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

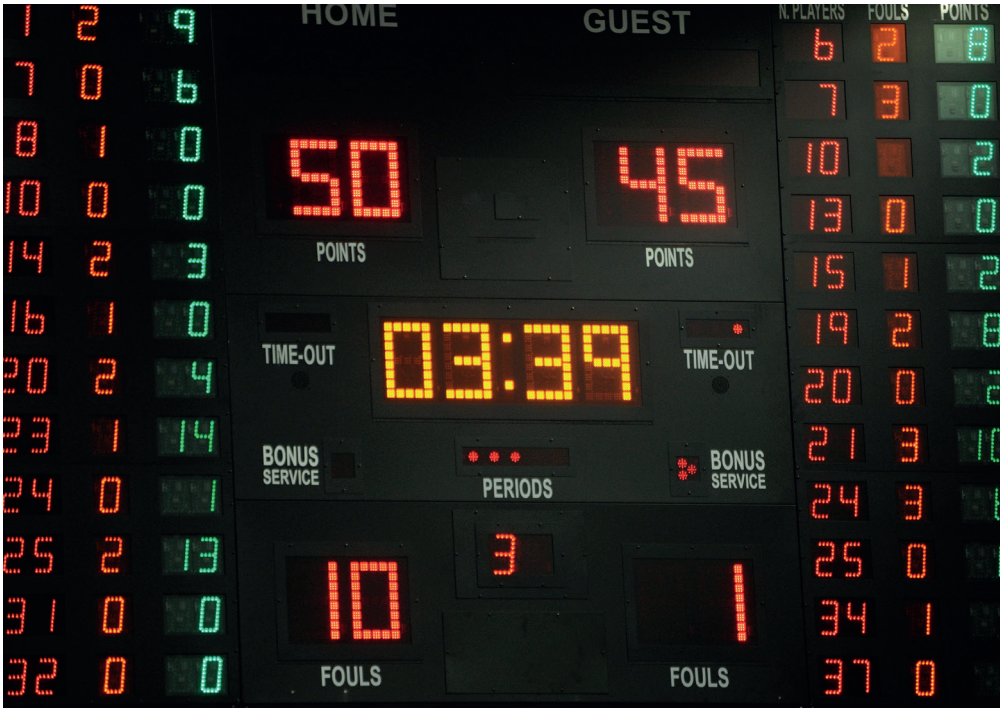
1. Değişken kavramını araştırıp günlük hayattan örnekler vererek arkadaşlarınızla tartışınız.
2. Python programlama dilinde kullanılan operatörleri araştırarak matematik dersinde kullanılan operatörlerle karşılaştırınız.

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.

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			

boşluk olmaz
sayı ile başlamaz

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

7^2=49

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	32
5//2	2
4+3	7
11%3	2
5/2	2.5
2*5	10
3-5	-2
(4-1)**2	9
(7//3)/2	1

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 :

```
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)	5
a=6 a*=3 print(a)	18
a=5 a**=3 print(a)	125
a=11 a%=3 a**=3 print(a)	8

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 :

```
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.

```
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. 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. 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 çalıştınız.

```
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.

```
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)
```

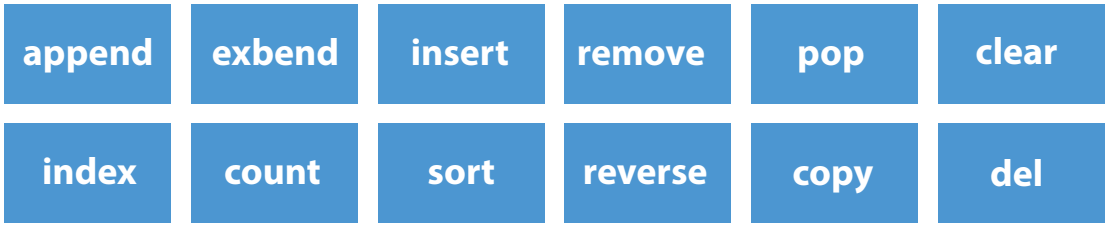
Ekran Çıktısı:

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).



Şekil1.1. *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:** indeksi 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.

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

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

   f) Listenin tüm elemanlarını siliniz.

   g) 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.

   d) Listedeki 81 sayısını siliniz.

   e) Listenin tüm elemanlarını siliniz.

   f) 64 elemanının indeksini bulunuz.

   g) 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** [ ], tuple 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 **key** 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.

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

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

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

g) 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.

a) önemli_bilgiler isimli sözlüğü siliniz.

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

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

a) 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.