Fonksiyonlar

Fonksiyonların tanımlanması

```
def champion():
    print("Şampiyon")
    print("FB")

type(champion)

champion()
```

```
Şampiyon
FB
```

```
champion("Galatasaray")
```

```
ŞAmpiyon : Fenerbahçe
FB
```

```
def ad(isim):
    print("İsminiz:", isim)

ad("xxxxxxx")
```

```
İsminiz: xxxxxxx
```

```
def toplama(a,b,c):
    print("Toplamları:", a + b + c)

toplama(3,4,5)
```

```
def faktoriyel(sayı):
    faktoriyel = 1
    if (sayı == 0 or sayı == 1):
        print("Faktoriyel:", faktoriyel)
    else:
        while(sayı >= 1):
            faktoriyel *= sayı
            sayı -= 1
        print("Faktoriyel:", faktoriyel)

faktoriyel(0)
```

```
Faktoriyel: 1
```

```
faktoriyel(1)
```

```
Faktoriyel: 1
```

```
faktoriyel(5)
```

```
Faktoriyel: 120
```

```
faktoriyel(20)
```

```
Faktoriyel: 2432902008176640000
```

Fonksiyonlarda Return

```
def toplama(a,b,c):
    print("Toplamları:", a+b+c)

def ikikatı(a):
    print("iki katı", 2 * a)

x = toplama(3,4,5)

ikikatı(x)
```

```
Toplamları: 12
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12464\3905052324.py in <module>
      1 x = toplama(3,4,5)
      2
----> 3 ikikatı(x)

~\AppData\Local\Temp\ipykernel_12464\737395467.py in ikikatı(a)
      1 def ikikatı(a):
----> 2     print("iki katı", 2 * a)

TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'
```

```
type(x)
```

```
NoneType
```

```
def toplama(a,b,c):
    return a+b+c

def ikikatı(a):
    return 2 * a

xx = toplama (3,4,5)

print(ikikatı(xx))
```

```
24
```

```
type(xx)
```

```
int
```

```
def toplama(a,b):
    print("1.fonksiyon çalıştı", a+b)
    return a + b

def ikikatı(a):
    print("2.fonksiyon çalıştı", 2*a)
    return 2 * a

def yarısı(a):
    print("3.fonksiyon çalıştı", a/2)
    return a / 2

print(yarısı(ikikatı(toplama(9,8))))
```

```
1.fonksiyon çalıştı 17
2.fonksiyon çalıştı 34
3.fonksiyon çalıştı 17.0
17.0
```

return' ün bir özelliği, fonksiyonu sonlandırmasıdır. Yani, fonksiyon tanımında return komutundan sonraki komutlar görünmez ve işletilemez

```
def toplama(a,b):
    return a + b
    print("toplama fonskiyonu")

toplama(4,5)
```

```
9
```

```
print(toplama(4,5))

def toplama(a,b):
    print("toplama fonskiyonu")
    return a + b

print(toplama(4,5))
```

```
toplama fonskiyonu
9
```

Dış dünyaya bilgi göndermeyen fonksiyonlara "void" fonksiyonlar denir.

```
toplama()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8832\642249608.py in <module>
----> 1 toplama()

TypeError: toplama() missing 2 required positional arguments: 'a' and 'b'
```

Fonksiyonlarda Parametre Türleri

```
def selamla(isim):
    print("selam", isim)

selamla("Cemre")

selamla()

def selamla(isim = "isimsiz"):
    print("selam", isim)

selamla("Cemre")

selam Cemre

selamla()

selam isimsiz

a, b, c = 24, 48, 63
a

def bilgiler(ad = "Bilgi yok",
             soyad = "Bilgi yok",
             numara = "Bilgi yok"):

    print("Ad: ", ad, "//" ,
          "Soyad: ", soyad, "//",
          "Numara: ", numara)

bilgiler()

Ad:  Bilgi yok // Soyad:  Bilgi yok // Numara:  Bilgi yok

bilgiler("Gökhan", "Türeci", "1234")

Ad:  Gökhan // Soyad:  Türeci // Numara:  1234

bilgiler("Gökhan","Türeci")
```

```
Ad:  Gökhan // Soyad:  Türeci // Numara:  Bilgi yok
```

```
bilgiler(2)
```

```
Ad:  2 // Soyad:  Bilgi yok // Numara:  Bilgi yok
```

```
bilgiler("1234")
```

```
Ad:  1234 // Soyad:  Bilgi yok // Numara:  Bilgi yok
```

```
bilgiler(numara = "1234")
```

```
Ad:  Bilgi yok // Soyad:  Bilgi yok // Numara:  1234
```

Esnek sayıda değerler

```
def toplama(a,b,c):
    print(a + b + c)
```

```
toplama(3,4,5)
```

```
12
```

```
toplama(3,4,5,6)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8832\4005063532.py in <module>
----> 1 toplama(3,4,5,6)

TypeError: toplama() takes 3 positional arguments but 4 were given
```

```
def toplama(*a):
    print(a)
```

```
toplama(1,2,3, 4, 5, 8,6,7,9)
```

```
(1, 2, 3, 4, 5, 8, 6, 7, 9)
```

```
def toplama(*a):
    toplam = 0
    print(a)
    for i in a:
        toplam += i
    print("toplam= ", toplam)
```

```
toplama(1,2,3,4,40,100)
```

```
(1, 2, 3, 4, 40, 100)
toplam=  150

toplama(1, 2, 3, 4, 5)
```

Global ve Yerel Değişkenler

```
def fonksiyon():
    a = 19
    print(a)

fonksiyon()
print(a)

b = 5
def fonksiyon():
    print(b)

fonksiyon()
print(b)

c = 10
def fonksiyon():
    c = 2
    print(c)
fonksiyon()
print(c)

d = 5

def fonksiyon():
    global d
    d = 3
    print(d)

fonksiyon()
print(d)
```

Not: if veya while içinde tanımlanan değişkenler global değişkenlerdir !

Lambda ifadeleri

```
#list comprehension
liste1 = [1,2,3,4,5]

liste2 = [i * 2 for i in liste1]
liste2
```

```
[2, 4, 6, 8, 10]
```

```
def dörtkatı(x):
    return x * 4

print(dörtkatı(5))
```

```
20
```

```
type(dörtkatı)
```

```
function
```

```
dörtX = lambda x : x * 4

print(dörtX(5))
```

```
20
```

```
type(dörtX)
```

```
function
```

```
def toplama(x,y,z):
    return x + y + z

print(toplama(10,20,90))
```

```
120
```

```
toplama2 = lambda x, y, z : x + y + z

print(toplama2(10,20,90))
```

```
120
```

```
def tersçevir(s):
    return s[::-1]

print(tersçevir("Python Programlama"))
```

```
amalmargorP nohtyP
```

```
ters = lambda s : s[::-1]

print(ters("Python"))
```

```
nohtyP
```

```
def tekçift(sayı):
    return sayı % 2 == 0

print(tekçift(15))
```

```
False
```

```
print(tekçift(8))
```

```
True
```

```
cifttek = lambda sayı : sayı % 2 == 0

print(cifttek(27))
```

```
False
```

```
print(cifttek(26))
```

```
True
```

UYGULAMA : Bir sayının asal sayı olup olmadığının kontrolü

```
"""
Asal sayılar sadece kendisine ve 1 sayısına bölünebilirler

"""

def asal_mi(sayı):
    if (sayı == 1):
        return True
    elif (sayı == 2):
        return True

    else:
        for i in range(2, sayı):
            if (sayı % i == 0):
                return False
        return True

""" 2'den sayıya kadar, sayı dahil değil bir liste oluşturduk, eğer bu
liste içindeki
```

```
bir sayı, sayıyı tam bölüyorsa o zaman sayımız asal değildir."""

while True:
    sayı = input("Sayı:")

    if (sayı == "q"):
        print("program sonlanıyor")
        break

    else:
        sayı = int(sayı)

    if (asal_mi(sayı)):
        print(sayı,"asal bir sayıdır")
    else:
        print(sayı, "asal bir sayı değildir.")
```

```
Sayı:1
1 asal bir sayıdır
Sayı:4
4 asal bir sayı değildir.
Sayı:5
5 asal bir sayıdır
Sayı:7
7 asal bir sayıdır
Sayı:98
98 asal bir sayı değildir.
Sayı:q
program sonlanıyor
```

```
"""
Bir sayının tam bölenleri

Sayıya kadar liste oluşturabilir ve bu liste içinde olup
sayıyı tam bölenleri ayırabiliriz.

"""

def tambölenler(sayı):
    tam_bölenler = []

    for i in range(2, sayı+1):

        if (sayı % i == 0):
            tam_bölenler.append(i)
    return tam_bölenler

while True:
    sayı = input("Sayı:")

    if (sayı == "q"):
```

```
        print("Program sonlanıyor")
        break
    else:
        sayı = int(sayı)
        print("Tam Bölenler:",tambölenler(sayı))
```

```
Sayı:97
Tam Bölenler: [97]
Sayı:90
Tam Bölenler: [2, 3, 5, 6, 9, 10, 15, 18, 30, 45, 90]
```