

# Python Eğitimi (Akış Diyagramları ve Algoritmalarla)

Eğitmen: Elif KARAKAŞ



Youtube: [Elif Hoca Robotik](#)  
Instagram: [Elif Hoca Robotik](#)  
Twitter: [Elif Hoca Robotik](#)  
Facebook: [Elif Hoca Robotik](#)  
LinkedIn: [Elif KARAKAŞ](#)  
İletişim: [bilgi@elifhocarobotik.com](mailto:bilgi@elifhocarobotik.com)

# İçindekiler

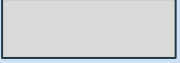
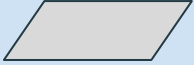
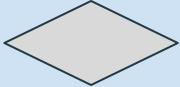
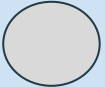

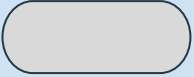

- İlk Python Programı ve Yorum Türleri
- Değişkenler ve Türleri
- Strings işlemleri
- Booleans
- Operatörler
- Listeler
- Koşullar
- Döngüler
- Sınıflar ve Nesneler
- Sözlükler
- Modüller ve Paketler



# Kontrol İfadeleri ve döngüler



# Akış Şeması Sembolleri / Flowchart Symbols /

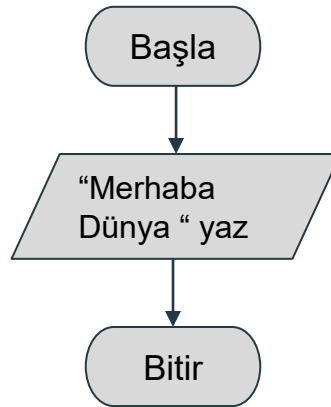
Sembol	İsim	Fonksiyon
	İşlem	İşlemci veya Bellek içindeki her türlü dahili işlemi gösterir
	Giriş / Çıkış	Herhangi bir Giriş / Çıkış (G / Ç) işlemi için kullanılır. Bilgisayarın girdi veya çıktı sonuçları elde edeceğini belirtir.
	Karar	İkili formatta cevaplanabilen bir soru sormak için kullanılır (Evet / Hayır, Doğru / Yanlış)
	Bağlayıcı	Akış şemasının kesişen çizgiler olmadan veya ters akış olmadan çizilmesine izin verir.
	Önden tanımlanmış işlem	Bir alt yordamı veya bir Kesme programını çağırmak için kullanılır.
	Terminal	Programın, işlemin veya kesintiye uğrayan programın başlangıcını veya bitişini gösterir
	Akış Çizgileri	Akış yönünü gösterir.

# İlk Python Programı çalıştırma

## PYTHON KODU

```
print("Merhaba Dünya")
```

## AKIŞ ÇİZELGESİ



## ALGORİTMA

1. Başla
2. Ekrana "Merhaba Dünya" Yaz
3. Bitir

# Python Yorum Türleri

Python'da iki tür yorum vardır.

## 1. Tek satırlı açıklama

```
# Bu sadece bir yorumdur. Burada yazılan her şey Python tarafından göz ardı edilir
```

## 2. Çok satırlı açıklama

```
'''
```

```
Burada basit bir program yazıyoruz
```

```
Bu çok satırlı bir yorumdur.
```

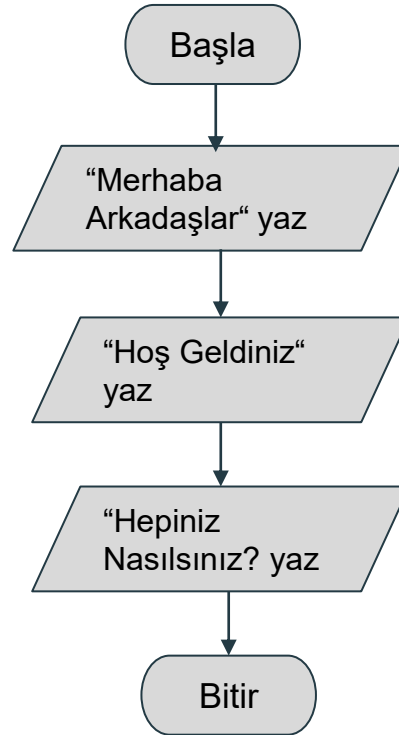
```
'''
```

# İlk Python Programı çalıştırma

## PYTHON KODU

```
print ( "Merhaba Arkadaşlar" )  
print ( "Hoş Geldiniz" )  
print ( "Hepiniz Nasılsınız?" )
```

## AKIŞ ÇİZELGESİ



## ALGORİTMA

1. Başla
2. Ekrana "Merhaba Dünya" Yaz
3. Ekrana "Hoş Geldiniz" Yaz
4. Ekrana "Hepiniz Nasılsınız?" Yaz
5. Bitir



# Python if - else kullanımı

Bir sayının negatif olup olmadığını bulma

## PYTHON KODU

```
sayi = -22
```

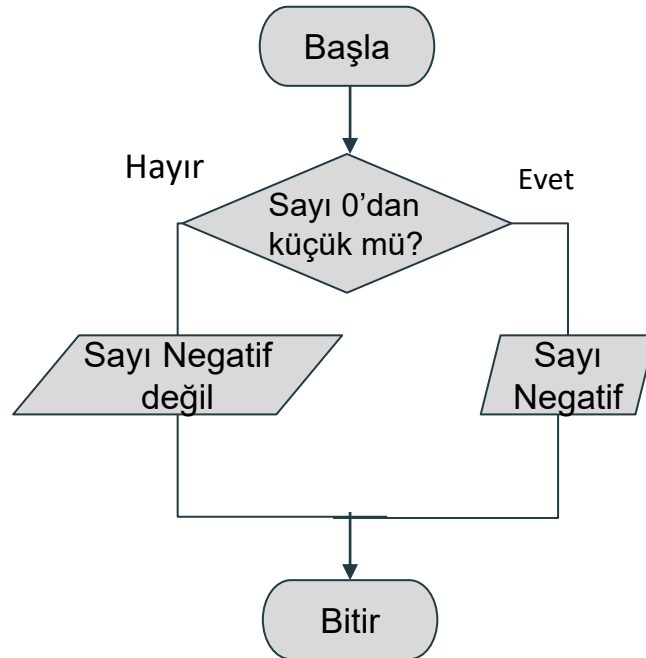
```
if sayi < 0:
```

```
    print("Sayı Negatif")
```

```
else:
```

```
    print("Sayı Negatif değil")
```

## AKIŞ ÇİZELGESİ



## ALGORİTMA

1. Başla
2. Sayı 0'dan küçük ise "Sayı Negatif " yazdır
3. Sayı 0'dan küçük değilse "Sayı Negatif değil" yazdır
4. Bitir

# Python If - elif - else kullanımı

Bir sayının negatif / pozitif yada nötr olup olmadığını bulma

## PYTHON KODU

```
sayi = 0
```

```
if sayi < 0:
```

```
    print("Sayı Negatif")
```

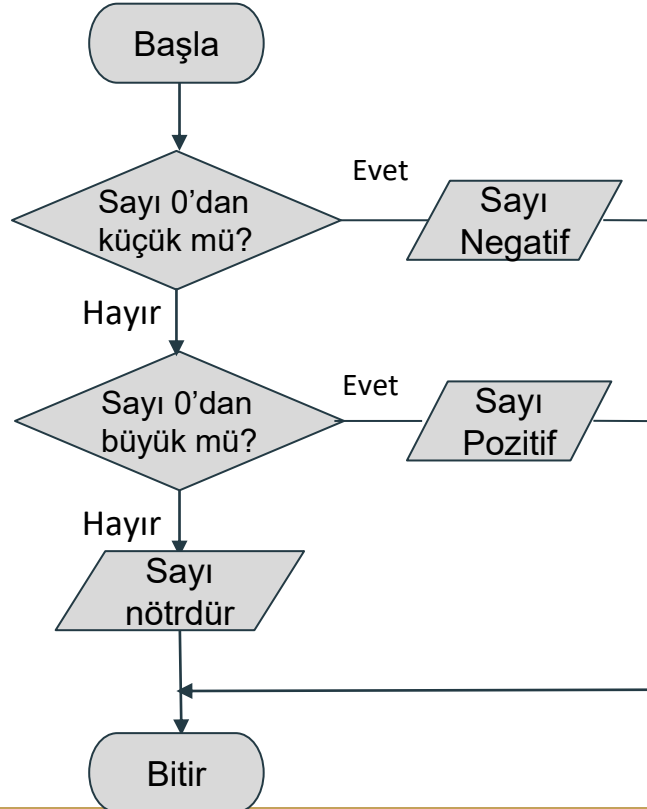
```
elif sayi > 0:
```

```
    print("Sayı pozitif")
```

```
else:
```

```
    print("Sayı Nötrdür")
```

## AKIŞ ÇİZELGESİ



## ALGORİTMA

1. Başla
2. Sayı 0'dan küçük ise "Sayı Negatif" yazdır
3. Sayı 0'dan büyük ise "Sayı pozitif" yazdır
4. Her ikisi de değilse "Sayı nötrdür" yazdır
5. Bitir

# Python while loop kullanımı

Klavyeden girilen sayı 0 değil İKEN; klavyeden girilen sayıları çarpma

## AKIŞ ÇİZELGESİ

### PYTHON KODU

```
i = 1  
carp=1  
while i!=0:  
    carp *=i  
    print(carp)  
    i=int(input("Sayı gir"))
```



### ALGORİTMA

1. Başla
2. Sayı 0 değil iken Sayıyı çarp, ekrana yazdır ve yeni sayı sor
3. Sayı 0 ise While Loop'dan çık
4. Bitir

# Python for loop kullanımı

4' den başlayarak 55'e kadar, sayıları 10'ar 10'ar arttırarak yazdırma

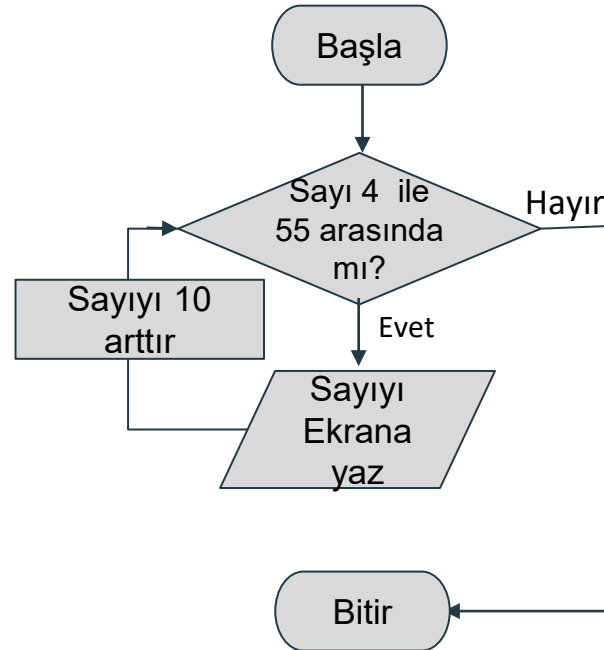
## PYTHON KODU

```
for x in range(4,55,10):
```

```
    print(x)
```

Sıra	Sonuç
0. sıra	4
1.sıra	14
2.sıra	24
3.sıra	34
4.sıra	44
5.sıra	54

## AKIŞ ÇİZELGESİ



## ALGORİTMA

1. Başla
2. Sayı 4 ile 55 arasındaysa Sayıyı ekrana yazdır ve sayıyı 10 arttır.
3. Değilse For Loop'dan çık
4. Bitir

# Python for loop break / continue kullanımı

Durum 1: Listede “x” harfi bulursan döngüden çık (break)/ Durum 2: “x” harfi bulursan o sırayı atla (continue)

## PYTHON KODU

```
metin= "Merhaba, x iyi misin?"
```

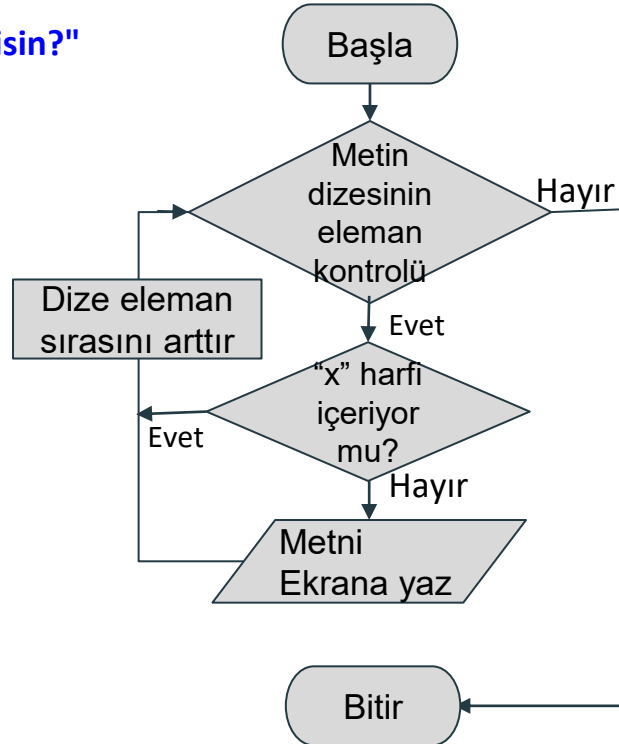
```
for x in metin:
```

```
    if x == "x":
```

```
        continue
```

```
    print(x)
```

## AKIŞ ÇİZELGESİ CONTIUNE



## ALGORİTMA

1. Başla
2. metin elemanlarını kontrol et,
3. Metnin elemanı “ x” mi evetse ekrana yazdırma
4. Metin elemanı “ x” değilse ekrana yazdır
5. Eleman sırasını arttır ve kontrol bitene kadar devam et
6. Bitir

# Python for loop break / continue kullanımı

Durum 1: Listedeki "x" harfi bulursan döngüden çık (break)/ Durum 2: "x" harfi bulursan o sırayı atla (continue)

## PYTHON KODU

```
metin = "Merhaba, x iyi misin?"
```

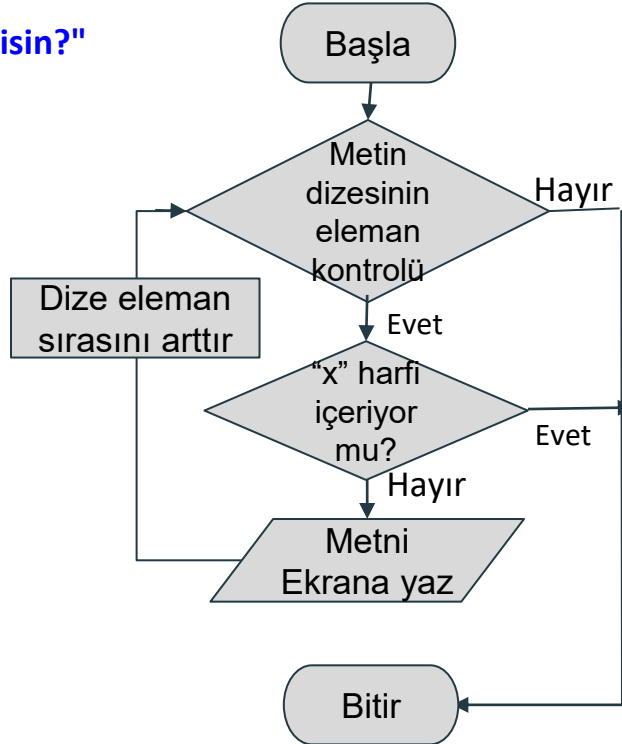
```
for x in metin:
```

```
    if x == "x":
```

```
        break
```

```
    print(x)
```

## AKIŞ ÇİZELGESİ BREAK



## ALGORİTMA

1. Başla
2. Metin elemanlarını kontrol et,
3. Metnin elemanı "x" mi evetse ekrana yazdırma ve döngüden çık . maddeye git
4. Metin elemanı "x" değilse ekrana yazdır
5. Eleman sırasını artır ve kontrol bitene kadar devam et
6. Bitir



# Python Booleans



# Python Booleans

## Boole Değerleri

Programlamada genellikle bir ifadenin `True` ( Doğru ) veya olup olmadığını bilmeniz gerekir

`False` (yanlış).

```
print(10 > 9) -> true
print(10 == 9) - > false
print(10 < 9) -> false
```

Bir metni yada sayıyı bool ile de kontrol edebiliriz

```
print(bool("Hello"))
print(bool(15))
```

Class yada fonksiyon gibi yapılarda return komutu `True` - `False` değeri döndürebilir

```
def fonksiyon() :
    return True
```

```
print(fonksiyon())
```

Bir nesnenin tam sayı olup olmadığını kontrol edin:

```
x = 200
print(isinstance(x, int))
```





# Değişkenler ve Veri Türleri



# Programlamada değişken | Python Veri Türleri

Bir işlemi gerçekleştirmek için yapılması gereken ilk şey o **veriyi hafızaya almak** ve istediğimizde de **veriyi hafızadan çağırıp** gerekli işlemleri yerine getirmektir.

Hafızadaki verileri ifade etmek için programlama dillerinde **değişkenleri** kullanırız.

Özetle; programlama dilinde işlediğimiz verileri bilgisayarın hafızasında tutmak için yapmış olduğumuz tanımlamalardır.

Metin Türü:	<code>str</code>
Sayısal Türler:	<code>int, float, complex</code>
Sıra Türleri:	<code>list, tuple, range</code>
Eşleme Türü:	<code>dict</code>
Set Türleri:	<code>set, frozenset</code>
Boole Türü:	<code>bool</code>
İkili Türler:	<code>bytes, bytearray, memoryview</code>

# Python Değişkenleri Tür Tablosu

Örnek	Veri Türü
<code>x = "Merhaba"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["elma", "armut", "kivi"]</code>	<code>list</code>
<code>x = ("elma", "armut", "kivi")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"ad" : "erdem", "yas" : 13}</code>	<code>dict</code>
<code>x = {"elma", "armut", "kivi"}</code>	<code>set</code>
<code>x = frozenset({"elma", "armut", "kivi"})</code>	<code>frozenset</code>
<code>x = True</code>	<code>bool</code>
<code>x = b"MErhaba"</code>	<code>bytes</code>
<code>x = bytearray(5)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(5))</code>	<code>memoryview</code>

# Değişkenleri Kullanımı / Türünü Öğrenme / Belirtme

Python'un bir **değişken** bildirmek için bir **komutu** yoktur. Bir değişken, ona **bir değer** atadığınız **anda** oluşturulur.

```
tamsayi= 100
ondaliklisayi= 1.5
metin= "Herkes merhaba"

print ( tamsayi)
print ( ondaliklisayi)
print ( metin)

print (type(sayi))
print (type(metin))
print (type(ondaliklisayi))
```

Bazen değişkenin veri tipini belirtmek gerekebilir, bunu çevrim ile yapabilirsiniz..

```
degisken1 = str("Elif Hoca")
degisken2 = int("25")
degisken3 = float("1.25")

print (type(degisken1))
print (type(degisken2))
print (type(degisken3))
```

# Python Çoklu Değişken işlemleri

Birden Çok Değişkene Birçok Değer atama

```
x, y, z = "Elma", "Armut", "Portakal"  
print(x)  
print(y)  
print(z)
```

Birden Çok Değişkene Bir Değer atama

```
x = y = z = "Elma"  
print(x)  
print(y)  
print(z)
```

Bir liste, tuple vb. Bir değer koleksiyonunuz varsa Python, değerleri değişkenlere çıkarmanıza izin verir. Buna paketten çıkarma denir

```
meyveler= "Elma", "Armut", "Portakal"  
x, y, z = meyveler  
print(x)  
print(y)  
print(z)
```

# Dip Not / Rastgele sayı üretme

```
import random  
  
rastgelesayi=random.randrange(1, 10)  
  
print(rastgelesayi)
```

Sonuç her çalıştırmada 1 ile 10 arasında farklı bir değer çıkar

# Dip Not / Değişkenleri birleştirip yazma

```
x, y, z = "Elma", "Armut", "Portakal"
```

```
print(x + y + z)
```

```
print("Birinci eleman:" + x + "İkinci  
eleman:" + y + "Üçüncü eleman:" + z)
```

```
print(x,y,z)
```

```
print("Birinci eleman:",x,"İkinci  
eleman:",y,"Üçüncü eleman:", z)
```



# Python Kullanıcı Girişi



# Python Kullanıcı Girişi

Kullanıcı adı soran kod

```
username = input("Kullanıcı adınız nedir?")  
print("Kullanıcı adın: " + username)
```

Kullanıcıdan alınan girdi sayısal işlem yaparken mutlaka dönüştürülmelidir.

```
sayi1= input("Sayi biri giriniz")  
sayi2=input("Sayi ikiyi giriniz")  
print(sayi1+sayi2)
```

```
sayi1= int(input("Sayi biri giriniz"))  
sayi2=float(input("Sayi ikiyi giriniz"))  
print(sayi1+sayi2)
```





# PYTHON OPERATÖRLER



# Python Aritmetik Operatörleri

Operatör	Adı	Kullanımı
+	Toplama	$x + y$
-	Çıkarma	$x - y$
*	Çarpma	$x * y$
/	Bölme	$x / y$
%	Mod	$x \% y$
**	Üs alma	$x ** y$
//	Floor division	$x // y$

Aritmetik operatörler, yaygın matematiksel işlemleri gerçekleştirmek için sayısal değerlerle kullanılır:

`x = 5`

`y = 3`

`print(x + y)`

Kullanıcıdan girilen sayılara işlemleri uygulayalım

# Python Karşılaştırma Operatörleri

Operator	Örnek	İfadenin açılımı
==	Eşit	$x == y$
!=	Eşit değil	$x != y$
>	Büyük	$x > y$
<	Küçük	$x < y$
>=	Büyük eşit	$x >= y$
<=	Küçük eşit	$x <= y$

Karşılaştırma operatörleri iki değeri karşılaştırmak için kullanılır

$x = 5$

$y = 3$

`print(x == y)`

İki sayıyı karşılaştırma

```
a = 200
b = 33
if b > a:
    print("b, a dan büyük")
elif a == b:
    print("b ve a birbirlerine eşit")
else:
    print("a, b den büyük")
```

Klavyeden girilen iki sayıyı karşılaştırma

# Python Atama Operatörleri

Operator	Örnek	İfadenin açılımı
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x  = 3	x = x   3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Atama operatörleri, değişkenlere değer atamak için kullanılır:

```
x = 5
x += 3
print(x)
```

0 dan 7 'ye kadar sayıları ekrana yazdırın

```
i = 0
while i < 7:
    print(i)
    i += 1
```

Yukarıdaki listede 3 ü görünce dur

```
i = 0
while i < 6:
    print(i)
    i += 1
    if (i == 3):
        break
```

Yukarıdaki listede 3 ü görünce atla

```
i = 0
while i < 6:
    print(i)
    i += 1
    if i == 3:
        continue
```

# Python Mantıksal Operatörleri

Mantıksal operatörler, koşullu ifadeleri birleştirmek için kullanılır:

Operator	Örnek	İfadenin açılımı
and	Her iki ifade de doğruysa True döndürür	$x < 5$ and $x < 10$
or	İfadelerden biri doğruysa True döndürür	$x < 5$ or $x < 4$
not	Sonucu ters çevir, sonuç doğruysa False döndürür	not( $x < 5$ and $x < 10$ )

```
x = 5
```

```
print(x > 3 and x < 10)
```

```
a = 200
```

```
b = 33
```

```
c = 500
```

```
if a > b and c > a:
```

```
    print("her iki durumda doğru")
```

```
a = 200
```

```
b = 33
```

```
c = 500
```

```
if a > b or a > c:
```

```
    print("koşullardan en az biri doğru")
```

# Klavyeden girilen üç sayının en büyüğünü bulma

```
sayi1 = int(input("1. Sayı: "))
```

```
sayi2 = int(input("2. Sayı: "))
```

```
sayi3 = int(input("3. Sayı: "))
```

```
if (sayi1 >= sayi2) and (sayi1 >= sayi3):
```

```
    print("Sayıların en büyük olanı",sayi1)
```

```
elif (sayi2 >= sayi1) and (sayi2 >= sayi3):
```

```
    print("Sayıların en büyük olanı",sayi2)
```

```
else:
```

```
    print("Sayıların en büyük olanı",sayi3)
```

Klavyeden Girilen üç sayıyı büyükten küçüğe  
sıralayalım

# Python Kimlik Operatörleri

Kimlik işlemleri, nesneleri karşılaştırmak için kullanılır, eğer eşitse değil, ama aslında aynı nesneyse, aynı bellek konumuna sahipse:

Operator	Örnek	İfadenin açılımı
is	Her iki değişken de aynı nesneyse True döndürür	x is y
is not	Her iki değişken de aynı nesne değilse True döndürür	x is not y

```
x = ["elma", "muz", "armut"]
y = ["elma", "muz", "armut"]
z = x

if z is x:
    print("dizinler birbirleri ile aynı" )

print(x is z) # True döndürür çünkü z, x ile aynı nesnedir
print(x is y) # Yanlış döndürür çünkü x, aynı içeriğe sahip olsalar bile, y ile aynı nesne değildir
print(x == y) # "eşittir" ve "==" arasındaki farkı göstermek için: bu karşılaştırma True döndürür çünkü x, y'ye eşittir
```



# Python Üyelik Operatörleri

Üyelik operatörleri, bir dizinin bir nesnede sunulup sunulmadığını test etmek için kullanılır:

Operator	Örnek	İfadenin açılımı
in	Nesnede belirtilen değere sahip bir sıra varsa True döndürür	x in y
not in	Nesnede belirtilen değere sahip bir sıra yoksa True döndürür	x in not y

```
x = ["elma", "muz", "armut"]
```

```
print("elma" in x)
```

```
if "elma" in x:
```

```
    print("Dizinde elma bulundu" )
```

```
for x in "robotik":
```

```
    print(x)
```



# Strings



# Python Strings - 1

Python'daki dizeler tek tırnak işaretleri veya çift tırnak işaretleri ile çevrilidir.

```
print("Merhaba")  
print('Merhaba')
```

Bir değişkene bir dize atamak, değişken adı ve ardından bir eşittir işareti ve dize ile yapılır:

```
a = "Merhaba"  
print(a)
```

Üç tırnak işareti kullanarak bir değişkene çok satırlı bir dize atayabilirsiniz:

```
a = """Merhaba Ben Elif Karakaş  
Bilgisayar ve Öğretim Teknolojileri Öğretmeniyim.  
Elif Hoca Robotik Youtube Kanalının sahibiyim."""  
print(a)
```

# Python Strings - 2

1. konumdaki karakteri alın (ilk karakterin 0 konumuna sahip olduğunu unutmayın):

```
a = "Merhaba Dünya"  
print(a[1])
```

Dizeler diziler olduğundan, bir dizedeki karakterler arasında bir `fordöngü` ile döngü yapabilir

```
for x in "robotik":  
    print(x)
```

Bir dizenin uzunluğunu bulmak için

```
a = "Merhaba Dünya"  
print(len(a))
```

Bir dizede belirli bir kelime öbeği veya karakter olup olmadığını kontrol etmek için anahtar kelimeyi kullanabiliriz

```
a = "Python Dersine Hoş geldiniz"  
print("Hoş" in a)    -----    print("Hoş" not in a)
```

Bir dizede belirli bir kelime öbeği veya karakter olup olmadığını kontrol etmek için anahtar kelimeyi kullanabiliriz

```
a = "Python Dersine Hoş geldiniz"  
if "Hoş" in a:  
    print("Hoş kelimesi bulundu" )
```

# Python Strings - 3

Karakterleri baştan 5. konuma alın ( 5 dahil değildir)

```
a = "Merhaba Dünya"  
print(a[:5])
```

Karakterleri 2. pozisyonundan sonuna kadar alın:

```
print(a[2:])
```

Dilimi dizinin sonundan başlatmak için negatif dizinler kullanın:

```
a = "Merhaba Dünya"  
print(a[-5:-2])
```

Büyük harf dönüştür

```
print(a.upper())
```

Böler

```
print(a.split())
```

Küçük harf dönüştür

```
print(a.lower())
```

Boşluğu kaldır

```
print(a.strip())
```

Dizeyi Değiştir

```
print(a.replace("H", "J"))
```

# Python Strings - 4 - Dize Biçim

Python Değişkenleri bölümünde öğrendiğimiz gibi, dizeleri ve sayıları şu şekilde birleştiremeyiz

```
yas = 45
metin = "Merhaba, benim adım Erdem, benim yaşıım " + yas
print(metin )
```

Ancak `format()` yöntemi kullanarak dizeleri ve sayıları birleştirebiliriz !

```
yas = 45
metin = "Merhaba, benim adım Erdem, benim yaşıım {}"
print(metin.format(yas ) )
```

`format()` yöntemi istediğiniz sayıda kullanabilirsiniz.

```
kalite= "1.sınıf"
adet=2
birimfiyati=3.5
alisveris = "Ürün kalitesi {}, ürün adedi {}, birim fiyatı{} TL"
print(alisveris .format(kalite, adet,birimfiyati ) )
```

```
metin = "degisken icinde \" çift tırnak \" yazma."
print(metin)
```



# List, Tuple, Set ve Dictionary



# Python List, Tuple, Set ve Dictionary

List, Tuple, Set ve Dictionary, verileri verimli bir şekilde depolamak ve düzenlemek için kullanılan python'daki veri yapılarıdır.

- Liste , sıralı ve değiştirilebilir bir koleksiyondur. Yinelenen üyelere izin verir.
- Tuple ,sıralı ve değiştirilemeyen bir koleksiyondur. Yinelenen üyelere izin verir.
- Set , sıralanmamış ve dizine eklenmemiş bir koleksiyondur. Yinelenen üye yok.
- Sözlük , sıralanmamış ve değiştirilebilir bir koleksiyondur. Yinelenen üye yok.

Liste	Tuple	Set	Dictionary
Liste [] ile temsil edilebilir	Tuple () ile temsil edilebilir:	Küme, {} ile temsil edilebilir	Sözlük {} ile temsil edilebilir
Liste, yinelenen öğelere izin verir	Tuple, yinelenen öğelere izin verir	Set, yinelenen öğelere izin vermeyecek	Set, yinelenen öğelere izin vermez, ancak anahtarlar kopyalanmaz
Örnek: [1, 2, 3, 4, 5]	Örnek: (1, 2, 3, 4, 5)	Örnek: {1, 2, 3, 4, 5}	Örnek: {1, 2, 3, 4, 5}
List () işlevi kullanılarak liste oluşturulabilir	Tuple, tuple () işlevi kullanılarak oluşturulabilir .	Set, set () işlevi kullanılarak oluşturulabilir	Dictionary, dict () fonksiyonu kullanılarak oluşturulabilir .
Liste değiştirilebilir, yani listede herhangi bir değişiklik yapabiliriz.	Tuple değişmez, yani tuple'da herhangi bir değişiklik yapamıyoruz	Set değiştirilebilir, yani sette herhangi bir değişiklik yapabiliriz. Ancak öğeler kopyalanmaz.	Sözlük değiştirilebilir. Ancak Anahtarlar kopyalanmaz.
Boş bir liste oluşturmak l = []	Boş bir Tuple oluşturma t = ()	Bir set oluşturmak a = set ()	Boş bir sözlük oluşturmak d = {}





# List İşlemleri



# Python List - Tanımlama - Arama

## Liste iki farklı şekilde oluşturulabilir:

```
meyveler= ["elma", "armut", "kivi"] | meyveler = list(("elma", "armut", "kivi"))
```

## String, int ve boolean veri türünde veya karma türde listeler olabilir:

```
meyveler= ["elma", "armut", "kivi", "mango", "çilek"]
```

```
sayılar = [1, 5, 7, 9, 3]
```

```
durumlar= [True, False, False]
```

```
karma= ["abc", 34, True, 40, "erkek"]
```

### Liste Elemanlarını yazdırma

```
print(meyveler)
```

### Liste Uzunluğu

```
print(len(meyveler))
```

### Liste Türü

```
print(type(meyveler))
```

### Dizin aralığı

```
print(meyveler[1:3])
```

### Öğenin Mevcut olup olmadığını kontrolü

```
if "elma" in meyveler:
```

```
    print("Evet, 'elma' listenin içerisinde")
```

```
else:
```

```
    print("Hayır, 'elma' listenin içerisinde yok")
```

# Python List - Değiştirme

## Liste Tek Öğeyi Değiştir

```
meyveler= ["elma", "armut", "kivi",  
"mango", "çilek"]  
meyveler[1] = "ARMUT"  
print(meyveler)
```

---

## Liste Değer Aralığını Değiştir

```
meyveler= ["elma", "armut", "kivi",  
"mango", "çilek"]  
meyveler[1:3] = "ARMUT", "KİVİ"  
print(meyveler)
```

---

## Liste Yeni Öğe Ekle / Sona ekle

```
meyveler.append("portakal")
```

---

## Liste Yeni Öğe Ekle / İstediğin sıraya

```
meyveler.insert(2, "portakal")
```

---

## Başka bir listeden mevcut listeye ekleme

```
klasik = ["elma", "muz", "kiraz"]  
tropik = ["mango", "ananas", "papaya"]  
klasik.extend(tropik)  
print(klasik)
```

# Python List - Liste Öğelerini Kaldır

## Belirtilen Öğeyi Kaldır

```
meyveler.remove("elma")
```

## Belirtilen indeksi/sırayı kaldırır

```
meyveler.pop(1)
```

---

## Listenin içeriğini silme

```
meyveler.clear()
```

## Tüm elemanı siler

```
del meyveler[2]
```

---

## Tüm listeyi silin

```
del meyveler[2]
```

# Python Döngü Listeleri

## Liste ile for kullanımı

```
gunler= ["pazartesi", "salı", "çarşamba", "perşembe"]  
for x in gunler:  
    print(x)
```

---

## Liste Dizin Numaralarıyla Döngü

```
gunler= ["pazartesi", "salı", "çarşamba", "perşembe"]  
for i in range(len(gunler)):  
    print(gunler[i])
```

## Liste ile kısaltılmış for kullanımı

```
gunler= ["pazartesi", "salı", "çarşamba", "perşembe"]  
[print(x) for x in gunler]
```

---

## While ile list kullanımı

```
gunler= ["pazartesi", "salı", "çarşamba", "perşembe"]  
i = 0  
while i < len(gunler):  
    print(gunler[i])  
    i = i + 1
```

# Python List Kavrama

Mevcut bir listenin değerlerine göre yeni bir liste oluşturmak istediğinizde daha kısa bir sözdizimi sunar. Bir meyve listesine bağlı olarak, yalnızca adında "a" harfi bulunan meyveleri içeren yeni bir liste istiyorsunuz

## Liste ile for kullanımı - not in

```
meyveler = ["elma", "armut", "ayva", "kivi", "mango", "ahududu", "çilek", "muz"]
yeniliste = []
for x in meyveler:
    if "a" not in x:
        yeniliste.append(x)
print(yeniliste)
```

---

## Liste ile for kullanımı - in

```
meyveler = ["elma", "armut", "ayva", "kivi", "mango", "ahududu", "çilek", "muz"]
yeniliste = []
for x in meyveler:
    if "a" in x:
        yeniliste.append(x)
print(yeniliste)
```

---

## Liste ile for kullanımı - Kısaltılmış hali

```
meyveler = ["elma", "armut", "ayva", "kivi", "mango", "ahududu", "çilek", "muz"]
yeniliste = [x for x in meyveler if "a" in x]
print(yeniliste)
```

---

## Liste ile for kullanımı - Kısaltılmış hali

```
meyveler = ["elma", "armut", "ayva", "kivi", "mango", "ahududu", "çilek", "muz"]
yeniliste = [x for x in meyveler if x != "kivi"]
print(yeniliste)
```

# Python Listeleri sıralama

## Listeyi Küçükten Büyüğe sıralama

```
meyveler = ["elma", "Armut", "ayva", "Kivi", "mango", "Ahududu", "çilek", "Muz"]  
meyveler.sort()  
print(meyveler)
```

## Listeyi Büyükten Küçüğe sıralama

```
meyveler.sort(reverse = True)
```

# Python Liste Kopyalama

## Listeyi Kopyalama

```
yeniliste = meyveler.copy()
```

## Listeyi Kopyalama 2. Yöntem

```
yeniliste = list(meyveler)
```

# Python - Listeleri Birleştir

## Listeleri birleştirme Yönetem1

```
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
list3 = list1 + list2  
print(list3)
```

## Listeleri birleştirme Yönetem2

```
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
for x in list2:  
    list1.append(x)  
print(list1)
```

## Listeleri birleştirme Yönetem2

```
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
list1.extend(list2)  
print(list1)
```



# Tuple İşlemleri





# Python Tuple-

- [Tuple](#) ,sıralı ve değiştirilemeyen bir koleksiyondur. Yinelenen üyelere izin verir.

## Tuple iki farklı şekilde oluşturulabilir:

```
cicekler=('Lale','Gül','Papatya','Papatya') |  
cicekler =tuple(("Lale","Gül","Papatya","Papatya"))
```

## String, int ve boolean veri türünde veya karma türde listeler olabilir:

```
tuple1 = ("elma", "muz", "kiraz")  
tuple2 = (1, 5, 7, 9, 3)  
tuple3 = (True, False, False) tuple1 = ("abc", 34, True, 40, "merhaba")
```

### Tuple Elemanlarını yazdırma

```
print(cicekler)
```

### Tuple Eleman Uzunluğu

```
print(len(cicekler))
```

### Tuple Türü

```
print(type(cicekler))
```

### Dizin aralığı

```
print(cicekler[1:3])
```

### Öğenin Mevcut olup olmadığını kontrolü

```
if "Gül" in cicekler:  
    print("Evet, 'Gül' listenin içerisinde")  
else:  
    print("Hayır, 'Gül' listenin içerisinde yok")
```

# Python Tuple- Değiştirme

Bir demet oluşturulduktan sonra değerlerini değiştiremezsiniz. Tuple'lar değiştirilemez veya aynı zamanda denildiği gibi değişmezdir . Ancak bir çözüm var. Demeti bir listeye dönüştürebilir, listeyi değiştirebilir ve listeyi tekrar bir demete dönüştürebilirsiniz.

---

## Tuple bir listeye dönüştürme

```
x = ("elma", "muz", "kiraz")
y = list(x)
y[1] = "kivi"
x = tuple(y)

print(x)
```

# Python Tuple Listeleri

## Tuple ile for kullanımı

```
ulkeler=("Türkiye","Almanya","Kıbrıs","İtalya","ABD")
for x in ulkeler:
    print(x)
```

---

## Tuple Dizin Numaralarıyla Döngü

```
ulkeler=("Türkiye","Almanya","Kıbrıs","İtalya","ABD")
for i in range(len(ulkeler)):
    print(ulkeler[i])
```

## Tuple ile kısaltılmış for kullanımı

```
ulkeler=("Türkiye","Almanya","Kıbrıs","İtalya","ABD")
[print(x) for x in ulkeler]
```

---

## While ile Tuple kullanımı

```
ulkeler=("Türkiye","Almanya","Kıbrıs","İtalya","ABD")
i = 0
while i < len(ulkeler):
    print(ulkeler[i])
    i = i + 1
```

# Python Tuple Birleştir

## Tuple birleştirme Yönetim1

```
tuple1 = ("a", "b" , "c")  
tuple2 = (1, 2, 3)  
tuple3 = tuple1 + tuple2  
print(tuple3)
```

---

## Tuple değerin kaç kez görüldüğü

```
ornek = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)  
  
x = ornek.count(5)  
  
print(x)
```

---

## Tuple çoğaltma

```
meyveler= ("elma", "muz", "kivi")  
yenituple= meyveler* 2  
print(yenituple)
```

## Tuple ilk geçtiği yeri arayın ve konumunu döndürün

```
ornek = (1, 3, 7, 8, 7, 5, 4, 6, 5)  
  
x = ornek.index(8)  
  
print(x)
```

---



# Set / Küme İşlemleri



# Python Set- Tanımlama - Arama

Set , sıralanmamış ve dizine eklenmemiş bir koleksiyondur. Yinelenen üye yok. Kümeler, birden çok öğeyi tek bir değışkende saklamak için kullanılır. Yinelenen değerler göz ardı edilir

## Set iki farklı şekilde oluşturulabilir:

```
turler={"roman","oyku","hikaye","oyku" } | turler =set(("roman","oyku","hikaye","oyku"))  
print(turler)          print(type(turler))    print(len(turler))
```

## String, int ve boolean veri türünde veya karma türde listeler olabilir:

```
set1 = {"elma", "armut", "kivi","mango", "çilek"}  
set2 = {1, 5, 7, 9, 3}  
set3 = {True, False, False}  
karma= {"abc", 34, True, 40, "male"}
```

## Set ile for kullanımı

```
turler={"roman","oyku","hikaye","oyku" }  
for x in turler:  
    print(x)
```

## Öğenin Mevcut olup olmadığını kontrolü

```
if "roman" in turler:  
    print("Evet, 'roman' listenin içerisinde")  
else:  
    print("Hayır, 'roman' listenin içerisinde yok")
```

## Öğenin Mevcut olup olmadığını kontrolü

```
print("oyku" in turler)
```

# Python Set- Değiştirme

## Set Öğe Ekle

```
turler={"roman","oyku","hikaye","oyku" }  
turler.add("deneme")  
print(turler)
```

## Set Öğe Ekle 2. Yöntem

```
turler={"roman","oyku","hikaye"}  
turlerek={ "deneme" ,"açıklama"}  
turler.update(turlerek)  
print(turler)
```

## Set Öğeyi kaldırmak

```
turler.remove("oyku")
```

## Set Öğeyi kaldırmak 2. Yöntem

```
turler.discard("hikaye")
```

## Pop Kullanımı

`pop()` Bir öğeyi kaldırmak için yöntemi de kullanabilirsiniz , ancak bu yöntem *son* öğeyi kaldıracaktır . Setlerin sırasız olduğunu unutmayın, böylece hangi öğenin kaldırılacağını bilemezsiniz.

`pop()` Yöntemin dönüş değeri , kaldırılan öğedir.

## Setin içeriğini silme

```
turler.clear()  
print(turler)
```

## Set Tamamen silme

```
del turler  
print(turler)
```

# Python Döngü Setleri

## Set ile for kullanımı

```
turler={"roman","oyku","hikaye","oyku" }  
for x in turler:  
    print(x)
```

## iki veya daha fazla seti birleştirme

```
set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
set3 = set1.union(set2)  
print(set3)
```

## iki veya daha fazla seti birleştirme

```
set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
set1.update(set2)  
print(set1)
```

## Yinelenenleri Gösterme

```
x = {"telefon", "bilgisayar", "tablet"}  
y = {"tablet", "mobil", "pc"}  
x.intersection_update(y)  
print(x)
```

## Yinelenenlerle yeni bir set

```
x = {"telefon", "bilgisayar", "tablet"}  
y = {"tablet", "mobil", "pc"}  
z = x.intersection(y)  
print(z)
```

## Yinelenenleri Çıkar

```
x = {"telefon", "bilgisayar", "tablet"}  
y = {"tablet", "mobil", "pc"}  
x.symmetric_difference_update(y)  
print(x)
```

## Yinelenenleri Çıkarıp yeni bir set oluştur

```
x = {"telefon", "bilgisayar", "tablet"}  
y = {"tablet", "mobil", "pc"}  
z = x.symmetric_difference(y)  
print(z)
```





# Sözlük / dict İşlemleri



# Python dict- Tanımlama - Arama

Sözlük , sıralanmamış ve değiştirilebilir bir koleksiyondur. Yinelenen üye yok.

## Dict oluşturma:

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz", "mavi"]  
}  
print(arac)  
print(len(arac))  
print(type(arac))
```

## Dict Elemanlarının tümünü yazdırma

```
print(arac)
```

## Dict Elemanlarının Uzunluğu

```
print(len(arac))
```

## Dict Türünü öğrenme

```
print(type(arac))
```

# Python - Sözlük Öğelerine Erişim

## Dict Öğelere Erişim 1. Yöntem

```
x = arac["Model"]  
print(x)
```

## Dict Öğelere Erişim 2. Yöntem

```
x = arac.get("Model")
```

---

### Dict Anahtarları alma

```
x = arac.keys()
```

### Dict Değerleri alma

```
x = arac.values()
```

### Dict Anahtar ve Değeri alma

```
x = arac.items()
```

---

## Dict Sözlüğün içerisinde var mı?

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz", "mavi"]  
}
```

```
if "Model" in arac:  
    print("Sözlüğün içerisinde var")
```

# Python - Sözlük Güncelleme

## Dict Değerlerini Güncelleme

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz",  
    "mavi"]  
}  
arac["Yıl"] = 2021  
print(arac)
```

## Dict Yeni Değer Ekleme

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz", "mavi"]  
}  
arac.update({"Yıl": 2021})  
print(arac)
```

# Python - Sözlük Öğe Ekleme

## Dict Yeni Değer Ekleme 1. Yöntem

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz", "mavi"]  
}  
arac["Vites"] = "Otomatik"  
print(arac)
```

## Dict Yeni Değer Ekleme 2. Yöntem

```
arac = {  
    "Marka": "Ford",  
    "Model": "Mustang",  
    "electric": False,  
    "Yıl": 2020,  
    "renk": ["kırmızı", "beyaz", "mavi"]  
}  
arac.update({"Vites": "Otomatik"})  
print(arac)
```

# Python - Sözlük Öğelerini Kaldır

## Belirtilen anahtar adı ile Silme Yöntem 1

```
arac.pop("Model")  
print(arac)
```

## Belirtilen anahtar adı ile Silme Yöntem 2

```
del arac["Marka"]  
print(arac)
```

## Sözlüğün içini Tamamen Silme

```
arac.clear()  
print(arac)
```

## Rastgele değeri silme

```
arac.popitem()  
print(arac)
```

## Sözlüğü Tamamen Silme Yöntem 1

```
del arac  
print(arac)
```

# Python - Sözlük İçinde Döngü

## Sözlük Anahtar adı döndürme

```
for x in arac:  
    print(x)
```

## Sözlük Anahtar adı döndürme Yöntem 2

```
for x in arac.keys():  
    print(x)
```

## Sözlüğün hem anahtar hem değerini döndürme

```
for x, y in arac.items():  
    print(x, y)
```

## Sözlük Değer döndürme

```
for x in arac:  
    print(arac[x])
```

## Sözlük Değer döndürme Yöntem 1

```
for x in arac.values():  
    print(x)
```

# Python - Sözlüğü Kopyalama

## Sözlüğün kopyalama

```
yenisozluk = arac.copy()
print(yenisozluk)
```

## Üç sözlük içeren bir sözlük oluştur

```
sinif = {
    "ogrenci1" : {
        "adı" : "Ali",
        "yıl" : 2004
    },
    "ogrenci2" : {
        "adı" : "Ceyda",
        "yıl" : 2007
    },
    "ogrenci3" : {
        "adı" : "Numan",
        "yıl" : 2011
    }
}
print(sinif)
```

## Sözlüğün kopyalama Yöntem 2

```
yenisozluk = dict(arac)
print(yenisozluk)
```

## Üç sözlük oluşturun, ardından diğer üç sözlüğü içerecek bir sözlük oluşturma

```
ogrenci1 = {
    "adı" : "Ali",
    "yıl" : 2004 }
ogrenci2 = {
    "adı" : "Ceyda",
    "yıl" : 2007 }
ogrenci3 = {
    "adı" : "Numan",
    "yıl" : 2011 }
sinif = {
    "ogrenci1" : ogrenci1,
    "ogrenci2" : ogrenci2,
    "ogrenci3" : ogrenci3
}
print(sinif)
```





# Fonksiyon



# Python - Fonksiyon İşlemleri

Fonksiyonlar tekrar eden ve sürekli kullanılan kod grubunu bir kere tanımlayarak tekrar tekrar kullanmamızı sağlarlar.

## Basit Fonksiyon oluşturma

```
def basitfonksiyon():  
    print("Merhaba, ilk fonksiyonunum")
```

## Fonksiyonu çağırma

```
basitfonksiyon()
```

## Değer alan fonksiyon

```
def degeralan(kullaniciadi, sifre):  
    print(kullaniciadi + " " + sifre)  
  
degeralan("Ekifkarakas", "12345678")
```

## Değer alan fonksiyonda kaç farklı değer girileceği bilinmiyorsa

```
def degeralan(*sifre):  
    print("Şifre" + " " + sifre[1])  
degeralan("naber", "123456")
```

## İşlevinize kaç anahtar kelime bağımsız değişkeni geçirileceğini bilmiyorsanız

```
def ogrenciototmasyon(**bilgi):  
    print("Öğrenci adı: " + bilgi["adi"])  
    print("Örenci Sınıfı: " + bilgi["sinifi"])  
ogrenciototmasyon(adi = "Ezgi", sinifi = "10")
```

## Varsayılan değer belirleme

```
def ulkeler(ulke = "Türkiye"):  
    print("Merhaba, benim ülkem " + ulke)  
ulkeler()  
ulkeler("Hindistan")  
ulkeler("Arjantin")  
ulkeler("Brazilya")
```

# Python - Fonksiyon İşlemleri

## Değer döndüren Fonksiyon

```
def degerdondurenfonksiyon(sayi):  
    return sayi*10  
soru=int(input("girdiğin sayının onkatını bulacağım, sayı gir "))  
print(degerdondurenfonksiyon(soru))
```

## True / Flase Döndüren Fonksiyon

```
def negatif(sayi):  
    if sayi<0:  
        return True  
    else:  
        return False  
soru=int(input("Sayı gir, negatif mi bulalım:"))  
print(negatif(soru))
```

# Python - Fonksiyon İşlemleri

Faktoriyel nasıl bulunur

```
#f(n)= n * n-1 * n-2 .....1
```

```
#f(4)= 4* 3* 2* 1
```

```
#f(4)= 4* f(3)
```

```
#f(3)= 3* f(2)
```

```
#f(2)= 3* f(1)
```

```
#f(1)=1
```

**Faktoriyel Hesaplam fonksyonu**

```
def faktoriyel(n):
```

```
    if n==1:
```

```
        return 1
```

```
    else:
```

```
        return n* faktoriyel(n-1)
```

```
print(faktoriyel(5))
```

---

**Pozitif sayılar dışında fonksiyon hesabı yapmayan kodu yazalım**

```
def bul(n):
```

```
    if n==1:
```

```
        return 1
```

```
    if n>0:
```

```
        return n* bul(n-1)
```

```
    else:
```

```
        return False
```

```
soru=int(input("Faktoriyelini bulma istediğin sayıyı gir: "))
```

```
if bul(soru)==False:
```

```
    print("Bu sayının faktoriyeli bulunamaz")
```

```
else:
```

```
    print(bul(soru))
```

# Python - Fonksiyon Soru

Fonksiyon kullanarak otomatik liste oluşturun.

Bu liste için önce kullanıcıya kaç elemanlı olacağını sorun.

Daha sonra klavyeden bu sayıları isteyerek listeye kaydedin.

Ekrana yazdırın

---

## Fonksiyon tanımı

```
def sayilarilisteyeekleyenfonksiyon(adet):  
    sayilar=[]  
    for x in range(adet):  
        print(x+1, ".Sayıyı giriniz")  
        sayi=int(input())  
        sayilar.append(sayi)  
    return sayilar
```

## Fonksiyon için değer isteme fonksiyonu çağırma

```
kacadet=int(input("kaç sayı girecekesin? "))  
print( sayilarilisteyeekleyenfonksiyon(kacadet) )
```

# Modül ve Paketler

# Python - Modul Nedir?

Tanımladığımız bir fonksiyon sadece o an içinde bulunduğumuz programda çalışır. Fonksiyonları başka bir program içinde çalıştırmak istersek kopyala yapıştır yapmayacağız. Programı içe aktarmamız (import) fonksiyonu kullanmak için yeterli olacaktır.

Python'da yazılmış her program aynı zamanda birer modüldür. Bu özellik sayesinde Python programlarında bulunan fonksiyon ve özellikler başka Python programlarında da rahatça kullanılabilirler.

---

**modul.py** adında dosyaya ekleyelim

```
def ilkfonksiyon(ad=""):  
    print("Merhaba, ", ad)
```

**koducagir.py**

```
import modul  
modul.ilkfonksiyon("Elif")
```

---

**modul.py** adında dosyaya ekleyelim

```
kisi1 = {  
    "adı": "Erdem",  
    "yaşı": 54,  
    "ülke": "Türkiye" }
```

**koducagir.py**

```
import modul  
a = modul.kisi1["yaşı"]  
print(a)
```

# Python - Modul Nedir?

**Bir Modülü Yeniden Adlandırma (modül ismi kısaltılabilir)**

```
import modul as ml  
a = ml.kisi1["adı"]  
print(a)
```

**Python'nın hazır bazı modüllerini içeri alıp kullanabiliriz**

```
import platform  
bilgi = platform.system()  
print(bilgi)
```

---

**Modülün tüm işlevlerini görmek için dir()**

```
import platform  
bilgi = platform.system()  
print(bilgi)  
print(dir(platform))
```

---

**Modulden sadece tek bir öğeyi almak istersek**

```
from modul import kisi1  
print (kisi1["ulke"])
```



# Python - Modul Paket - Soru

Datetime modülünü kullanarak haftanın çalışma gününde mi, tatil gününde mi olduğu programı yazınız.

---

## Modul çağırma ve yeni fonksiyon oluşturma

```
from datetime import datetime
def gunkontrolu(girilentarikh):
    day=girilentarikh.weekday()
    if day<5:
        return True
    else:
        return False
```

## Fonksiyonu çağırma

```
if gunkontrolu( datetime.now() ):
    print("Hafta içi çalışma günü")
else:
    print("Hafta sonu tatil günü")
```

Datetime modülü için daha fazla örnek için: <https://www.programiz.com/python-programming/datetime>



# Sınıflar ve Nesneler



# Python Sınıfları ve Nesneleri

Python, nesne yönelimli bir programlama dilidir.

Python'daki hemen hemen her şey, özellikleri ve yöntemleri ile bir nesnedir.

Sınıf, bir nesne oluşturucu veya nesneler oluşturmak için bir "taslak" gibidir.

---

**Bir sınıf oluşturmak için anahtar kelime class:**

```
class ilksinif:  
    x = 5  
print(ilksinif)
```

**Artık nesneler oluşturmak için "ilksinif" adlı sınıfı kullanabiliriz:**

```
nesne = ilksinif()  
print(nesne.x)
```

---

## **`__init__` () İşlevi**

Sınıfların anlamak için `__init__` () işlevini anlamamız gerekir.

Tüm sınıfların `__init__` () adında bir işlevi vardır ve bu, sınıf başlatılırken her zaman çalıştırılır.

Nesne özelliklerine değer atamak için `__init__` () kullanın

# Python Sınıfları ve Nesneleri - init()

Kişi adında bir sınıf oluşturun, ad ve yaş için değerler atamak için `__init__()` işlevini kullanın:

```
class kisisinifi:
    def __init__(self, ad, yas):
        self.ad = ad
        self.yas = yas
nesne = kisisinifi("Emre", 14)
print(nesne.ad)
print(nesne.yas)
```

Not: `__init__()` Fonksiyonu sınıf için otomatik olarak her zaman yeni bir nesne oluşturmakta

# Python Sınıfları ve Nesneleri- Nesne Yöntemleri

Nesneler ayrıca yöntemler içerebilir. Nesnelerdeki yöntemler, nesneye ait işlevlerdir. Kişi sınıfında bir yöntem oluşturalım:

```
class kisisinifi:
    def __init__(self, ad, yas):
        self.ad = ad
        self.yas = yas
    def merhaba(self):
        print("Merhaba , benim adım " + self.ad)
nesne = kisisinifi("Emre", 14)
nesne.merhaba()
```

# Python Sınıfları ve Nesneleri - Self

Not: self Parametre sınıfı mevcut örneği için bir referanstır ve sınıfa ait olan erişim değişkenleri için kullanılır. Adlandırılması gerekmez, istediğiniz gibi adlandırabilirsiniz self, ancak sınıftaki herhangi bir işlevin ilk parametresi olmalıdır:

---

## Self örneği

```
class kisisinifi:
    def __init__(self, ad, yas):
        self.ad = ad
        self.yas = yas
    def merhaba(self):
        print("Merhaba , benim adım " +
self.ad)
nesne = kisisinifi("Emre", 14)
nesne.merhaba()
```

## Adlandırılmış hali

```
nclass kisisinifi:
    def __init__(deneme1, ad, yas):
        deneme1.ad = ad
        deneme1.yas = yas
    def merhaba(deneme2):
        print("Merhaba , benim adım " +
deneme2.ad)
nesne = kisisinifi("Emre", 14)
nesne.merhaba()
```

# Python Sınıfları ve Nesne Özellik Değişimi

NEsenenin yaşını güncелейelim

Bir sınıf oluşturmak için anahtar kelime **class**:

```
class kisisinifi:
    def __init__(self, ad, yas):
        self.ad = ad
        self.yas = yas
    def merhaba(self):
        print("Merhaba , benim adım " + self.ad)
nesne = kisisinifi("Emre", 14)
nesne.yas=24
print(nesne.yas)
```

Nesne Özelliklerini Silin

```
del nesne.yas

print(nesne.yas)
```

Nesneleri Silin

```
del nesne

print(nesne)
```

## Pass

Classtanımlar boş olamaz, ancak herhangi bir nedenle class içeriği olmayan bir tanımınız varsa , pass hata almamak için ifadeyi ekleyin.

```
class kisi:
    pass
```



Artık Algoritma yapısı ve  
Python dilini biliyorsunuz =)







YouTube™

elif hoca robotik



Ana Sayfa



Keşfet



Abonelikler



Kitaplık



Geçmiş



Videolarınız



Daha sonra izle

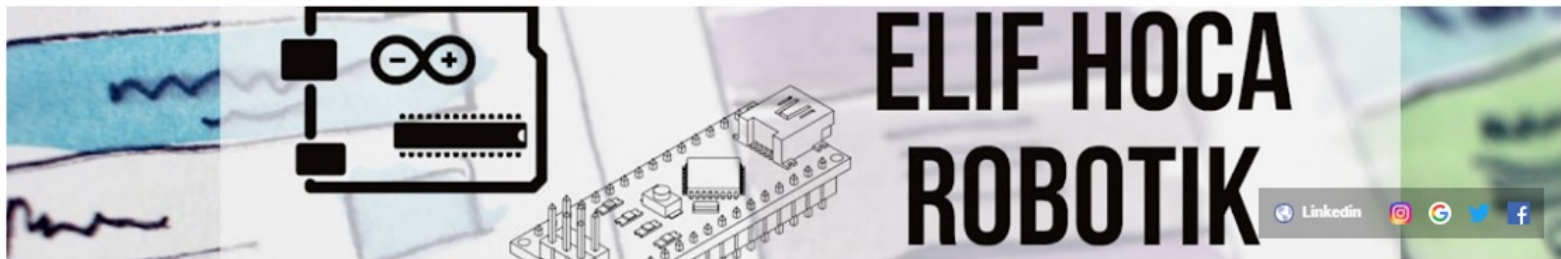


2021 Roket Takımı izl...



Daha fazla göster

ABONELİKLER



ELİF HOCA ROBOTİK

KATIL

ABONE OLUNDU



ANA SAYFA

VİDEOLAR

OYNATMA LİSTELERİ

TOPLULUK

KANALLAR

HAKKINDA



Teknofest Takımları ▶ TÜMÜNÜ OYNAT

Teknofest'te derece almış takımların tecrübelerinden yararlanıyoruz. Teknofest sürecini yaşamış ve derece almış takımların tavsiyelerini bu kanalda bulabilirsin. #teknofesttakımları



# Kaynakça

1. <https://endustri.eskisehir.edu.tr/gurkan.o/ENM104/icerik/w03%20enm%20104%20Flowchart%20and%20Examples.pdf>
2. <https://bmci.edu.pk/wp-content/uploads/2020/05/Algorithm-and-Flow-Chart-notes-for-10th-class.pdf>
3. <https://press.rebus.community/programmingfundamentals/chapter/hello-world/>
4. <https://programming-steps.blogspot.com/2013/06/hello-world-program-flow-chart.html>
5. [https://www.rff.com/flowchart\\_shapes.php](https://www.rff.com/flowchart_shapes.php)
6. <https://beginnersbook.com/2019/03/python-variables/>