıklarında ortaya çıkar.

Lambda kullanarak bir parametre alıp onu henüz bilinmeyen bir sayı ile çarpan bir fonksiyon tanımlanabilir.

```
def carpan(n):
  return lambda a : a * n
```

Bu fonksiyonun kullanımı aşağıda gösterilmiştir.

Örnek 23:

```
def carpan(n):
  return lambda a : a * n

carpilan = carpan(5)

print(carpilan(10))
```

Çıktı:

```
50
```

Bu kullanımla fonksiyona gönderilen sayı ile çarpan belirlenebilir.

```
def carpan(n):
  return lambda a : a * n

carpilan = carpan(2)
print(carpilan(20))
```

Çıktı:

```
40
```

Örnek 24:

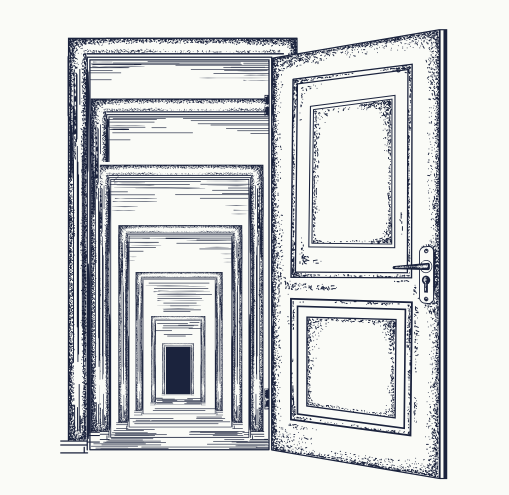
```
def carpan(n):
  return lambda a : a * n

ikikat = carpan(2)
uckat = carpan(3)
dortkat = carpan(4)

print(ikikat(10))
print(uckat(10))
print(dortkat(10))
```

Çıktı:

```
20
30
40
```

6.4. Özyinelemeli Fonksiyonlar

Bir fonksiyon, çözmek için tasarlandığı problemi çözerken kendi içinde yeniden kendini çağırabilir. Fonksiyonun kendi kendini çağırmasına özyineleme (recursion), bu tip fonksiyonlara ise özyinelemeli (recursive) fonksiyon denir. Bu tarz çözüm algoritmalarında döngüler, fonksiyonun kendini kopyalayıp çağırması ile oluşur. Bu kopya, fonksiyonların işleri bittiğinde yok olur. Özyinelemeli fonksiyon kendi kendini çağırabilen fonksiyon türüdür (Görsel 6.2).



Görsel 6.2: Özyinelemeli fonksiyon

Özyinelemeli fonksiyonlar, bazı durumlarda çok kullanışlı ve bazı problemlerin çözümünde elzem olsalar da fonksiyonlarda karmaşıklığı ve programlarda bellek kullanımını arttırır. Bu nedenle Python programlama dili özyineleme derinliğini varsayılan olarak 1000 ile sınırlandırır. Bu sınır aşıldığında fonksiyonunuz aşağıdaki gibi bir RecursionError (özyineleme hatası) verir.

```
Traceback (most recent call last):
   [Previous line repeated 996 more times]
RecursionError: maximum recursion depth exceeded
>>>
```

6.4.1. Özyinelemeli Fonksiyonların Çalışma Şekli

Daha önce yazdığınız fonksiyonlardan birinin içinden tekrar kendini çağırmayı denerseniz programınız sonsuz döngüye girer ve yukarıdaki hatayı alırsınız.

Özyinelemeli fonksiyonlar her zaman kodun başlangıcına yazılan ve yinelemenin sınırlarını belirleyen şart (Base Case) ve fonksiyonun kendisini geri çağıran döngü (Recursive Case) kısmından oluşur. Bu sayede özyinelemeli fonksiyon hata vermeden çalışır.

Tümevarım işlemi yapan yani verilen sayıdan küçük tüm sayıları toplayan bir fonksiyonun özyinelemeli olarak yazılmış hâli aşağıdaki örnekte gösterilmiştir.

Örnek 25:

```
def tumevarim(sayi):
    # Base Case(Şart Durumu)
    if sayi == 1:
        # azalan sayinin 1'e eşit olması
        return 1
    # Recursive Case(Döngü)
    else:
        return sayi + tumevarim(sayi - 1)


print(tumevarim(5))
```

Çıktı:

```
15
```

Fibonacci dizisi, 0 ve 1 ile başlayan ve her sayının kendisinden önce gelen iki sayının toplanması ile elde edilen bir sayı dizisidir. İtalyan matematikçi Leonardo Fibonacci'den adını alır. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946,…

Fibonacci dizisinin ilk 20 elemanını yazdıran bir özyinelemeli fonksiyon aşağıda gösterilmiştir.

Örnek 26:

http://kitap.eba.gov.tr/KodSor.php?KOD=22366

```
def fibonacciler(n):
    if n==1:
        return 1
    elif n==2:
        return 1
    else:
        return fibonacciler(n-1)+fibonacciler(n-2)

for i in range(1,21):
    print(fibonacciler(i), end=" ")

print()
```

Çıktı:

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

Sıra Sizde: Öğrendiklerinizi kullanarak parametre olarak girilen sayı ile faktöriyel hesabı yapan özyinelemeli fonksiyonu yazınız ve deneyiniz.

6.5. Fonksiyonlarda Kullanılan Değişkenlerin Kapsamı

Python programlama dilinde değişkenler yerel (local) ve genel (global) değişken olmak üzere iki farklı şekilde tanımlanabilir. Bir program içinde tanımlanmış tüm değişkenlere program içindeki tüm konumlardan erişilemeyebilir. Örneğin bir fonksiyon içinde tanımladığınız değişken yerel olarak konumlanır ve kapsamı sadece o fonksiyon içindedir. Yani bir fonksiyon içinde tanımladığınız bir değişkene fonksiyon dışından veya bir başka fonksiyon içinden erişilemez.

Fonksiyon dışında tanımlanmış değişkenlerin ise genel kapsamı vardır ve program içinde herhangi bir konumdan ulaşabilirsiniz.

Aynı isimli bir yerel bir de genel değişken tanımlanmış ise öncelik yerel değişkenindir.

Örnek 27:

```
sonuc = 0;
# iki sayıyı toplayan fonksiyon
def toplama( sayi1, sayi2 ):
   # Toplama
   sonuc = sayi1 + sayi2 # Yerel değişken tanımı Genel ile aynı isimlendirilmiş
   print("Fonksiyonun içinde toplam: ", sonuc)
   return sonuc

# Fonksiyonu çağırma
toplama( 5, 15 )
print("Fonksiyonun dışında toplam: ", sonuc)
```

Çıktı:

```
Fonksiyonun içinde toplam:  20
Fonksiyonun dışında toplam:  0
```

Fonksiyon içinde tanımlanan sonuc değişkeni 20 değeri almış olmasına rağmen öncelik yerelde olduğu için genel olarak tanımlanmış sonuc değişkeni 0 değerinde kalmıştır.

Program aşağıdaki gibi düzenlendiğinde:

```
sonuc = 10;
# iki sayıyı toplayan fonksiyon
def toplama( sayi1, sayi2 ):
   # Toplama

   #sonuc = sayi1 + sayi2 # Yerel değişken tanımı Genel ile aynı isimlendirilmiş
   print("Fonksiyonun içinde toplam: ", sonuc)
   return sonuc

# Fonksiyonu çağırma
toplama( 5, 15 )
print("Fonksiyonun dışında toplam: ", sonuc)
```

Çıktı:

```
Fonksiyonun içinde toplam:  10
Fonksiyonun dışında toplam:  10
```

Fonksiyon içinde herhangi bir atama veya tanımlama yapmadan genel (global) değişkene erişildi ve içerdiği değer ekrana yazdırıldı. Ama değer ataması yapılınca "sonuc" adında yeni bir yerel değişken tanımlandı ve atama önceliği onun oldu.

Genel bir değişkene fonksiyon içinden nasıl atama yapılır? Burada devreye global ifadesi girer. Fonksiyon içinde global genel_degisken_adi şeklinde global değişken tanımlaması yapılabilir.

Örnek 28:

```
sonuc = 0;
# iki sayıyı toplayan fonksiyon
def toplama( sayi1, sayi2 ):
   # Toplama
   global sonuc  #kullanacağımız genel değişkeni bildiriyoruz.
   sonuc = sayi1 + sayi2
   print("Fonksiyonun içinde toplam: ", sonuc)
   return sonuc

# Fonksiyonu çağırma
toplama( 5, 15 )
print("Fonksiyonun dışında toplam: ", sonuc)
```

Çıktı:

```
Fonksiyonun içinde toplam:  20
Fonksiyonun dışında toplam:  20
```

Genel değişkene fonksiyon içinden erişilmiş olur.

Sıra Sizde:

1. Parametre olarak girilen sayıdan 0'a kadar olan sayıların çarpımını yapan aşağıdaki özyinelemeli fonksiyonda bir hata bulunmaktadır. Bu hatayı belirleyip hata içermeyen programı yazınız.

```
def sifira_kadar_carp(sayi):
    if sayi == 0:
        return 0
    return sayi * sifira_kadar_carp(sayi-1)

print sifira_kadar_carp(4)

#İşleyişte sıkıntımız nedir, neden sonuc 0 geliyor?
```

UYGULAMA FAALİYETİ 2

Adam Asmaca Oyunu

http://kitap.eba.gov.tr/KodSor.php?KOD=22368

```
   import random


   kelime_listesi = ["türkiye" , "gaziantep" , "istanbul" , "programlama" , "bilgisa-
yar" , "bilişim" , "okul" , "deniz"]
   secili_kelime = random.choice(kelime_listesi) #rastgele bir kelime seçiliyor
   tahmin_sayisi = 5
   harfler = [] #kullanicinin girdigi harleri saklayacağımız liste
   x = len(secili_kelime)
   z = list('_' * x)
   print(' '.join(z), end='\n')
   while tahmin_sayisi > 0:
       harf = input("Bir harf giriniz : ")
       if harf in harfler:
           print("Lütfen daha önce tahmin ettiginiz harfleri tekrar girmeyiniz...")
           continue


       elif len(harf) > 1:
           print("Sadece bir harf girilebilir.")
           continue


       elif harf not in secili_kelime:   #girilen harf kelime icinde yoksa
           tahmin_sayisi -= 1
          print("Harf kelimede yok!. {} tane tahmin hakkınız kaldı.".format(tahmin_sa-
yisi))
   else:
           for i in range(len(secili_kelime)):
               if harf == secili_kelime[i]:
                   print("Doğru Tahmin")
                   z[i] = harf
                   harfler.append(harf)
           print(' '.join(z), end='\n')


       cevap = input("Kelimenin tamamını tahmin etmek istiyor musunuz? ['e' veya 'h']
: ")
```

```
      if cevap == "e":
          tahmin = input("Kelimenin tamamını tahmin edebilirsiniz : ")
          if tahmin == secili_kelime:
              print("Tebrikler bildiniz...")
              break
          else:
              tahmin_sayisi -= 1
              print("Yanlis tahmin ettiniz. {} tane tahmin hakkınız kaldı.".format(-
tahmin_sayisi))



      if tahmin_sayisi == 0:
          print("Tahmin hakkınız kalmadı. Kaybettiniz! Adam Asıldı.")
          break
```

ÖLÇME VE DEĞERLENDİRME 6

1. Belirli işlevleri yerine getiren fonksiyonların bir arada bulunduğu Python dosyalarına ..... denir.
2. sayi isimli parametre alan kare_al fonksiyonunun tanımı ..... satırı ile yapılır.
3. Kendi kendini çağırabilen fonksiyonlara ..... denir.
4. Aşağıdakilerden hangisi fonksiyon kullanmanın faydalarından biri değildir?

   A) Kod tekrarından kurtulunur.
   B) Bellek kullanımından tasarruf edilir.
   C) Karmaşık problemlerin çözümü kolaylaşır.
   D) Kod güzel görünür.
   E) Hata yapma olasılığı azalır.
5. Programlama dili ile beraber gelen fonksiyonlar hangisidir?

   A) Void fonksiyonlar
   B) Özyinelemeli fonksiyonlar
   C) Gömülü fonksiyonlar
   D) Modüller
   E) Math modülündeki fonksiyonlar
6. Aşağıdakilerden hangisi bir gömülü fonksiyon değildir?

   A) input()
   B) int()
   C) print()
   D) str()
   E) selamla()
7. Fonksiyon tanımlamaya yarayan komut aşağıdakilerden hangisidir?

   A) return
   B) def
   C) print
   D) while
   E) global
8. Aşağıdaki hangi ifade değişken tanımlama ile ilintilidir?

   A) global
   B) return
   C) def
   D) print
   E) while

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.

ÖĞRENME BİRİMİ 7

TARİH VE STRİNG (METİN) İŞLEMLERİ

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Python dilinde tarih ve zaman nesnelerinin nasıl saklandığını öğrenecek,
- Datetime modülünü kullanmayı öğrenecek,
- Tarih ve zaman verileri ile işlem yapabileceksiniz.

Anahtar Kelimeler:

Tarih, zaman, nesne, metin biçimlendirme.

Hazırlık Çalışmaları
1. Programlamada nesne kavramını araştırınız.
2. Diğer programlama dillerinde tarih işlemlerinin nasıl yapıldığını araştırıp bilgi sahibi olunuz.

7. TARİH VE METİN İŞLEMLERİ

7.1. Tarih Nesnesi

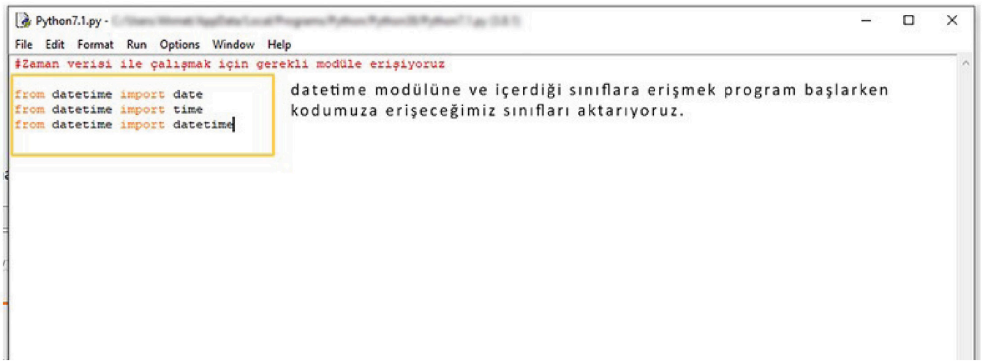
Python dilinde zaman bilgisi tutmak için kendine ait bir veri tipi bulunmaz. Python zaman ve tarih bilgilerini bir değişkene atanmış veri değil de oluşturulmuş bir nesne olarak görüp işler. Bu nedenle zaman nesnesi ile çalışmak için programlama dili ile birlikte gelen datetime modülünü kullanmak gerekir. datetime modülü; zaman, saat ve tarihlerle ilgili işlemler için çeşitli fonksiyonlar ve özellikler sağlayan sınıfları içerir.

Sınıflar, barındırdıkları nesnelerin özelliklerinin ve davranışlarının tanımlandığı genel şablonlardır.

Python dilinin mevcut sürümü olan Python 3.8.2 versiyonunda datetime modülünün barındırdığı, programlarınızda kullanabileceğiniz sınıflar şunlardır:

1. datetime.date: Tarihle ilgili nitelikleri ve fonksiyonları barındıran sınıftır. year (yıl), month (ay) ve day (gün) özelliklerini içerir.
2. datetime.time: Zamanla ilgili nitelikleri ve fonksiyonları barındıran sınıftır. hour (saat), minute (dakika), second (saniye) , microsecond (mikrosaniye) ve tzinfo (saat dilimi) özelliklerini içerir.
3. datetime.datetime: date ve time sınıflarının birleşiminden ve ilave birkaç fonksiyondan oluşur. Örneklerde sıklıkla kullanılacak sınıf budur.
4. datetime.timedelta: İki date, time veya datetime nesnesi arasındaki zaman farkını mikrosaniye cinsinden veren sınıftır.
5. datetime.tzinfo: date ve time sınıflarının saat dilimi özelliklerini tutmak için oluşturulmuş abstract (temel) sınıftır.

Python dili ile modüllere erişmeyi ve içerdiği fonksiyonları kullanmayı geçen bölümde öğrenmiştiniz.



Görsel 7.1: datetime modülünden sınıfları içe aktarma

```
from datetime import datetime
```

satırı ile istediğiniz sınıfı kodunuza aktarıp erişebileceğiniz gibi

```
import datetime
```

satırı ile bütün datetime modülünü içe aktarıp erişebilirsiniz. Ancak fonksiyonlar bölümünden hatırlanacağı üzere kullanılacak tipin başına ait olduğu modül isminin de yazılması gerekir.

now( ) : datetime modülü içindeki datetime sınıfına ait bu fonksiyon içinde bulunulan andaki tarih ve saat bilgilerini verir.

Aşağıdaki örnekte bir tarih nesnesi oluşturulmuş ve o anki zaman bilgisi oluşturulan nesneye atanmıştır.

Örnek 1:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.
an = datetime.now() #tarih nesnesi oluşturulup now() fonksiyonu ile zaman bilgisi atanıyor.


print("tarih ve saat = ",an)  #an nesnesi ekrana yazdırılıyor.
```

Çıktı:

```
tarih ve saat = 2020-06-11 11:38:14.170414
```

today( ) : now() fonksiyonu ile aynı işleve sahiptir. Bulunulan günün tarih ve saat bilgilerini verir. Eğer datetime sınıfı ile değil de date sınıfı ile beraber kullanılırsa sadece yıl, ay ve gün bilgisini verecektir. now() fonksiyonu saat bilgisi içerdiğinden date sınıfı ile beraber kullanılmaz.

Örnekte date modülü ile today( ) fonksiyonu kullanılmıştır.

Örnek 2:

```
from datetime import date #date sınıfı içe aktarılıyor.
bugun = date.today()
print("bugun = ",bugun)
```

Çıktı:

```
bugun =  2020-06-11
```

İstenirse tarih nesnesinin içerdiği veri; yıl, ay, gün, saat, dakika, saniye ve mikrosaniye olarak çağrılıp kullanılabilir.

Örnekte tarih nesnesini oluşturan zaman verileri ayrı ayrı yazdırılmıştır. weekday() fonksiyonu ise haftanın kaçıncı gününde olunduğunu verir.

Örnek 3:

http://kitap.eba.gov.tr/KodSor.php?KOD=22370

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

bugun = datetime.today()

print("bugun = ",bugun)
print(bugun.weekday()) #haftanın kaçıncı günü - Pazartesi 0 ile Pazar 6 arası
print(bugun.year) # yıl bilgisi
print(bugun.month) # ay bilgisi
print(bugun.day)    # gün bilgisi
print(bugun.hour)    # saat bilgisi
print(bugun.minute)   # dakika bilgisi
print(bugun.second)    # saniye bilgisi
```

Çıktı:

```
bugun =  2020-06-11 12:20:25.946320
3
2020
6
11
12
20
25
```

Tarih nesnelerini, kendi tarih verilerinizi vererek de tanımlayabilirsiniz.

Örnekte yıl, ay ve gün verileri verilerek tarih tanımlanmıştır.

Örnek 4:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

dogum_tarihi = datetime(year=1923,month=10,day=29)
print("doğum tarihi = ",dogum_tarihi)
```

Çıktı:

```
dogum tarihi =  1923-10-29 00:00:00
```

Saat verilerini de tanımlama sırasında atayabilirsiniz.

Örnek 5:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

dogum_tarihi = datetime(year=1923,month=10,day=29, hour=8, minute=30, second=35)
print("doğum tarihi = ",dogum_tarihi)
```

Çıktı:

```
dogum tarihi =  1923-10-29 08:30:35
```

Aşağıda programa girdi olarak verilen doğum günü verilerini tarih nesnesine çevirip bugün itibari ile aradaki farkı bulan program gösterilmiştir.

Örnek 6:

```
from datetime import datetime

bugun = datetime.today()
print("Bu günün tarihi", bugun)
yil = int(input("Lütfen doğduğunuz yılı girin:"))
ay = int(input("Lütfen doğduğunuz ayı girin:"))
gun = int(input("Lütfen doğduğunuz günü girin:"))
dogum = datetime(year=yil,month=ay,day=gun)
yas = bugun-dogum
print(yas)
```

Çıktı:

```
Bu günün tarihi 2020-06-11 12:57:33.983500
Lütfen doğduğunuz yılı girin:1979
Lütfen doğduğunuz ayı girin:2
Lütfen doğduğunuz günü girin:20
15087 days, 12:57:33.983500
```

Bu örnekte elde edilen yas verisi aslında yukarıda sınıflarda bahsedilen timedelta nesnesidir. İki tarih arasında aritmetiksel bir işlem yapıldığında sonuç timedelta yani zaman farkı nesnesi ile tutulur.

Yukarıdaki örneğe aşağıdaki satırları ekleyerek timedelta nesnesinin niteliklerine ulaşabilirsiniz.

```
print(yas)
print(yas.days," gün")
print(yas.seconds," saniye"ı
print(yas.microseconds," mikrosaniye")
```

Çıktı:

```
15087 days, 13:43:43.797852
15087  gün
49423  saniye
797852  mikrosaniye
```

timedelta nesnesi ile iki tarih arasındaki farka gün, saniye veya mikrosaniye biçiminde ulaşılabilir.

```
bugun = datetime.date.today()
dun = bugun - datetime.timedelta(days = 1)
yarin = bugun + datetime.timedelta(days = 1)
print("Dün :",dun)
print("Bugün :",bugun)
print("Yarın :", yarin)
```

Çıktı:

```
Dün : 2020-06-10
Bugün : 2020-06-11
Yarın : 2020-06-12
```

Örnekte datetime modülü içe aktarıldı ve bu modüldeki date ve timedelta nesneleri kullanıldı. today() fonksiyonu ile bugünün tarihi alındı. timedelta nesnesine 1 gün fark tanımlanıp bugünün tarihine eklendi ve çıkarıldı.

Aşağıdaki örnekte bugünden 150 gün sonra tarihin ne olacağı hesaplanmıştır.

Örnek 8:

```
import datetime

bugun = datetime.datetime.today()
fark = datetime.timedelta(days = 150)
gelecek = bugun + fark

print("150 gün sonrası :",gelecek)
print("Biçimlendirilmiş Tarih :",gelecek.strftime('%c'))
```

Çıktı:

```
150 gün sonrası : 2020-11-08 15:29:52.305214
Biçimlendirilmiş Tarih : Sun Nov  8 15:29:52 2020
```

Not: Programın sonunda eldeki tarih verisini biçimlendirmek için strftime() isimli bir metot kullanıldı.

7.2. Tarih Bilgisinin Biçimlendirilmesi

Görsel 7.2: Tarih bilgisi

strftime() : strftime() fonksiyonu date, datetime veya time nesnesini kullanarak bize tarih ve zamanı bildiren biçimlendirilmiş string bir değer verir. Böylece size tarih ve zaman bilgilerini ihtiyaçlarınız doğrultusunda biçimlendirme imkânı sunar.

Yukarıdaki örnekte '%c' biçimlendiricisi kullanılarak

```
Sun Nov  8 15:29:52 2020
```

formatında bir çıktı elde edildi. Biçimleyicileri bir arada kullanarak tarih verilerinizi istediğiniz gibi biçimlendirebilirsiniz.

Örnek 9:

```
import datetime


an = datetime.datetime.now()


print(an.strftime('%Y')) # Yıl
print(an.strftime('%X')) # Saat
print(an.strftime('%d')) # Gün - ayın kaçıncı günü
print(an.strftime('%A')) # Gün - İsim olarak
print(an.strftime('%B')) # Ay - İsim olarak
```

Çıktı:

```
2020
16:31:34
11
Thursday
June
```

Biçimlendiriciler kullanarak elinizdeki tarih verisini istediğiniz gibi biçimlendirebilirsiniz. Çıktıda ay ve gün isimlerinin İngilizce olarak yazıldığı görülebilir. Biçimlendirme işlemlerinde yerel yazım formatlarına uygun davranılmak isteniyorsa locale isimli modül çağrılarak programın çalıştığı bilgisayarın yerel dil ayarlarını almasını sağlayabilirsiniz.

```
import locale
locale.setlocale(locale.LC_ALL, '')
```

Satırları programın çalıştığı bilgisayarın dil ayarları ile biçimlendirme yapmasını sağlar veya

```
import locale
locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')
```

çalışılan platformdan bağımsız olarak biçimleyicileri tek bir dile ayarlayabilirsiniz. Yukarıdaki satır programa Türkçe biçimlendirme ayarlarını kullanmasını söyler.

Örnek 10:

```
import datetime
import locale

locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')
an = datetime.datetime.now()

print(an.strftime('%Y')) # Yıl
print(an.strftime('%X')) # Saat
print(an.strftime('%d')) # Gün - ayın kaçıncı günü
print(an.strftime('%A')) # Gün - İsim olarak
print(an.strftime('%B')) # Ay - İsim olarak

print(an.strftime('%d %B %Y')) # gün ay ismi yıl şeklinde
print(an.strftime('%d.%m.%Y tarihinde buluşalım.')) # aralarına nokta konarak
```

Çıktı:

```
2020
16:53:54
11
Perşembe
Haziran
11 Haziran 2020
11.06.2020 tarihinde buluşalım.
```

Tabloda strftime() fonksiyonu ile kullanabileceğiniz biçimlendiricileri inceleyebilirsiniz (Tablo 7.1).

Tablo 7.1: Strftime() Fonksiyonu ile Kullanılabilen Biçimlendiriciler

Biçimlendirici	Açıklama	Örnek
%a	Kısaltılmış gün ismi	Çar
%A	Gün ismi	Çarşamba
%w	Sayı olarak haftanın kaçıncı günü (Pazar 0, Cumartesi 6)	3
%d	Sayı olarak ayın kaçıncı günü	30
%b	Kısaltılmış ay ismi	Haz
%B	Ay ismi	Haziran
%m	Sayı olarak ay (tek haneli ise başına 0 eklenerek)	06
%-m	Sayı olarak ay	6
%y	Yıl – son iki rakam olarak	20
%Y	Yıl	2020
%H	Saat (24-Saatlik format) – 0 eklenmiş sayı	07
%-H	Saat (24-Saatlik format)	18
%I	Saat (12-Saatlik format) 0 eklenmiş sayı	07
%-I	Saat (12-Saatlik format)	6
%p	AM / ÖÖ – PM / ÖS bilgisi	AM
%M	Dakika – 0 eklenmiş sayı	06
%-M	Dakika	6
%S	Saniye – 0 eklenmiş sayı	05
%-S	Saniye	5
%f	Mikrosaniye – 0 eklenmiş sayı	000000
%j	Yılın kaçıncı günü	273
%U	Yılın kaçıncı haftası (Pazar gününden başlayarak)	39
%W	Yılın kaçıncı haftası (Pazartesi gününden başlayarak)	39
%c	Yerel biçim ayarlarına göre tarih ve saat	Çar Haz 10 07:06:05 2020
%x	Yerel biçim ayarlarına göre tarih	06/10/20
%X	Yerel biçim ayarlarına göre tarih ve saat	07:06:05
%%	Karakter girdisi	%

Örnekte biçimlendiricilerin kullanımlarını inceleyebilirsiniz.

Örnek 11:

```
import datetime
import locale
locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')

bugun = datetime.datetime.now()
yarin = datetime.date(bugun.year, bugun.month, bugun.day+1)

print('Bugün :',bugun)
print(bugun.strftime('%d/%m/%Y'))
print('Yarın :',yarin)
print(yarin.strftime('%d/%m (%Y)'))

print(bugun.strftime('%m/%d/%Y, %H:%M:%S'))
print(bugun.strftime('%d %b, %Y'))
print(bugun.strftime('%d %B, %Y'))
print(bugun.strftime('%I%p'))
```

Çıktı:

```
Bugün : 2020-06-12 13:04:50.267157
12/06/2020
Yarın : 2020-06-13
13/06 (2020)
06/12/2020, 13:04:50
12 Haz, 2020
12 Haziran, 2020
01ÖS
```

7.2.1. String (Metin) Olarak Girilen Değerlerin Tarih Bilgisinin Biçimlendirilmesi

Tarih nesnesi oluşturmayı ve içeriğini biçimlendirerek yazdırmayı öğrendiniz. Şimdi metin hâlindeki veriyi tarih nesnesine çevirmeyi öğreneceksiniz. Diyelim ki elinizde input() fonksiyonu ile girilmiş veya başka bir kaynaktan gelmiş "29 Ekim 1923 saat 14:24:11" gibi bir metin verisi var. Bu veriyi çeşitli string fonksiyonları kullanarak anlamlı parçalara bölebilirsiniz ama bu çok meşakkatli bir işlem olacaktır.

strptime(): Bu işlemler için datetime modülünde yukarıdaki biçimlendiriciler ile beraber kullanılacak strptime() fonksiyonu bulunur. Bu fonksiyon tarih ve zaman bilgisi içeren herhangi bir metni veya karakter dizisini tarih nesnesine dönüştürür. Çalışırken bilgiyi içeren metni ve biçimlendiricileri parametre olarak alır.

Örnek 12:

```
import datetime
import locale

locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')

tarih_metni = '29 Ekim 1923 saat 14:32:11'
print("Metin halindeki tarih = ", tarih_metni)
tarih_nesnesi = datetime.datetime.strptime(tarih_metni,'%d %B %Y saat %H:%M:%S')
print("Tarih nesnesi = ",tarih_nesnesi) print(tarih_nesnesi.year)
print(tarih_nesnesi.month)
print(tarih_nesnesi.day)
```

Çıktı:

```
Metin halindeki tarih =  29 Ekim 1923 saat 14:32:11
Tarih nesnesi =  1923-10-29 14:32:11
1923
10
29
```

Sıra Sizde: Aşağıdaki metni oluşturacağınız bir tarih nesnesine atayacağınız programı yazıp deneyiniz. Metindeki formatın ABD tarih formatında olduğunu gözden kaçırmayınız.

Tarih metni : 'Oct. 30, 2014, 6:17 a.m.'

7.3. String (Metin) İşlemleri

Python'da stringler karakterleri temsil eden baytları içeren listelerdir. Python doğrudan karakterleri temsil eden bir veri tipi içermez. Bu nedenle tek bir karakteri temsil eden veri, tek elemanlı bir string dizisidir. String verileri kolayca düzenlemek ve değiştirmek için dil içinde gelen gömülü fonksiyonlar vardır. Python'da string verilerle nasıl çalışıldığı örnekler eşliğinde incelenebilir.

7.3.1. String Verileri Birleştirme

Aşağıdaki kod bloğunda iki karakter dizisi yani string veri oluşturulup birleştirilmiştir.

Örnek 13:

```
ad = "Cüneyt"
soyad = "Arkın"
ad_soyad = ad + soyad
print(ad_soyad)
```

Çıktı:

```
CüneytArkın
```

Ad, soyad arasına boşluk bırakılsın. Bunun için boş bir karakter içeren yeni değişken oluşturulsun.

```
ad = "Cüneyt"
soyad = "Arkın"
ara=" "
ad_soyad = ad + ara + soyad
print(ad_soyad)
```

Çıktı:

```
Cüneyt Arkın
```

Gördüğünüz gibi boşluk da bir karakter olarak algılanır.

7.3.2. String Veri İçindeki Bir Karaktere Erişme

Stringler, karakterlerden oluşmuş listeler olduğundan köşeli parantez ([]) ve index sayısı ile istenilen karaktere rahatça erişebilir.

Örnek 14:

```
sehir = "Gaziantep"
karakter = sehir[3] #dördüncü harfi alıyoruz
print(karakter) #i
print(sehir[0]) #G
print(sehir[5]) #n
```

Çıktı:

```
i
G
n
```

7.3.3. String Verinin Uzunluğu

Daha önce listelerde kullanılan len() fonksiyonu ile string verinin uzunluğunu öğrenebilirsiniz.

Örnek 15:

```
sehir = "Gaziantep"
print("sehir değişken boyutu: ", len(sehir)) #sehir değişkeninin boyutu
boyut=len(sehir)
son_karakter=sehir[boyut-1] # değişkendeki son karaktere erişilir.
print("Son karakter: ", son_karakter)
for harf in sehir:     #stringdeki bütün harflere for döngüsü ile erişilir.
    print(harf)
```

Çıktı:

```
sehir değişken boyutu: 9
Son karakter: p
G
a
z
i
a
n
t
e
p
```

7.3.4. String Veriyi Parçalama (Slice) ve Bölme (Split)

slice() fonksiyonu ile köşeli parantezleri kullanarak metinden parçalar alınabilir. Bunun için köşeli parantez içine çekilmek istenilen karakter aralığının indisini girmek yeterlidir.

Degisken[baslangicIndex : bitisIndex] şeklinde kullanılır. Burada, baslangicIndex dâhil, bitisIndex dâhil değildir.

Örnek 16:

http://kitap.eba.gov.tr/KodSor.php?KOD=22371

```
veri = "Bilişim Teknolojileri"
# Köşeli parantez içerisine yazılan değerler dâhildir.
print(veri)
print(veri[2:8]) # string'in 2 ile 8 aralığındaki değerini alır(2.index dâhil, 8.index dâhil değildir.).
print(veri[2:]) # 2.indexten itibaren stringi bütünüyle alır.
print(veri[5:15]) #(5.index dâhil, 15.index dâhil değildir.)
print(veri[:12]) # 0. indexten 12. index'e kadar string'i parçalayacaktır.
```

Çıktı:

```
Bilişim Teknolojileri
lişim
lişim Teknolojileri
im Teknolo
Bilişim Tekn
```

split() fonksiyonu ile metin verisi belirlenen karakterler baz alınarak bölünebilir. Bölünen metin parçaları dizi hâlinde verilecektir.

Örnek 17:

```
veri = "Bilişim Teknolojileri"
metin="ilkbahar,yaz,sonbahar,kış"

veri_bolum = veri.split(" ") #veri değişkenini boşluktan itibaren bölüyoruz.
metin_bolum = metin.split(",") #metin değişkenini virgüllerden bölüyoruz.
metin_bolum_ikieleman = metin.split(",",1) #1 parametresi vererek metni sadece 2 parçaya bölüyoruz.

print(veri_bolum)
print(metin_bolum)
print(metin_bolum_ikieleman)
```

Çıktı:

```
['Bilişim', 'Teknolojileri']
['ilkbahar', 'yaz', 'sonbahar', 'kış']
['ilkbahar', 'yaz,sonbahar,kış']
```

7.3.5. String Veri İçinde Karakter Değiştirme, Karakter Ekleme ve Çıkarma

replace() fonksiyonu ile bir string içindeki herhangi bir karakter değiştirilebilir.

Örnek 18:

```
kelime = "Bilişim"
cumle = "Merhaba Dünya"
print(kelime)
print(kelime.replace("i","o")) #kelimedeki i'leri o karakteri ile değiştirelim.
print(cumle)
print(cumle.replace("Merhaba","Selam")) #cümledeki "Merhaba"'yı "Selam" ile değiştirelim.
```

Çıktı:

```
Bilişim
Boloşom
Merhaba Dünya
Selam Dünya
```

strip() fonksiyonu ile bir string içinden karakter çıkarılabilir.

Örnek 19:

```
cumle = "  bilişim teknolojilerine giriş  "
# baştaki ve sondaki boşluklar silinir.
print(cumle.strip())
# <bosluk>,b,i,l karakterleri silinir.
print(cumle.strip(" bil"))
# Parametre cümlede bulunmadığı için
# hiçbir karakter çıkarılmaz.
print(cumle.strip("prg"))
cumle2 = 'python çok kullanışlı'
print(cumle2.strip("pyt"))
```

Çıktı:

```
bilişim teknolojilerine giriş
şim teknolojilerine giriş
  bilişim teknolojilerine giriş
hon çok kullanışlı
```

join() fonksiyonu ile bir string verinin içerdiği her bir karakterden sonra yeni bir karakter eklenebilir.

Örnek 20:

```
#Formatı: "eklemek istenilen string ya da karakter değer".join(değişkenin kendisi
= elimizde olan string veri)
kelime = "Gaziantep"
print(".".join(kelime))
```

Çıktı:

```
G.a.z.i.a.n.t.e.p
```

7.3.6. String Veri İçinde Bir Karakterin Yerini veya Metnin Karakteri İçerip İçermediğini Bulma

find() fonksiyonu ile bir metin içinde aranılan karakterin kaçıncı indekste olduğu bulunabilir.

Örnek 21:

```
kelime = "Bilişim Teknolojileri"
# metin içinde 'm' nin indexini arıyoruz.
print(kelime.find('m'))
# metnin içinde 'no' nın indexini arıyoruz, burada indexi n'nin indexi(11)
olarak geri döndürecektir.
print(kelime.find('no'))
# 4. indexten başlayarak metin içinde i'yı tarayacaktır.
print(kelime.find('i',4))
# 1. index ile 7. index dahil olmak üzere aradaki metinde 'i' yı arayacaktır.
print(kelime.find('i',1,7))
# eğer aradığınız veri metinde yoksa -1 sonuç olarak döndürülecektir.)
print(kelime.find('z'))
```

Çıktı:

```
6
11
5
1
-1
```

in operatörü ile bir metin içinde herhangi bir karakterin olup olmadığını sorgulayabilirsiniz. Bu operatör geriye boolean , "True" ya da "False" bir değer döndürür. Aranan karakterler metin içinde bulunduğunda "True", aksi takdirde "False" döndürür.

Örnek 22:

```
kelime = "Gaziantep"
print("G" in kelime)
print("k" in kelime)
```

Çıktı:

```
True
False
```

Örnek 23:

```
kelime = "Gaziantep"
print("G" not in kelime)
print("k" not in kelime)
```

Çıktı:

```
False
True
```

7.3.7. String Veri İle Büyük ve Küçük Harf Değişimi Yapma

Python'da karakter dizilerinde büyük ve küçük harf değişikliği için kullanılabilecek aşağıdaki fonksiyonlar vardır.

upper() : Karakter dizisindeki bütün harfleri büyütür.

lower() : Karakter dizisindeki bütün harfleri küçültür.

capitalize() : Karakter dizisinin ilk harfini büyütür.

title() : Karakter dizisindeki her kelimenin ilk harfini büyütür.

swapcase() : Karakter dizisindeki büyük harfleri küçük, küçük harfleri büyük hâle getirir.

Örnek 24:

```
kelime = 'Bilişim teknolojileri'
print(kelime.upper()) # metni büyük harfe çevirir.
print(kelime.lower()) # metni küçük harfe çevirir.
print(kelime.capitalize()) # dizinin harfini büyütür.
print(kelime.title()) # kelimelerin ilk harflerini büyütür.
print(kelime.swapcase()) # büyük  küçük değişimi yapar.
```

Çıktı:

```
BILIŞIM TEKNOLOJILERI
bilişim teknolojileri
Bilişim teknolojileri
Bilişim Teknolojileri
bILIŞIM TEKNOLOJILERI
```

ÖLÇME VE DEĞERLENDİRME 7

1. Aşağıdakilerden hangisi datetime modülünün içerdiği sınıflardan değildir?

   A) date
   B) datetime
   C) math
   D) time
   E) timedelta

2. Bulunulan günün tarih ve saat bilgilerini veren fonksiyon hangisidir?

   A) print()
   B) float()
   C) pow()
   D) now()
   E) sin()

3. Bir metindeki bütün harfleri büyük yapan fonksiyon aşağıdakilerden hangisidir?

   A) lower()
   B) capitalize()
   C) upper()
   D) title()
   E) print()

4. Aranılan karakterin bir metin içinde kaçıncı indekste olduğunu veren fonksiyon ..... fonksiyonudur.
5. Gün ismini "Çarşamba" olarak yazdıran biçimlendirici aşağıdakilerden hangisidir?

   A) %b
   B) %A
   C) %a
   D) %w
   E) %d

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.

ÖĞRENME BİRİMİ 8

HATA YAKALAMA İŞLEMLERİ

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Hata türlerini açıklayabilecek,
- Hangi durumlarda hata kontrolü yapmanız gerektiğini öğrenecek,
- Hata durumunda, hata yakalama ve işleme işlemlerini yapabilecek,
- Kod ile hata üretebilecek,
- Programınıza test ifadeleri yazabileceksiniz.

Anahtar Kelimeler:

Hata, yazım hatası, mantıksal hata, bug, istisnai hata, try, except, finally, raise, assert.

Hazırlık Çalışmaları

1. Kullanıcılardan alınan verilere her zaman güvenmeli miyiz?
2. Geçmişte yapılmış en büyük yazılım hatalarını araştırınız.

8. HATA YAKALAMA İŞLEMLERİ

8.1. Hata Kavramı ve Hata Türleri

8.1.1. Hata Nedir?

Programlar, -özellikle başlangıç seviyesinde- genellikle en iyi duruma göre yazılır. Yani tüm yazılım ve donanım kaynaklarının beklenen şekilde çalışacağı ve kullanıcıların programı yazılımcının ondan beklediği şekilde kullanacağı varsayılır. Benzer şekilde bir hesaplama işleminin her durumda doğru çalışması beklenir. Ancak özellikle profesyonel seviyede, bu asla olmaması gereken bir durumdur. Programcının bütün iyi niyetli yaklaşımına rağmen işler her zaman istenildiği gibi gitmeyebilir. Bu nedenle iyi bir programcı; yazılım, donanım ve kullanıcı kaynaklı birçok hatayla karşılaşacağını bilmeli ve bunlara yönelik önlemlerini almalıdır. Programın asla kendi kontrolü dışında sonlanmasına izin vermemelidir. En kötü durumda bile kullanıcıların anlayabileceği hata mesajları vererek programı sonlandırmalıdır.

İki sayıyı toplarken herhangi bir hata olmayacağı varsayılabilir. "3+5" çoğu zaman doğru ve hatasız bir şekilde hesaplanır. Ancak farklı durumlar için her ihtimal göz önünde bulundurulmalıdır. Örneğin; herhangi bir veri hard diske kaydetmeye çalışıldığında karşılaşılabilecek durumlar şunlardır:

Dosya adı hatalı olabilir.
Diskte, dosyayı kaydetmeye yetecek boş yer kalmamış olabilir.
Dosya oluşturma / yazma izni olmayabilir.
Disk bozuk olabilir.
İşletim sistemi doğru çalışmayabilir.
Dosyanın kaydedilmesi anında elektrik kesintisi yaşanabilir.
Dosya uzak bir makineye kaydedilecekse
- o Kullanıcı adı / şifresi hatalı olabilir.
- o Ağ bağlantısında sıkıntı olabilir.
- o Kullanıcının yetkilendirmesinde hata olabilir.

Programcı, programını yazarken bu gibi durumları her zaman göz önünde bulundurmalıdır.

8.1.2. Hata Türleri

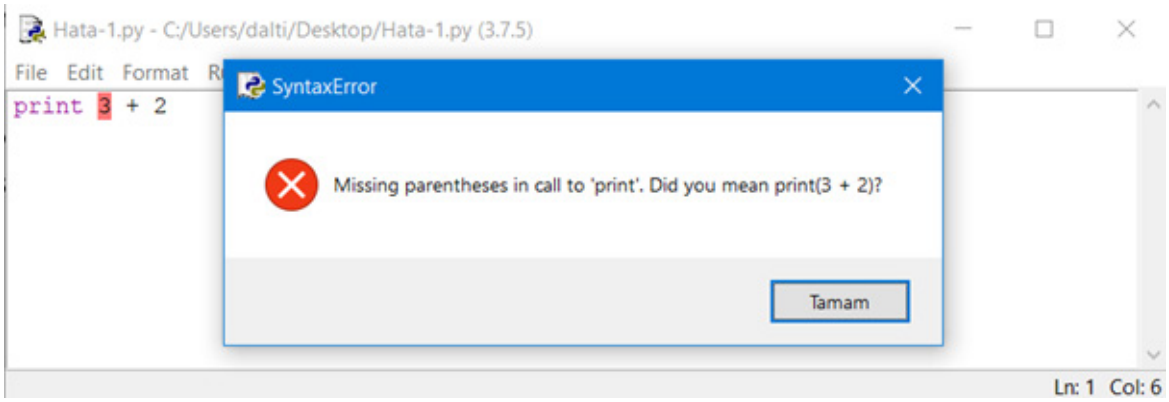
Hata kavramının sadece program dışı parametrelerde oluşabileceği düşünülmemelidir. Hataları üçe ayırmak mümkündür:

1. Programcı hataları / Yazım hataları (Syntax errors)
2. Mantıksal hatalar (Bugs)
3. İstisnai hatalar (Exceptions)

8. 1. 2. 1. Programcı Hataları/Yazım Hataları

Programı yazan kişiden kaynaklanan hatalardır. Çoğunlukla dikkatsizlik sonucu oluşur.

Örneğin; 2 sayının toplamını hesaplayan bir program aşağıdaki şekilde yazılsın ve çalıştırılsın.



Program, doğal olarak doğru çalışmadı. Açıklama mesajında da yazdığı üzere doğrusu

```
print(3 + 2)
```

şeklinde olmalıdır. Parantez açma / kapama unutulduğu için kod doğru bir şekilde çalışmamıştır.

Bir değişkene değer atanıp ileriki satırlarda değişkenin değeri ekrana yazdırılmak istensin.

```
ogrenci_adi = "Filiz"
# Diğer kodlar
# Diğer kodlar
# Diğer kodlar
print(ogrnci_adi)
```

Program çalıştırıldığında yine düzgün çalışmadığı görülecektir çünkü değişkenin değeri ekrana

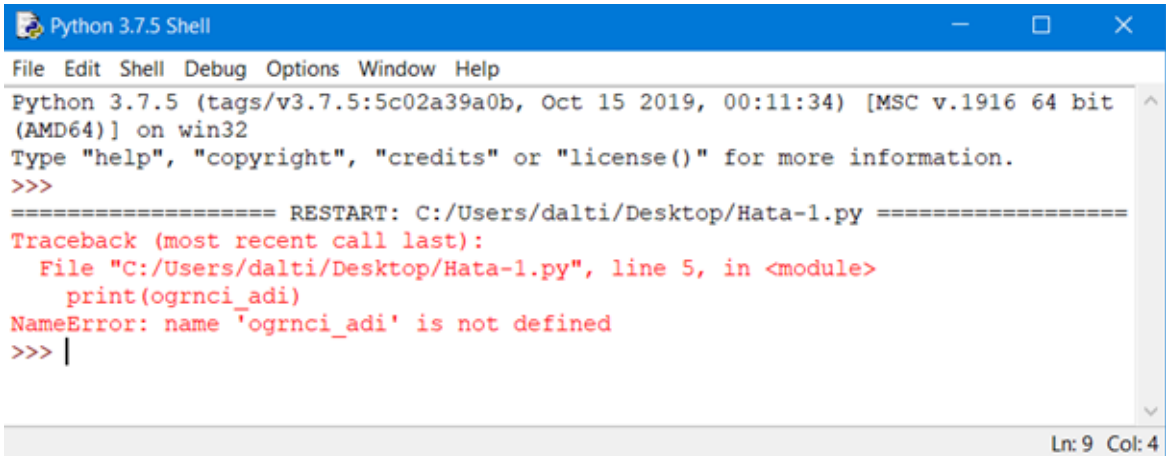
yazdırılmak istendiğinde değişken adı programcı tarafından yanlış yazılmıştır ("ogrenci_adi" yerine "ogrnci_adi" yazıldığına dikkat ediniz.).

Doğrusu;

```
print(ogrenci_adi)
```

şeklinde olmalıdır.

Bu hata türü, çözülmesi en kolay hata türüdür çünkü hatalı olan satır rahatlıkla tespit edilip hata hemen düzeltilebilir.



Yukarıdaki hatalı program çalıştırıldığında, 5. satırda "ogrnci_adi" değişkeninin tanımlı olmadığı mesajı gösterilmektedir.

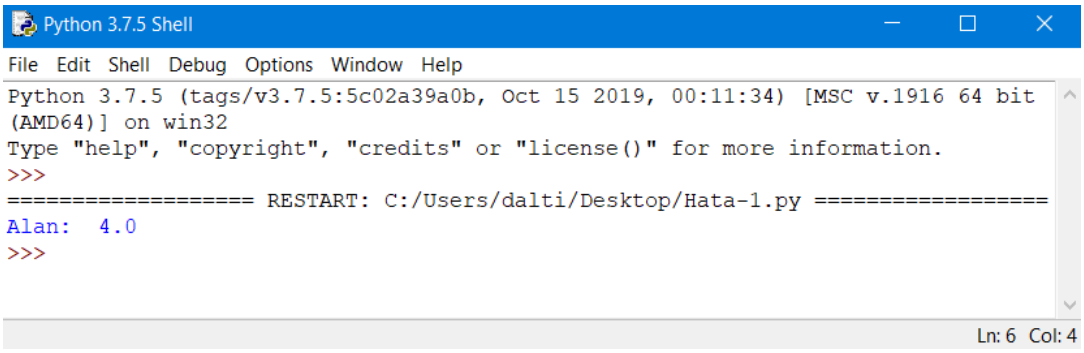
8. 1. 2. 2. Mantıksal Hatalar (Bugs)

Bu hata türünün tespiti ve çözülmesi daha zordur. Çünkü program hata vermeden çalıştığı hâlde programda hesaplanan sonuçlar yanlıştır. Özellikle programdaki satır sayısı arttıkça bu tür hataların tespiti de zorlaşmaktadır.

Örneğin, bir üçgenin alan hesabı formülü "(taban kenarı * yükseklik) / 2" şeklindedir. Bu formüle göre hesap yapan bir program yazılsın:

```
taban_kenari = 4
yukseklik = 3
alan = (taban_kenari * yukseklik) / 3
print("Alan: ", alan)
```

Programcı 3. satırda "/ 2" yazması gerekirken yanlışlıkla "/ 3" yazmıştır.



Her şey yolundaymış gibi görünse de üçgenin alan hesabı yanlış yapılmıştır. Program hata vermeden çalışmış ve sonucu göstermiştir. Bu hatayı fark etmek zordur ve programcı mutlaka hesaplama değerlerini test etmelidir.

Bu tür hatalara bug (böcek), ilgili hatayı bulup düzeltme işlemine de debug (ayıklama) denir.

8. 1. 2. 3. İstisnai Hatalar

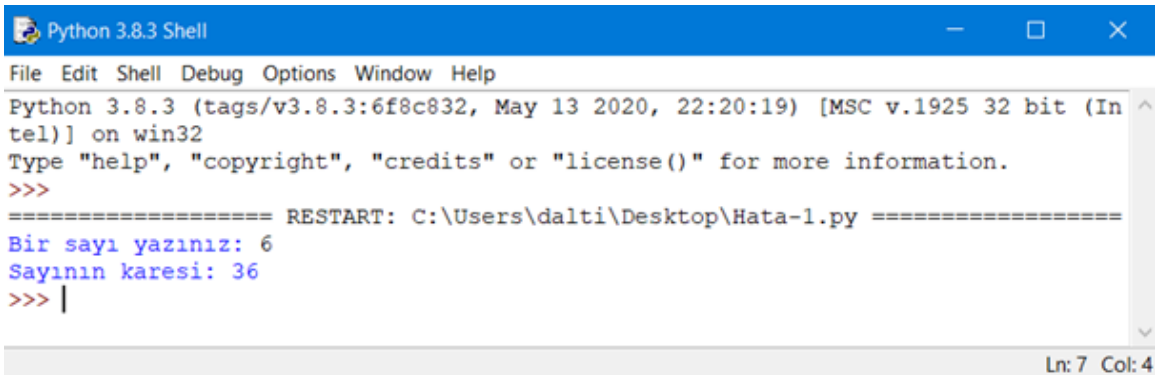
Bu tür hatalar, programın çalışması esnasında oluşan, aslında gerçekleşmesi beklenmeyen hatalardır.

Örneğin; kullanıcıdan bir sayı alıp karesini ekrana yazan bir program yazılsın.

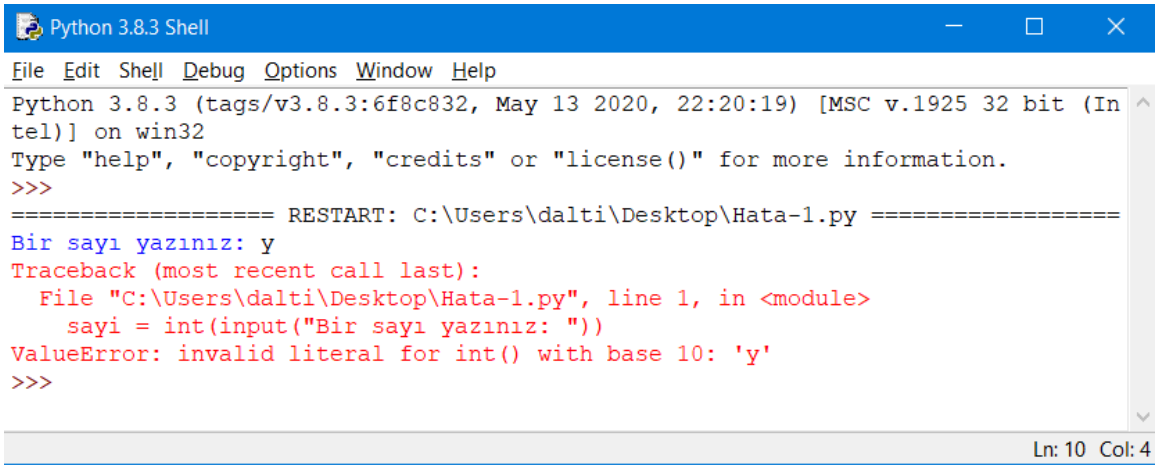
```
sayi = int(input("Bir sayı yazınız: "))
karesi = sayi * sayi
print("Sayının karesi:", karesi)
```

Kullanıcının girdiği metinsel bilgi sayıya çevrilip sonraki satırda karesi hesaplanmakta ve son olarak ekrana yazdırılmaktadır.

Kullanıcının ilk olarak "6" sayısını girdiği düşünülsün.



Program düzgün ve doğru bir şekilde çalıştı. Program bir kez daha çalıştırılsın ve bu kez kullanıcının "6" yerine yanlışlıkla klavyeden "y" girdiği düşünülsün:



Program "y" bilgisini sayıya çeviremediği için "invalid literal for int()…" şeklinde bir hata mesajı alınmaktadır.

Bu örnekte kullanıcının sayı girmemesiyle ortaya çıkan hata, bir istisnadır. Dolayısıyla programın istenmeden sona ermesine sebep olan bu durumun programcı tarafından kontrol altına alınması gerekir.

8.2. Hata Yakalama

Bir önceki konuda hatalardan ve hata türlerinden bahsedildi. Python programlama dilinde hata yakalama try-except blokları aracılığıyla yapılmaktadır.

try.. except..

En temel try-except bloğu şu şekildedir:

```
try:
    # hata oluşması muhtemel kod bloğu
except:
    # hata durumunda yapılacak işlemler
```

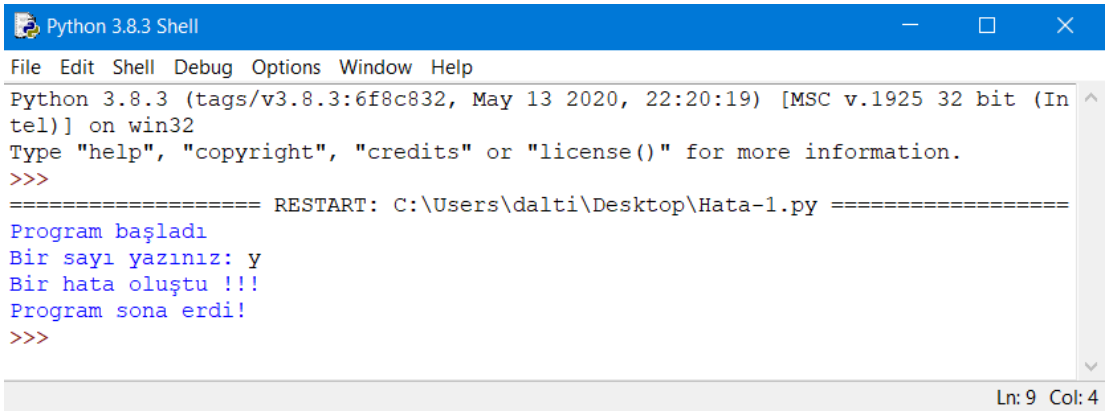
try bloğunda, hata oluşma ihtimali bulunan kodlar yazılır ve eğer bir hata oluşursa except bloğu devreye girer. Hata oluştuğunda except bloğunun devreye girmesi "Hatayı yakalama" olarak adlandırılır. Hata olmazsa program çalışmaya except bloğundan sonraki satırdan devam edecektir.

Daha önceden örnek olarak yazılan sayının kareye çevrilmesi programı, try-except bloklarıyla tekrar yazılırsa şu şekilde bir kod ortaya çıkar:

http://kitap.eba.gov.tr/KodSor.php?KOD=22372

```
print("Program başladı")
try:
    sayi = int(input("Bir sayı yazınız: "))
    karesi = sayi * sayi
    print("Sayının karesi:", karesi)
except:
    print("Bir hata oluştu !!!")
print("Program sona erdi!")
```

Kullanıcının yine "6" yerine yanlışlıkla "y" girdiği düşünüldüğünde program şu şekilde çalışacaktır:



Görüldüğü üzere, program hata mesajı vererek sonlanmadı. Program, programcının kontrolü altında çalışmaya devam etti. Programcı burada, istisnai bir durum karşısında programının yarıda kesilmesini engellemiştir.

8.3. Python Hata Türleri

Python'da oluşabilecek tüm istisnai durumlar için özel hata türleri bulunmaktadır. Hata türlerinin hiyerarşik yapısı aşağıda görülebilir:

```
+-- Exception
      +-- StopIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
```

```
 +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
     |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```

8.3.1. Birden Fazla "Except" Bloğu

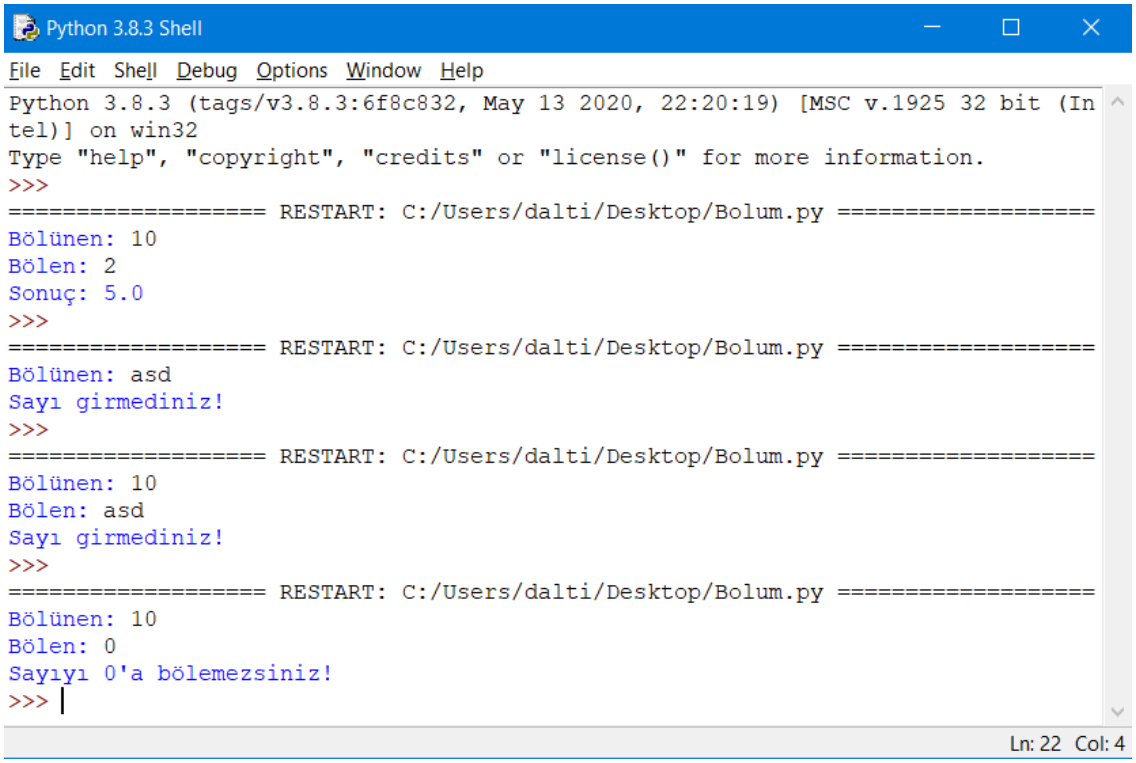
try-except bloğu ile tüm hataları yakalayabilirken gerektiğinde oluşabilecek hata türüne göre except bloğu özelleştirilebilir.

Örnek olarak 2 sayının birbirine bölümü istenildiği düşünülsün. Kullanıcıdan 2 adet sayı bilgisi istensin ve bölüm işlemi hesaplanarak ekrana yazdırılsın.

Burada hemen akla gelen iki muhtemel durum vardır. Birincisi, kullanıcı sayı değil de başka bir şey girerse, ikincisi de kullanıcı bölen olarak "0" girerse oluşacak hatalardır. Matematikte ve bilgisayarlarda bir sayının sıfıra bölünmesi tanımsızdır. Bir sayının sıfıra bölünmesi mümkün değildir. Bu hata ihtimallerini de düşünerek yazılabilecek kod aşağıdaki şekilde olabilir:

```
try:
    sayi1  = int(input("Bölünen: "))
    sayi2  = int(input("Bölen: "))
    sonuc = sayi1 / sayi2
    print("Sonuç:", sonuc)
except ValueError:
    print("Sayı girmediniz!")
except ZeroDivisionError:
    print("Sayıyı 0'a bölemezsiniz!")
```

Programa farklı değerler girilerek birkaç kez çalıştırıldığında aşağıdaki gibi bir sonuç elde edilir:



Sonuç ekranında da görüldüğü üzere hem sayıya çevrilme hatası hem de sıfıra bölme hatası ayrı ayrı yakalanmıştır. Bu örnekten yola çıkarak farklı türde hatalar oluştuğunda farklı işlemler yaptırılması mümkündür denilebilir. Genel yapı şu şekildedir:

```
try:
   # çalıştırılacak kodlar
except hatatipi1:
   #  Yapılacak işlemler
except hatatipi2:
   #  Yapılacak işlemler
except hatatipi3:
   #  Yapılacak işlemler
```

Eğer birden fazla hatada aynı işlemlerin yapılması isteniyorsa şu şekilde de bir kullanım mümkündür:

```
try:
    # çalıştırılacak kodlar
except (hatatipi1, hatatipi2):
    #  Yapılacak işlemler
except hatatipi3:
    #  Yapılacak işlemler
```

8.3.2. "as" İfadesi ile Orijinal Hata Mesajı Gösterme

except bloğunda istenilen hata mesajı gösterilebildiği gibi Python tarafından oluşturulan orijinal hata mesajı da gösterilebilir.

Bunun için as deyimi kullanılır.

```
try:
    sayi1  = int(input("1. sayı: "))
    sayi2  = int(input("2. sayı: "))
    toplam = sayi1 + sayi2
    print("Toplam:", toplam)
except ValueError as hata:
    print("Sayı girmediniz!")
    print("Orjinal hata mesajı:", hata)
```

except bloğunda, oluşan hata bilgisi "hata" değişkeninde tutulmaktadır.

8.3.3. "finally" Bloğu

try bloğunda yazılan kodlarda hata olsa da olmasa da çalışması istenilen kodlar finally bloğuna yazılır.

Genel yapısı şu şekildedir:

```
try:
    # çalıştırılacak kodlar
except:
    # hata oluştuğunda yapılacak işlemler
finally:
    # hata olsa da olmasa da çalışacak kodlar
```

Özellikle dosya işlemlerinde, veri tabanı işlemlerinde kullanılması gereken bir yapıdır. Bir dosya ya da veri tabanı bağlantısı açıldığında mutlaka bir şekilde kapatılmalıdır.

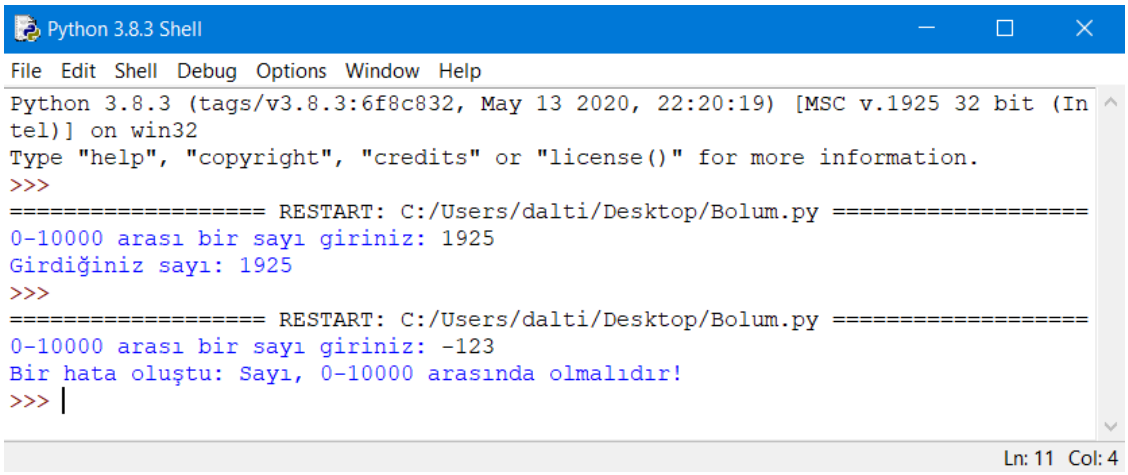
8.3.4. "raise" İfadesi

Yazılan kodlarda kasıtlı olarak bir hata oluşturulması istenebilir. Örneğin; kullanıcıdan 0-100 arası bir sayı girmesi istendiğinde, kullanıcı "-5" ya da "110" değerini girerse, Python açısından hiçbir hata olmamasına rağmen istenilen değer aralığında olmadığı için bir hata üretilebilir (Bu durum bazı kaynaklarda "hata fırlatma" olarak da geçmektedir.). Oluşan bu hatayı da except bloğu içinde yakalamak mümkündür.

```
try:
    sayi = int(input("0-10000 arası bir sayı giriniz: "))
    if(sayi not in range(0, 10001)):
        raise Exception("Sayı, 0-10000 arasında olmalıdır!")
    print("Girdiğiniz sayı:", sayi)
except Exception as hata:
    print("Bir hata oluştu:", hata)
```

http://kitap.eba.gov.tr/KodSor.php?KOD=22373

raise Exception("…") satırında, istenilen hata mesajı üretilmiş ve except bloğunda yakalanmıştır.



8.3.5. "assert" İfadesi

http://kitap.eba.gov.tr/KodSor.php?KOD=22375

Program yazılırken programın herhangi bir satırında bir değişkenin istenilen değere sahip olup olmadığının test edilmesi gerekebilir. Bunun için print komutu ile değişkenin değerini ekrana yazdırmak bir çözüm olsa da çok fazla ekran çıktısı arasında istediğimiz değeri görmek zorlaşacağı için sadece test ifadesi sağlanmadığında bir hata üretmek assert ifadesi ile mümkündür.

Genel olarak kullanımı şu şekildedir:

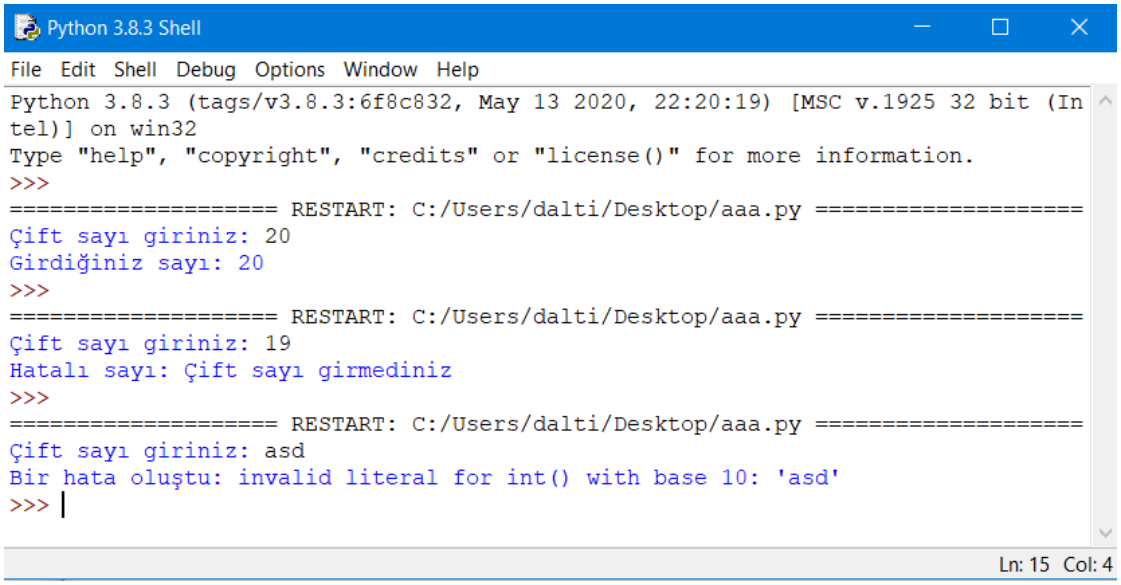
```
assert koşul, mesaj
```

Koşul ifadesi true ya da false değerlerinden birini üreten bir ifadedir ve eğer bu değer true ise bir sonraki satırdan program devam edecek; eğer false ise "mesaj" bilgisi kullanılarak "AssertionError" tipinde bir hata oluşturulacaktır.

Örneğin kullanıcıdan bir çift sayı girmesi istensin:

```
try:
    cift_sayi = int(input("Çift sayı giriniz: "))
    assert cift_sayi % 2 == 0, "Çift sayı girmediniz"
    print("Girdiğiniz sayı:", cift_sayi)
except AssertionError as hata:
    print("Hatalı sayı:", hata)
except Exception as hata:
    print("Bir hata oluştu:", hata)
```

"cift_sayi % 2 == 0" ifadesi ile kullanıcının girdiği değerin çift sayı olup olmadığı test edilmektedir. Eğer çift sayı ise bir sonraki satırda ekrana yazdırılacak, çift sayı değilse de assert ifadesi otomatik olarak "AssertionError" tipinde bir hata oluşturacaktır. Except bloğu içinde as ifadesi ile oluşturulan hata mesajının ekrana yazdırıldığı görülebilir.



ÖLÇME VE DEĞERLENDİRME 8

1. Bir try-except bloğunda kaç tane except bloğu olabilir?
2. Aşağıdaki kod bloğu geçerli midir?

```
try:
    # Kodlar...
except:
    # Kodlar...
finally:
    # Kodlar...
```

3. Bir except bloğunda birden fazla hata yakalanabilir mi? Örnek vererek anlatınız.
4. Aşağıdaki kod bloğunun çıktısı ne olur?

```
def func():
    try:
        return 1
    finally:
        return 2
k = func ()
print(k)
```

A) 0 B)1 C) 2 D) 3 E) Hata verir.

5. 1 ifadesi ne sonuç verir?

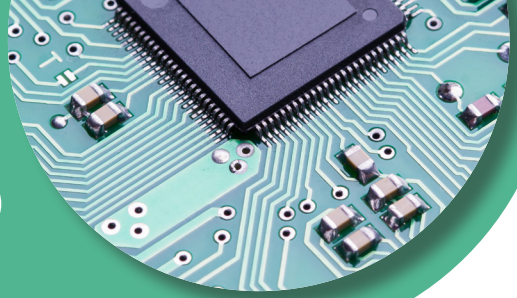
A) True B) False C) Hiçbir sonuç vermez.

D) TypeError hatası E) ValueError hatası

6. Aşağıdaki kodun çıktısı ne olur? Satır satır açıklayınız.

```
print("Program başladı")
try:
    raise Exception('Bir hata oluştu!')
except:
    print("Except bloğuna geldik.")
    raise
finally:
    print("Finally bloğuna geldik.")
print("Program bitti")
```

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.



ÖĞRENME BİRİMİ 9

DOSYA İŞLEMLERİ

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Python dilini kullanarak dosya ve dizin oluşturabilecek,
- Dosyalara erişip içeriğini okuyabilecek ve değiştirebilecek,
- Dosyaları silme ve yedekleme işlemlerini yapabileceksiniz.

Anahtar Kelimeler:

Dosya, dizin, veri yazma, kopyalama.

Hazırlık Çalışmaları
1. İnternet üzerinden dosya, klasör, dosya sistemi konularını araştırınız.
2. Günümüzde bilgisayarlarda kullanılan dosya sistemlerini araştırınız.

9. DOSYA İŞLEMLERİ

9.1. Çalışma Dizini Ayarları ve Klasör Oluşturma

Dosya okuma ve yazma işlemlerine geçmeden önce Python'un çalışma dizini ile ilgili ayarlarının yapılması gerekmektedir. Python dilinin varsayılan çalışma dizini, programın kurulduğu klasördür. Python'un bilgisayarın dosya sistemlerine erişmesi için "os" adındaki Python modülünün çağrılması gerekir. Komut satırından bu modüle ait getcwd() fonksiyonu çağrılarak programın mevcut çalışma dizini öğrenilebilir.

```
>>> import os
>>> os.getcwd()
'C:\\Users\\Ahmet\\AppData\\Local\\Programs\\Python\\Python38'
>>>
```

Aynı komutlar aşağıdaki gibi bir Python dosyasına yazılarak da çalıştırılabilir. Klasör, dosya oluşturma ve silme gibi işlemlerin çoğunlukla program içinden yapılması gerekir. Bu nedenle bu yöntemi kullanmanız önerilir.

Örnek 1:

```
import os
dizin = os.getcwd()
print(dizin)
```

Çıktı:

```
C:\Users\Ahmet\AppData\Local\Programs\Python\Python38
```

Mevcut çalışma dizinini değiştirmek için chdir() fonksiyonu kullanılır. Yukarıdaki kod bloğu, çalışma dizinini c sürücüsünde test klasörü ("c:\test") olarak değiştirecek şekilde geliştirilebilir.

```
import os
dizin = os.getcwd()
print(dizin)
os.chdir('c:\\test')
yeni_dizin = os.getcwd()
print(yeni_dizin)
```

Çıktı:

```
Traceback (most recent call last):
  File "C:/Users/Ahmet/AppData/Local/Programs/Python/Python38/2.py", line 4, in
<module>
    os.chdir('c:\\test')
FileNotFoundError: [WinError 2] The system cannot find the file specified: 'c:\\test'
```

Kod bloğu çalıştırıldığında yukarıdaki hata ile karşılaşılacaktır. C: sürücüsünde "test" isimli bir klasör olmadığı için Python belirtilen dizini bulamaz. Bu nedenle çalışma dizini değiştirme işlemi başarısız olur. Bilgisayarın c: sürücüsünde ilgili klasör oluşturularak bu sorun çözülebilir ancak klasör ve dosya oluşturma işlemlerini kod bloğu içinde yapmak daha doğru bir yöntemdir. Python dili ile dosya veya dizin oluşturmak için gerekli bilgiler ve fonksiyonlar, aşağıdaki konu başlıklarında ve örneklerde adım adım incelenecektir.

9.1.1. Yol (Path) Tanımlama

Yol (Path), dosya veya klasörün bilgisayarın dosya sistemindeki konumunu belirtir. Doğal olarak bir dosyaya erişmek, bir dosyayı silmek veya değiştirmek için o dosyanın yolunun bilinmesi gerekir. Yazılan kodların platformdan bağımsız olarak Windows, Linux, Unix ve Mac OS gibi bütün işletim sistemlerinde çalışabilmesi için dosya ve dizin işlemlerini platformdan bağımsız yollar tanımlayarak yapmak gerekecektir. Python, yol tanımlama işlemleri için kullanışlı birçok fonksiyonu barındıran os.path alt modülünü içerir.

join() fonksiyonu, Windows işletim sistemi için ters bölü (\), Unix işletim sistemi için bölü (/) işaretini yol bileşenlerinin arasına ekleyerek birleştirir. Tek bir fonksiyon ile hem Windows hem de Unix işletim sisteminde doğru çalışacak "path" bilgisi oluşturulmuş olur.

split() fonksiyonu ise verilen yol bilgisini parçalara ayırarak liste hâlinde verir.

Örnek 2:

```
import os
yol = os.path.join('test','python')
print(yol) # test\python (Windows Sistemde)
pc = os.path.split(yol)
print(pc) # ('test', 'python')
```

Çıktı:

```
test\python
('test', 'python')
```

Önemli Not: Yukarıdaki kod bloğunda "print(yol)" komut satırı Windows işletim sisteminde "test\python" şeklinde çıktı üretirken Unix işletim sisteminde "test/python" şeklinde çıktı üretecektir. Ayraçların birbirinden farklı olduğuna dikkat ediniz.

9.1.2. Yolu Bilinen Klasör veya Dosyanın Var Olup Olmadığını Kontrol Etme

Program yazılırken dosya yolu verilen bir dosyanın veya dizinin var olup olmadığının bazı durumlarda kontrol edilmesi gerekir. Bunun için yolun var olup olmadığını belirten exists() ve dizin olup olmadığını kontrol eden isdir() fonksiyonları kullanılır.

Örnek 3:

```
import os
yol = os.path.join("C:\\","test")
if os.path.exists(yol):
    print(yol + ' : var')
    if os.path.isdir(yol):
        print(yol + ' : bir dizin')
else:
    print("Yol bulunamadı.")
```

Çıktı:

```
Yol bulunamadı.
```

Kod bloğu çalıştığında program belirtilen yolun varlığını araştırmış ve bulamadığı için hata vermek yerine ekrana "Yol bulunamadı" çıktısı vermiştir.

9.1.3. Klasör Oluşturma

mkdir() veya makedirs() fonksiyonları kullanılarak belirlediğiniz konumda yeni bir klasör oluşturulabilir. Yeni bir klasör oluşturulurken öncelikle belirtilen konumda aynı klasörün var olup olmadığı kontrol edilmelidir. Yukarıdaki kodları eğer klasör yoksa oluşturacak şekilde düzenleyebilirsiniz.

```
import os


yol = os.path.join("C:\\","test2")
if os.path.exists(yol):
    print(yol + ' : var')
    if os.path.isdir(yol):
        print(yol + ' : bir dizin')
else:
    print("Yol bulunamadı!")
    os.mkdir(yol)
    print("oluşturuluyor...")
```

Çıktı:

```
Yol bulunamadı!
oluşturuluyor...
```

Kodunuzu bir kez daha çalıştırdığınızda klasörün artık oluşturulduğunu ve ekran çıktısının değiştiğini görürsünüz.

Çıktı:

```
C:\test : var
C:\test : bir dizin
```

Yukarıdaki örnek tekrar çalıştırıldığında çalışma dizini değiştirilmiş olacaktır.

```
import os
dizin = os.getcwd()
print(dizin)
os.chdir('c:\\test')
yeni_dizin = os.getcwd()
print(yeni_dizin)
```

Çıktı:

```
C:\Users\Ahmet\AppData\Local\Programs\Python\Python38
c:\test
```

Programın çalışma dizini "c:\test" olarak değiştirilmiş oldu. Sonraki birimlerde dosyaları oluşturmak ve saklamak için bu dizin kullanılacaktır.

Sıra Sizde: os modülü ile gelen rename() fonksiyonu dosya / dizin isim değiştirme işlemleri için rmdir() fonksiyonu ise dizin silme işlemleri için kullanılır. Siz de bu fonksiyonları kullanarak deneme isimli bir dizin oluşturup daha sonra bu dizinin ismini değiştirip silen programı öğretmenlerinizle birlikte yazınız.

9.1.4. Dosyalara Erişme ve Okuma

Python dili dosya işlemleri için oldukça kullanışlı gömülü fonksiyonlar içerir. Bu fonksiyonları herhangi bir modülü içe aktarmadan kullanabilirsiniz. Önce var olan bir dosyayı açıp içeriğinin nasıl okunabileceği öğrenilmelidir.

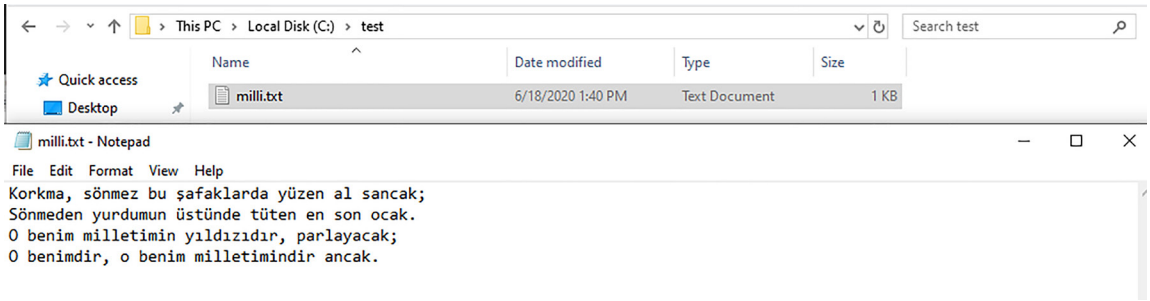
Çalışma dizini "C:\test" klasörü olarak ayarlanmıştı. Örneklere başlamadan önce bu dizinde milli.txt isminde bir metin dosyası oluşturunuz ve içeriğini yazıp kaydediniz.

"Korkma, sönmez bu şafaklarda yüzen al sancak;

Sönmeden yurdumun üstünde tüten en son ocak.

O benim milletimin yıldızıdır, parlayacak;

O benimdir, o benim milletimindir ancak."



Görsel 9.1: Test klasöründe milli.txt dosyası

Mevcut bir dosyayı açmak için open() fonksiyonu kullanılır. open() fonksiyonu, belirtilen yolda bulunan dosyayı açar ve dosya nesnesi olarak programa döndürür.

open() fonksiyonu, dosya ismi ve mod (dosya açma biçimi) olarak iki parametre alır. Dosyayı açmak için 4 farklı mod kullanılır.

"r" - Okuma – Okumak için bir dosya açar. Dosya hedefte yoksa hata verir(Varsayılan).

"a" - Ekleme – Var olan dosyada düzenleme yapmak için açar. Dosya hedefte yoksa oluşturulur.

"w" - Yazma – Yazma modunda bir dosya açar. Dosya hedefte yoksa oluşturulur.

"x" - Oluştur – Belirtilen dosyayı oluşturur. Dosya varsa hata döndürür.

read() fonksiyonu, açılmış dosyanın içeriğini tek bir metin bilgisi olarak okur.

Örnek 4:

http://kitap.eba.gov.tr/KodSor.php?KOD=22377

```
Yimport os
os.chdir('C:\\test')

dosya = open("milli.txt")
print(dosya.read())
```

Çıktı:

```
Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.
```

readlines() fonksiyonu, içeriği satırlara ayırır ve satırlardan oluşmuş bir liste döndürür.

Örnek 5:

http://kitap.eba.gov.tr/KodSor.php?KOD=22377

```
import os
os.chdir('C:\\test')

dosya = open("milli.txt")
print(dosya.readlines())
```

Çıktı:

```
['Korkma, sönmez bu şafaklarda yüzen al sancak; \n', 'Sönmeden yurdumun üstünde tüten en son ocak.\n', 'O benim milletimin yıldızıdır, parlayacak;\n', 'O benimdir, o benim milletimindir ancak.']
```

readline() fonksiyonu, mevcut satırı okur ve her çağrıldığında bir sonraki satırı getirir.

Örnek 6:

```
import os
os.chdir('C:\\test')

dosya = open("milli.txt")
print(dosya.readline())
print(dosya.readline())
```

Çıktı:

```
Korkma, sönmez bu şafaklarda yüzen al sancak;

Sönmeden yurdumun üstünde tüten en son ocak.
```

readlines() veya readline() fonksiyonlarını döngü ile beraber kullanarak dosyalar satır satır işlenebilir. Açılan her dosya belleğe atılır ve hafızada yer kaplar. Bu nedenle verimli bir bellek yönetimi için dosyalar çalışma sonunda close() fonksiyonu kullanılarak kapatılmalı ve bellekten kaldırılmalıdır.

Örnek 7:

```
import os
os.chdir('C:\\test')

dosya = open("milli.txt")
for satir in dosya:
    print(satir.upper(),end="")
dosya.close()
```

Çıktı:

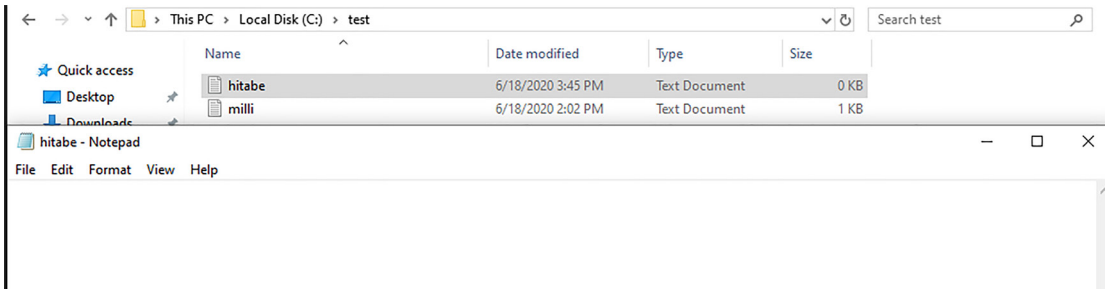
```
KORKMA, SÖNMEZ BU ŞAFAKLARDA YÜZEN AL SANCAK;
SÖNMEDEN YURDUMUN ÜSTÜNDE TÜTEN EN SON OCAK.
O BENIM MILLETIMIN YILDIZIDIR, PARLAYACAK;
O BENIMDIR, O BENIM MILLETIMINDIR ANCAK.
```

9.2. Dosya Oluşturma ve Yazma

Dosya açmak için kullanılan open() fonksiyonu "w" (write) parametresi ile beraber kullanılırsa dosya yazma modunda açılır ve veri yazımına hazır hâle gelir.

```
dosya = open("hitabe.txt", "w")
```

satırı dosyanın mevcut olması durumunda önceki içeriği siler ve hitabe.txt dosyasını yazma modunda açar. Eğer böyle bir dosya bulunmuyorsa dizinde bu isimle boş bir dosya oluşturur.



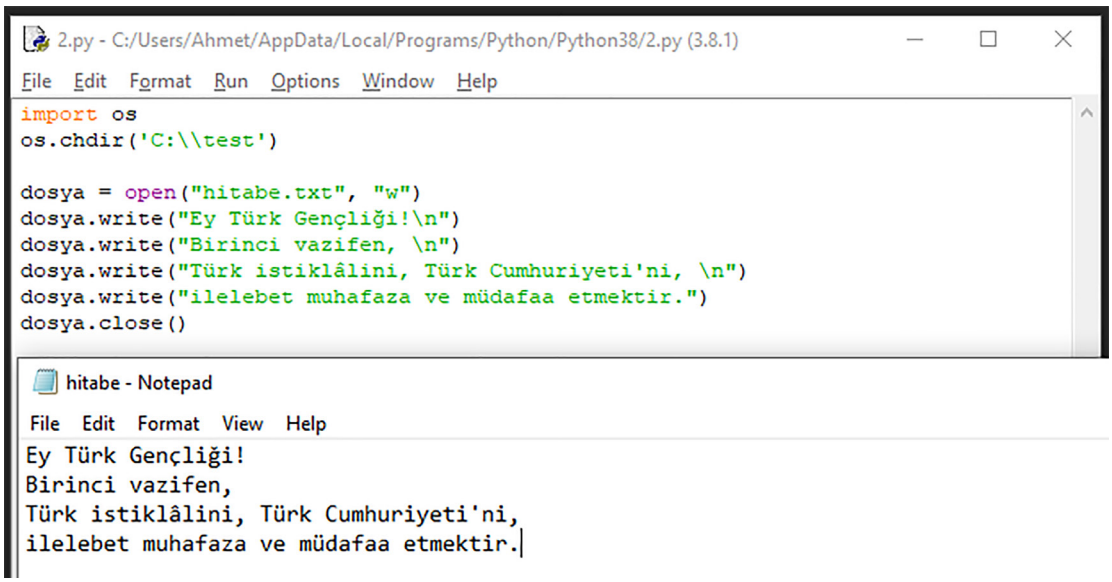
Görsel 9.2: Oluşturulan boş hitabe.txt dosyası

Açılan dosyaya veri yazdırmak için write() fonksiyonu kullanılır.

Örnek 8:

```
import os
os.chdir('C:\\test')

dosya = open("hitabe.txt", "w")
dosya.write("Ey Türk Gençliği!\n")
dosya.write("Birinci vazifen, \n")
dosya.write("Türk istiklâlini, Türk Cumhuriyeti'ni, \n")
dosya.write("ilelebet muhafaza ve müdafaa etmektir.")
dosya.close()
```

Kod bloğu çalıştırıldığında hitabe.txt dosyasının içeriğine "Gençliğe Hitabe"nin dört satırının yazıldığı görülecektir.



Görsel 9.3: hitabe.txt dosyası

Dosya write (yazma) modunda açıldığından dosyanın içeriğini okumak için kapatılıp tekrar açılması gerekmektedir. Dosya kapatılmadan dosyanın içeriği okunmak istenirse program hata verecektir.

Dosya içeriğine mevcut veriyi silmeden ekleme yapmak için dosyanın "a" parametresi kullanılarak append (ekle) modunda açılması gerekmektedir. Bu modda açılan dosyaya eklediğiniz her yeni satır son verinin altına yazılacaktır.

Örnek 9:

```
import os
import datetime
os.chdir('C:\\test')

dosya = open("tarih.txt", "w")
tarih = datetime.date.today()
dosya.write(str(tarih) +"\n")
dosya.close()

dosya = open("tarih.txt")
print(dosya.read())
dosya.close()

dosya = open("tarih.txt", "a")
saat = datetime.datetime.now().time()
dosya.write(str(saat) +"\n")
dosya.close()

dosya = open("tarih.txt")
print(dosya.read())
dosya.close()
```

Çıktı:

```
2020-06-18


2020-06-18
16:18:52.916795
```

Kod bloğunda yeni bir dosya oluşturulmuş, içeriğine tarih bilgisi içeren bir satır eklenmiştir. Daha sonra düzenlemek için açılan dosyaya saat bilgisi eklenmiş ve dosya okunup ekrana yazdırılmıştır.

Belirtilen konumda dosyanın var olup olmadığın kontrolü dizin örneğinde olduğu gibi exists() fonksiyonu ve isfile() fonksiyonları ile birlikte yapılabilir. İsfile() fonksiyonu, erişilen konumun dosya olup olmadığını kontrol eder.

Örnek 10:

```
import os
yol = os.path.join("C:\\","test","tarih.txt")
if os.path.exists(yol):
    print(yol + ' : konumu var.')
    if os.path.isfile(yol):
        print(yol + ' : bir dosya')
    else:
        print("bir dosya değil")
else:
    print("Yol bulunamadı!")
```

Çıktı:

```
C:\test\tarih.txt : konumu var.
C:\test\tarih.txt : bir dosya
```

9.3. Dosya Silme ve Yedekleme

Görsel 9.4: Yedekleme

Mevcut bir dosyayı silmek için os modülünün remove() fonksiyonu kullanılır. Aşağıdaki örnekte, belirtilen dosyanın var olup olmadığı kontrol edilerek dosya varsa silinmektedir.

Örnek 11:

```
import os
os.chdir('C:\\test')

if os.path.exists("tarih.txt"):
    print("Dosya mevcut, siliniyor...")
    os.remove("tarih.txt")
else:
  print("Dosya mevcut değil.")
```

Çıktı:

```
Dosya mevcut, siliniyor...
```

Kod çalıştırıldığında daha önce oluşturulan tarih.txt dosyası silinecektir. Kod bir daha çalıştırılırsa dosya silindiği için "Dosya mevcut değil." çıktısı verir.

Çalışılan dosyaların herhangi bir olumsuz duruma karşı yedeklenmesi gerekir. Bu sayede veri ve emek kaybının önüne geçilebilir. Python'da os modülü ile birlikte dosya kopyalama, taşıma ve üst düzey dosya işlemlerinde kullanabileceğiniz shutil adlı modül bulunmaktadır. Aşağıdaki örnekte os ve shutil modülleri kullanılarak daha önce oluşturulan test klasöründeki tarih.txt dosyası, yeni oluşturulacak yedek isimli klasöre tarihyedek.txt ismi ile kopyalanarak yedeklenecektir.

Örnek 12:

```
import os, shutil
os.chdir('C:\\test')

yedek_dizini = os.path.join("C:\\","yedek")
dosya = "tarih.txt"

if os.path.exists(yedek_dizini):
    print(yedek_dizini + ' : var')
    if os.path.isdir(yedek_dizini):
        print(yedek_dizini + ' : bir dizin')
else:
    print("Yedek dizini bulunamadı!")
    os.mkdir(yedek_dizini)
    print("oluşturuluyor...")

yedek_dosya = os.path.join(yedek_dizini,"tarihyedek.txt")

if os.path.exists(dosya):
    print("Dosya mevcut, yedekleniyor.")
    shutil.copy(dosya,yedek_dosya)

if os.path.exists(yedek_dosya):
    print("Dosya başarı ile yedeklendi.")
```

Çıktı:

```
Yedek dizini bulunamadı!
oluşturuluyor...
Dosya mevcut, yedekleniyor.
Dosya başarı ile yedeklendi.
```

ÖLÇME VE DEĞERLENDİRME 9

1. Verilen konumun var olup olmadığı bilgisini veren ..... veren fonksiyonu, dosya olup olmadığı bilgisini veren ise.. ..... fonksiyonudur.
2. Açılmış dosyalar üzerinde farklı modlarda işlem yapılabilmesi için önce kapatılmaları gerekir. (D/Y)
3. Aşağıdakilerden hangisi dosya sistemlerine erişmek için gerekli modüldür?

   A) os
   B) math
   C) for
   D) time
   E) timedelta

4. Verilen yolun klasör olup olmadığı bilgisini veren fonksiyon hangisidir?

   A) getcwd()
   B) isdir()
   C) exists()
   D) split()
   E) sin()

5. Aşağıdakilerden hangisi dosya işlemleri ile ilgili bir fonksiyon değildir?

   A) getcwd()
   B) join()
   C) mkdir()
   D) exists()
   E) print()

NOT: Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız.

KAYNAKÇA

Bilişim Teknolojileri Çerçeve Öğretim Programı

Algan, S. (2008). Her Yönüyle C#. İstanbul: Pusula Yayıncılık.

El Harezmi. (2014, 17 Mayıs). Erişim adresi: http://matematik.dpu.edu.tr/index/sayfa/3119/el-harezmi

Hunt, J. (2019). A Beginners Guide to Python 3 Programming. Springer.

Python Documentation. (2020, 14 Nisan). Erişim adresi: https://docs.python.org/3/

Su, G. (2018). Scratch İle Programlama. İstanbul: Kodlab.

Tanimoto, S. (2017). Game Design for Problem Solving with Python. Erişim adresi: https://courses.cs.washington.edu/courses/cse190d/17sp/materials/GDFPSP-ch1-3.pdf

Topal, A.D., Alkan, A. (2010). Mayer'in Bilimsel ve Matematiksel Mesaj Tasarım İlkelerine

Göre Tasarlanmış Öğrenme Ortamının Öğrenci Başarısı Üzerine Etkisi. Kocaeli Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 20(2), 93-106.

Türk Dil Kurumu Sözlükleri. (2020, 10 Mayıs). Erişim adresi: https://sozluk.gov.tr/

Türk Dil Kurumu. (2020, 10 Mayıs). Erişim adresi: https://www.tdk.gov.tr/

Scratch web sitesi. Erişim adresi: http://scratch.mit.edu/about, kaynağından Nisan 27, 2020 tarihinde erişilmiştir.

*Kaynakça kısmı APA6 referanslama sistemi kullanılarak oluşturulmuştur.

GÖRSEL KAYNAKÇALARI

GÖRSEL NO	ERİŞİM ADRESİ		ERİŞİM TARİHİ
ÖĞRENME BİRİMİ 1			
Öğrenme Birimi Kapak Resimleri	123rf.com	İd:45725682	
	123rf.com	İd:88225741	
	123rf.com	İd:111714055	
Görsel 1.1	https://www.shutterstock.com/	İd: 104723360	
Görsel 1.2	https://www.shutterstock.com/	İd: 599345642	
Görsel 1.3	https://courses.cs.washington.edu/courses/cse190d/17sp/materials/GDFPSP-ch1-3.pdf sitesindeki figure 1.4 resmi ile https://www.shutterstock.com/ sitesindeki id: 320117219 Resimlerinin birleştirilmesi ile elde edilmiştir.		30.04.2020
Görsel 1.4	https://www.cs.princeton.edu/courses/archive/spring18/cos226/assignments/8puzzle/index.html		15.04.2020
Görsel 1.5	https://www.shutterstock.com/	İd: 1135956632	
Görsel 1.6	https://www.shutterstock.com/	İd: 127871471	
ÖĞRENME BİRİMİ 2			
Öğrenme Birimi Kapak Resimleri	123rf.com	İd:145813081	
	https://www.shutterstock.com/	İd:726501535	
	123rf.com	İd:86327943	
ÖĞRENME BİRİMİ 3			
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd:604363835	
	https://www.shutterstock.com/	İd:1646703295	
	https://www.shutterstock.com/	İd:164090489	
Görsel 3.1	https://www.shutterstock.com/	İd: 772227652	
Görsel 3.2	https://www.shutterstock.com/	İd: 701798683	
Görsel 3.3	https://www.shutterstock.com/	İd: 288042338	
Görsel 3.4	https://www.tiobe.com/tiobe-index/)		14.04.2020
	ÖĞRENME BİRİMİ 4		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd:1681185799	
	https://www.shutterstock.com/	İd:1188132619	
		İd:649380919	
Görsel 4.1	https://www.shutterstock.com/	İd:516701455	
Görsel 4.2	123rf.com	image ID: 89201019- resmi- Komisyon- Görsel- Tasarım- uzmanı- tarafından- düzenerek- kullanılmıştır.	

	ÖĞRENME BİRİMİ 5		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd:1673512366	
	https://www.shutterstock.com/	İd: 708208342	
	https://www.shutterstock.com/	İd:424439902	
Görsel 5.1	https://www.shutterstock.com/	İd: 1188320323	
	ÖĞRENME BİRİMİ 6		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd:1478984780	
	https://www.shutterstock.com/	İd:39409432	
	123rf	İd:82427415	
Görsel 6.2	https://www.shutterstock.com/	İd: 771551869	
	ÖĞRENME BİRİMİ 7		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd: 766847608	
	https://www.shutterstock.com/	İd:1388181686	
		İd:442939405	
Görsel 7.2	https://www.shutterstock.com/	İd: 316441487	
	ÖĞRENME BİRİMİ 8		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd:1147086383	
	https://www.shutterstock.com/	İd:173105192	
	https://www.shutterstock.com/	İd:1728589483	
	ÖĞRENME BİRİMİ 9		
Öğrenme Birimi Kapak Resimleri	https://www.shutterstock.com/	İd: 279072458	
	https://www.shutterstock.com/	İd.:543238333	
	https://www.shutterstock.com/	İd:116342143	
Görsel 9.2	https://www.shutterstock.com/	İd: 730338076	
Kitap Kapağı Görseli	Kitap kapağı görseli grafik tasarımcı tarafından tasarlanmıştır.		

ÖĞRENME BİRİMLERİ ÖLÇME VE DEĞERLENDİRME CEVAP ANAHTARLARI

ÖĞRENME BİRİMİ 1'İN CEVAP ANAHTARI

1. E
2. B
3. B
4. D
5. Adım1: başla, Adım 2: tencereye su koy, Adım 3: tuz ekle, Adım 4: suyu kaynat, Adım 5: makarnayı ekle, Adım 6: makarnayı pişir, Adım 7: makarnanın suyunu süz, Adım 8: bitir

ÖĞRENME BİRİMİ 2'NİN CEVAP ANAHTARI

1. E
2. C
3. A
4. D
5. A
6. E

ÖĞRENME BİRİMİ 3'ÜN CEVAP ANAHTARI

1. Hayır. Çünkü; Python programlarını çalıştırabilmek için bir işletim sistemi gereklidir. Dolayısıyla Python ile işletim sistemi yazılamaz.
2. En son versiyonu kullanmak gerekliliği ile ilgili şunlar söylenebilir.
   Yeni işlev ve özellikler gelmiştir.
   Hatalardan arındırılmıştır.
   Daha az hafıza ve işlemci gücü gereksinimi vardır.
3. Şu IDE'ler tercih edilebilir.
   PyCharm: Dünyanın en çok kullanılan IDE'si olduğu için tercih edilebilir.
   Spyder: Açık kaynak kod olduğu için tercih edilebilir.
   Visual Studio Code: Sadece Python'a değil, birçok dile destek verdiği için tercih edilebilir.
4. Komutların işlevleri:
   I. 10 + 20 : İki sayıyı toplar.
   II. 4 * 30 : İki sayıyı çarpar.
   III. 2 ** 1000 : 2 üzeri 1000'i hesaplar.
   IV. print("merhaba") : Ekrana "merhaba" yazdırır.
   V. 36 / 4 * (3 + 2) * 4 + 2 : İşlem önceliğine göre hesaplama yapar.
5. Evet.

ÖĞRENME BİRİMİ 4' ÜN CEVAP ANAHTARI

1. B
2. E
3. C
4. A
5. E
6. B
7. C
8. A
9. D
10. D

ÖĞRENME BİRİMİ 5'İN CEVAP ANAHTARI

1. B
2. True
3. Break
4. Random
5. Range
6. D
7. B
8. C
9. A
10. E

ÖĞRENME BİRİMİ 6'NIN CEVAP ANAHTARI

1 – Modül
2 – def kare_al(sayi) :
3 – Özyinelemeli fonksiyon
4 – D
5 – C
6 – E
7 – B
8 – A

ÖĞRENME BİRİMİ 7'NİN CEVAP ANAHTARI

1 – C
2 – D
3 – C
4 – find()
5 – B

ÖĞRENME BİRİMİ 8'İN CEVAP ANAHTARI

1. İstenildiği kadar kullanılabilir. Her hata için birden fazla except bloğu da kullanılabilir. Sadece try-finally şeklindeki kullanımlarda exept bloğu kullanılmayabilir.
2. Evet geçerli bir kullanımdır.
3. Evet yakalanabilir.

```
try:
  pass
except ValueError:
  pass
except (TypeError, ZeroDivisionError):
  pass
except:
  # handle all other exceptions
  pass
```

4. C) 2
5. B) False
6.

```
print("Program başladı") # Ekrana "Program başladı" yazar.
try: # try bloğu başlangıcı
  raise Exception('Bir hata oluştu!') # kodla hata fırlatılır.
except: # fırlatılan hata burada yakalanır.
  print("Except bloğuna geldik.") # ekrana "Except bloğuna geldik" yazar.
  raise # hata tekrar fırlatılır.
finally: # bu kod mutlaka çalıştırılır.
  print("Finally bloğuna geldik.") # ekrana "Finally bloğuna geldik" yazar.
print("Program bitti") # Bu satır çalışmaz. Çünkü raise ifadesi ile hata tekrar fırlatılır ve program yarıda kesilir.
```

ÖĞRENME BİRİMİ 9' UN CEVAP ANAHTARI

1. exists() , isfile()
2. Doğru
3. A
4. B
5. E