Değişkenler

```
a=5
```

```
b=4.5
```

```
c="deneme"
```

Degisken İsimlendirme

- Değişken adı harf, sayı veya alt çizgi karakteri içerebilir.
- Değişken adları büyük küçük harf duyarlıdır.

Kabul Edilmeyen Değişken İsim Atama Senaryoları

- Sayi ile baslamak
- Bosluk icermek
- "*(/ gibi semboller içermek
- Rezerve edilmis isimleri kullanmak

```
d=True  #case sensitive
```

```
#D
```

```
#[a zA Z0 9_] 1abc
```

```
_abc123=45
```

Veri Tipleri

- Numerik (sayısal) veri tipleri → int, float, complex
- String (metin) veri tipleri → str
- Boolean veri tipleri → bool
- Sequence (sıralama) veri tipleri → list, tuple
- Set veri tipleri → set

```
"""
int float bool str complex
```

```
3 list set dict tuple
4 """
```

```
' \nint float bool str complex \nlist set dict tuple \n'
```

Aritmetik Operatörler

- Toplama için +
- Çıkarma için -
- Çarpma için *
- Bölme için /
- Mod almak %
- Floor division operatörü //
- Üs almak **

```
1 45+20
```

```
65
```

```
1 2**3**2
```

```
512
```

```
1 2**4
```

```
16
```

```
1 type(None)
```

```
NoneType
```

```
1 type(10.5)
```

```
float
```

```
1 2**(1/2)
```

```
1.4142135623730951
```

```
1 print(0.1 + 0.2)
```

```
0.30000000000000004
```

```
1 2/7
```

```
0.2857142857142857
```

```
2//7
```

```
0
```

```
7//3
```

```
2
```

```
-13/4
```

```
-3.25
```

```
-13//4
```

```
-4
```

Bazı Hata Çeşitleri

- SyntaxError:
  - Syntax kurallarına uymadan yazdığımız durumlarda bu hata ile karşılaşırız
  - Yanlış veya eksik noktalama işaretleri ya da parantez kullanımı
  - Gçersiz değişken veya fonksiyon isimleri
- TypeError:
  - Birbirinden farklı veri tipleri ile aritmetik işlemler yapmak istediğimiz durumlarda karşılaşırız
  - Mesela integer bir değerle string bir değere toplama işlemi yaptırmak
- IndentationError:
  - Boşluk hatası olarak adlandırılabilir
  - if else veya for döngülerinde yapılacak işlemler bölümünde yeterli boşluk bırakılmaması durumlarında ortaya çıkar.
- NameError:
  - Python'da daha önce tanımlamadığımız değişkenleri kullanmaya çalışırsak NameError ile karşılaşırız
- ZeroDivisionError:
  - Sayının sıfıra bölünmesi durumunda ortaya çıkar

```
2/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
```

```
1 ff
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16316\638493014.py in <module>
----> 1 ff

NameError: name 'ff' is not defined
```

SEARCH STACK OVERFLOW

```
1 if a<5:
2 print()
```

```
  File "C:\Users\SAMİ CİHAN\AppData\Local\Temp\ipykernel_16316\2368111651.py",
line 2
    print()
    ^
IndentationError: expected an indented block
```

SEARCH STACK OVERFLOW

```
1 17%5
```

```
2
```

```
1 10*(2+5)
```

```
70
```

```
1 (2**3)**5
```

```
32768
```

```
1 2**3**5
```

```
141347765182270746366663800059433481266198711750049516649728496103409582 08
```

Print Fonksiyonu

- print() fonksiyonunun görevi ekrana çıktı vermemizi sağlamaktır.

Escape Karakterler

- \t Karakteri → Kelimeler arası 1 tab boşluk bırakmak için kullanılır.
- \n Karakteri → Bir satır aşağıya inmek için kullanılır
- \" karakteri → çift tırnak yazdırmak için kullanılır

```
1 print("merhaba dünya!")
```

```
merhaba dünya!
```

```
1 print('ben', 'sen', 'biz')
```

```
ben sen biz
```

```
1 print('Merhaba ben "ahmet" ')
```

```
Merhaba ben "ahmet"
```

```
1 print("Merhaba ben \"ahmet\" ")
```

```
Merhaba ben "ahmet"
```

```
1 print('Merhaba \nben \t\t"ahmet" ')
```

```
Merhaba
ben             "ahmet"
```

```
1 print("edejk elkjdlkej deknmdjklenld ejdlke \
2 dsfklsdşfşs sdfk şsdf s")
```

```
edejk elkjdlkej deknmdjklenld ejdlke dsfklsdşfşs sdfk şsdf s
```

```
1 print("sonuc: " + str(a))
```

```
sonuc: 5
```

Döküman dizeleri (docstring)

- Pythonda yazdığımız fonksiyonları, sınıfları, modülleri ve metotları dökümante ederken belirli kurallara göre yazdığımız yorum satırlarıdır
- Python tarafından farklı karşılanır ve yazılan kod okunurken bir çeşit döküman oluşturur
- Bir fonksiyon, method veya sınıf tanımlandıktan hemen altındaki satıra üçlü tırnak (" " ") ile bir yorum satırı açılır ve docstring yazılır

""" docstring """ class """açıklama""" def """açıklama"""

```
1 print("""jkhjde  " fgdfg"  'khkhkj' """)
```

```
jkhjde  " fgdfg"  'khkhkj'
```

```
1 abc= """ klsdfşk şkşfs ş şkş dfsşkş sdf
2 fklgşsdflgşlsdf  gsdf gsdf """
```

```
1 abc
```

input() Fonksiyonu

- Kullanıcıdan veri alma fonksiyonudur

type() Fonksiyonu

- Verinin tipini veren fonksiyondur

Type Casting (tür dönüşümü)

bu fonksiyolar ile yapılır;

- int() → veriyi inteteger bir değere dönüştürür
- float() → veriyi ondalıklı bir değere dönüştürür
- str() → veriyi string bir değere dönüştürür

```
1 sayi1 = input("bir deger giriniz: ")
2 sayi2 = input("bir deger giriniz: ")
```

```
bir deger giriniz: 5
bir deger giriniz: 6
```

Stringler ile matematiksel işlemler

- Python'da stringler bir karakter dizisi olduğu ve liste formatına benzediği için bazı matematiksel işlemleri yapmak mümkündür
- İki dize birbiri ile toplama (+) operatörü ile birleştirilebilir
- Python'da karakter dizeleri bir sayı gibi başka bir sayı ile çarpılabilir

```
1 print(sayi1+sayi2)
```

```
56
```

```
1 "ne oldu" *int(sayi2)
```

```
'ne oldune oldune oldune oldune oldune oldu'
```

```
1 print("ne oldu"*sayi2)
```

```
ne oldune oldune oldune oldune oldune oldu
```

```
1 type(sayi1)
```

```
str
```

Taype casting ve "input()" fonksiyonu kullanım örneği

```
sayi1 = int(input("bir deger giriniz: "))
sayi2 = int(input("bir deger giriniz: "))
sayi1+sayi2
```

```
bir deger giriniz: 4
bir deger giriniz: 6
10
```

```
int('3f', 16)
```

```
63
```

```
int("5")
```

```
5
```

```
a=125
```

```
id(a)
```

```
2263668775152
```

"type()" fonksiyonu kullanım örnekleri:

```
type(a)
```

```
int
```

```
a="deneme"
```

```
id(a)
```

```
2263751004976
```

```
type(a)
```

```
str
```

```
a=15.4
```

```
type(a)
```

```
float
```

```
1 a=True
```

```
1 type(a)
```

```
bool
```

```
1 l="",""
```

```
1 type(l)
```

```
tuple
```

Karşılaştırma Operatörleri

- " == " → Eşitse
- " != " → Eşit değilse
- " > " → Büyükse
- " < " → Küçükse
- " >= " → Büyük veya eşitse
- " <= " → Küçük veya eşitse

Mantıksal Operatörler

- and → İşlemlerin hepsinin sonucu True ise sonuç da True olur
- or → İşlemlerden en az bir tanesinin sonucu True olursa sonuç da True olur
- not → Mantıksal işlemin sonucunu ters çevirir

```
1 120 < 50
```

```
False
```

```
1 4 >= 5
```

```
False
```

if Sorguları

kullanım senaryosu-1:

if <koşul sağlanıyorsa>:

```
   <burda yazılanı yap>
```

kullanım senaryosu-2:

if <koşul sağlanıyorsa>:

```
   <burda yazılanı yap>
```

else:

```
   <burda yazılanı yap>
```

kullanım senaryosu-3:

if <koşul sağlanıyorsa>:

```
<burda yazılanı yap>
```

elif <alternatif koşul sağlanıyorsa>:

```
<burda yazılanı yap>
```

else:

```
<burda yazılanı yap>
```

```
1 x=5
2 if x<10:
3     print("dede")
4     if x==0:
5             print("ddd")
6             print("ccc")
7     print("eee")
8 print("xxx")
```

```
dede
eee
xxx
```

```
1 1 <= x <= 10
```

```
True
```

```
""" en kucuk sayi """

sayi1 = int(input("bir deger giriniz: "))
sayi2 = int(input("bir deger giriniz: "))
sayi3 = int(input("bir deger giriniz: "))

minSayi=sayi1
if minSayi>sayi2:
    minSayi=sayi2

if minSayi>sayi3:
    minSayi=sayi3

print(f'En küçük sayı: {minSayi}')
```

```
bir deger giriniz: 5
bir deger giriniz: 6
bir deger giriniz: 4
En küçük sayı: 4
```

math Kütüphanesi

- Math kütüphanesi Python'un standart kütüphanelerinden biridir
- Matematiksel işlemleri yapmanıza kolaylık sağlamaktadır

```
from math import *
```

```
e=23
```

```
e
```

```
23
```

min(), max() ve sum() Fonksiyonları

- min() fonksiyonu girilen değerlerin en küçüğünü döndürür
- max() fonksiyonu girilen değerlerin en büyüğünü döndürür
- sum() fonksiyonu girilen değerlerin toplamını döndürür

```
min(10,50,20)
```

```
10
```

```
max(10,50,20,50,40)
```

```
50
```

```
1 sum([10,20,30])
```

```
60
```

```
1 sum??
```

TypeError hatası örneği:

```
1 sum(10,20,30)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16316\3240342679.py in <module>
----> 1 sum(10,20,30)

TypeError: sum() takes at most 2 arguments (3 given)
```

SEARCH STACK OVERFLOW

```
1 x=100
2 if x>=500:
3     sonuc="deneme"
4 else:
5     sonuc="cc"
```

```
1 sonuc
```

```
'cc'
```

```
1 f='dd' if x>= 100 else "ff"
```

```
1 if x>=100:
2     print("A")
3 elif x>=90:
4     print("B")
5 elif x>=80:
6     print("B")
7 else:
8     print("gg")
```

```
A
```

if Kullanım Örneği

- Girilen sayıları büyükten küçüğe sıralıyor

```

```

```
num1 = int(input("Sayı 1 :"))
num2 = int(input("Sayı 2 :"))
num3 = int(input("Sayı 3 :"))

if num1 >= num2 >= num3:
    print(num1,num2,num3)

if num1 >= num3 >= num2:
    print(num1,num3,num2)

if num2 >= num1 >= num3:
    print(num2,num1,num3)

if num2 >= num3 >= num1:
    print(num2,num3,num1)

if num3 >= num1 >= num2:
    print(num3,num1,num2)

if num3 >= num2 >= num1:
    print(num3,num2,num1)
```

```
Sayı 1 :3
Sayı 2 :4
Sayı 3 :5
5 4 3
```

While Döngüsü

- While döngülerinde belirttiğimiz bir koşul doğru olduğu sürece while bloğu içerisinde tanımladığımız kod satırları tekrarlar

```
x = 4
while x <= 15:
    print(x)
    x+=1
```

```
4
5
6
7
8
9
10
11
12
13
14
15
```

For Döngüleri

- for döngüleri bir eleman grubundaki (list, tuple, dictionary, set ya da string) her bir elemana ulaşmak için kullanılır
- kullanım yapısı:

for <degisken> in <iterable>:

```
<burda yazılıanı yap>
```

Iterable

- Python'da gezinilebilen(iterable) nesne olarak kullanılabilen bir nesneye iterable obje denir
- Liste(list), sözlük(dict), kümeler(set) ve hatta range gibi Python koleksiyonlarının çoğu iterabledir

range() fonksiyonu

- Range fonksiyonu ile for döngüsünün dönme sayısı kadar tekrarlayan işlemler yapılabilir

```
1 for i in range(5):
2     print(i, end=' ')
```

```
0 1 2 3 4
```

```
1 a=range(5)
```

```
1 c="deneme"
```

```
1 c[5]
```

```
'e'
```

For döngüsü ve range() fonksiyonu beraber kullanım örnekleri:

```
1 for i in "range(5)":
2     print(i, end=' ')
```

```
r a n g e ( 5 )
```

```
1 for i in [10,20,30,40]:
2     print(i, end=' ')
```

```
10 20 30 40
```

```
1 print(10,20,30,40, sep=",")
```

```
10,20,30,40
```

```
toplam=0
for i in [10,20,30,40]:
    toplam+=i
```

```
toplam
```

```
100
```

```
l=[i*10 for i in range(1,11)]
```

```
l
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Kullanıcının girdiği sayıların ortalamalarını hesaplayan döngü örnekleri:

```
sayac=int(input("döngü sayısını giriniz: "))
toplam=0
for i in range(sayac):
    deger=int(input("döngü sayısını giriniz: "))
    toplam+=deger
print(f"Toplam: {toplam} ve ortalama: {(toplam/sayac):.2f}")
```

```
döngü sayısını giriniz: 3
döngü sayısını giriniz: 5
döngü sayısını giriniz: 4
döngü sayısını giriniz: 5
Toplam: 14 ve ortalama: 4.67
```

```
deger = int(input ("veri giriniz: cikmak icin -1 "))
toplam = 0
sayac = 0

while deger != -1:
    toplam+=deger
    sayac+=1
    deger=int(input("veri giriniz: cikmak için -1 "))

if sayac!=0:
    print(f"Toplam: {toplam} ve ortalama: {(toplam/sayac):.2f}")
else:
    print("en az bir sayı giriniz: ")
```

```
veri giriniz: cikmak icin -1 5
veri giriniz: cikmak için -1 -1
Toplam: 5 ve ortalama: 5.00
```

```
1 for i in range(10, 1, -2):
2     print(i)
```

```
10
8
6
4
2
```

```
1 0.1 + 0.2
```

```
0.30000000000000004
```

```
1 x=52.31
```

```
1 x
```

```
52.31
```

```
1 print(f'{x:.20f}')
```

```
52.31000000000000227374
```

Decimal Sınıfı

Decimal sınıfının kullanıldığı durumlar

- Tam ondalık gösterime ihtiyaç duyan finansal uygulamalar yaparken
- Gerekli hassasiyet seviyesini kontrol etmek istediğimizde
- Önemli ondalık basamaklar kavramını uygulamak istediğimizde

```
1 from decimal import Decimal
```

```
1 a=Decimal('1000.00')
```

```
1 a
```

```
Decimal('1000.00')
```

```
1 type(a)
```

```
decimal.Decimal
```

```
1 print(f'{a:.20f}')
```

```
1000.00000000000000000000
```

```
1 b=Decimal('0.05')
```

```
1 a+b
```

```
Decimal('1000.05')
```

```
1 a/b
```

```
Decimal('20000')
```

```
1 a//b
```

```
Decimal('20000')
```

For döngüsü kullanım örnekleri:

```
1 for i in range(1,11):
2     s=a* (1+b)**i
3     print(f"{i:<2}{s:>10.2f}")
```

```
1    1050.00
2    1102.50
3    1157.62
4    1215.51
5    1276.28
6    1340.10
7    1407.10
8    1477.46
9    1551.33
10   1628.89
```

```
1 for x in range(5):
2     for i in range(10):
3         if i == 5 or i==6:
4             continue
5         print(i, end=' ')
6     print()
```

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

```
1 l
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
1 sum(l)/len(l)
```

```
55.0
```

```
1 len(l)
```

```
10
```

Statistics kütüphanesi

- Bu modül, sayısal verilerin matematiksel istatistiklerini hesaplamak için işlevler sağlar

```
1 import statistics
```

```
1 statistics.mean(l)
```

```
55
```

```
1 statistics.median(l)
```

```
55.0
```

```
1 statistics.mode(l)
```

```
10
```

```
1 sorted(l)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
1 l
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
1 import numpy as np
```

```
1 np.percentile(l,25)
```

```
32.5
```

```
1 import math
```