def <fonksiyon_ismi>(<argümanlar>): # snake_case """ Bu kod ne ise yarar # docstring """ ... # return/print

In [124]:
```
def bes_bastir():
    """
    burası aynı # gibi çalışır ve adı docstringtir
    satırlarca
    açıklama
    yapabilirsin
    """
    print(5)
```

In [125]:
```
bes_bastir()
```

```
5
```

### Fonksiyonlarda print ve return farki

In [7]:
```
def bes_dondur():
    return 5 # print gibidir ama print gibi satır satır çağrılmaz sadece ilki çağrılır
    return 6 # mesela burayı basmaz
```

In [9]:
```
bes_dondur()
```

Out [9]: 5

In [13]:
```
a = bes_bastir()
a
```

```
5
```

In [14]:
```
print(a) #print print içine girmedi
```

```
None
```

In [15]:
```
b = bes_dondur()
b
```

Out [15]: 5

In [16]:
```
print(b) #ama return print içine girdi
```

```
5
```

### Fonksiyonlarda Argümanlar

In [20]:
```
def sayi_dondur(sayi): #parantez içi argüman
    return sayi
```

In [22]:
```
sayi_dondur(250)
```

Out [22]: 250

In [26]:
```
def sayi_bastir(sayi=12): #parantez içi default argüman ile return bağlandı
    return sayi
```

In [27]:
```
sayi_bastir() # boş girersen default döner
```

Out [27]: 12

In [30]:
```
sayi_bastir(21) # input verirsen onu kullanır
```

Out [30]: 21

In [35]:
```
def buyuk_sayi_dondur(a,b):
    if a>b:
        return a
    elif b>a:
        return b
```

In [36]:
```
print(buyuk_sayi_dondur(10,5))
print(buyuk_sayi_dondur(8,24))
```

```
10
24
```

### Fonksiyonların Birbiri ile İlişkisi

In [42]:
```
def buyuk_sayi_yazdır(a,b):
    buyuk_sayi=buyuk_sayi_dondur(a,b) #fonksiyonlar birbirleri içinde çağrılabilir
    sablon_metin="{} sayısı daha büyüktür".format(buyuk_sayi)
    return sablon_metin
```

In [45]:
```
buyuk_sayi_yazdır(15,20)
```

Out [45]:
```
'20 sayısı daha büyüktür'
```

### Fonksiyonlar Birden Fazla Sonuc Döndürebilir

In [51]:
```
def ad_soyad_ayirma(adSoyad): #ikinci kelimenin ilk harfini büyük yazmak ex. adSoyad javada yaygın bir isimlendirme yöntemi
    ad=adSoyad.split(" ")[0]
    soyad=adSoyad.split(" ")[1]
    return ad,soyad
```

In [52]:
```
ad_soyad_ayirma("Yavuz Baştemur")
```

Out [52]:
```
('Yavuz', 'Baştemur')
```

In [53]:
```
def ad_soyad_ayirma2(adSoyad):
    ad=adSoyad.split(" ")[0]
    soyad=adSoyad.split(" ")[1]
    print(ad)
    print(soyad)
```

In [54]:
```
ad_soyad_ayirma2("Yavuz Baştemur")
```

```
Yavuz
Baştemur
```

### *args Argümani

In [58]:
```
#join kullanımı
" ".join(["Yavuz","Selim"]) # ilk kısım " " ne ile birleştirileceğini belirtiyor, burada boşlukla birleşmelerini istiyoruz
                            # join(buraya liste şeklinde verildiğinde liste birleştirilir) bu yüzden liste halinde vermek g
```

Out [58]:
```
'Yavuz Selim'
```

In [61]:
```
def adSoyad_birlestir(ad,soyad):
    return " ".join([ad,soyad])
```

In [62]:
```
adSoyad_birlestir("Yavuz", "Baştemur")
```

Out [62]:
```
'Yavuz Baştemur'
```

In [63]:
```
adSoyad_birlestir("Yavuz","Selim","Baştemur") #fonksiyonun 2 argümanı varken 3 girdide bulununca hata verdi
```

```
---------------------------------------------------------------------------TypeError                                 Traceback (most recent call last)Cell In[63], 
----> 1 adSoyad_birlestir("Yavuz","Selim","Baştemur")
TypeError: adSoyad_birlestir() takes 2 positional arguments but 3 were given
```

In [68]:
```
def adSoyad_birlestir(*args): #args girdilerden oluşan bir argümanlar "listesini" belirttiğinden [] gerekmedi * unutulmamal
    return " ".join(args) #burada * yok (bilmiom neden?)
```

In [69]:
```
adSoyad_birlestir("Yavuz","Selim","Baştemur")
```

Out [69]:
```
'Yavuz Selim Baştemur'
```

In [91]:
```
def isim_soyisim_yazdir(**kwargs):
    for kw in kwargs:
        print(kw)
```

In [86]:
```
isim_soyisim_yazdir(isim="Yavuz",ikinci_isim="Selim",soyisim="Baştemur")
```

```
isim
ikinci_isim
soyisim
```

In [88]:
```
def isim_soyisim_yazdir2(**kwargs):
        print(kwargs)
isim_soyisim_yazdir2(isim="Yavuz",ikinci_isim="Selim",soyisim="Baştemur")
```

```
{'isim': 'Yavuz', 'ikinci_isim': 'Selim', 'soyisim': 'Baştemur'}
```

In [101]:
```
def ikinci_isim_yazdir(**kwargs):
    if "ikinci_isim" in kwargs:      #eleman arandığı için tırnak var
        print(kwargs["ikinci_isim"]) #listeceki eleman arandığı için parantez var, eleman olduğu için tırnak var !!
                                     #parantez ve tırnaklar önemli
```

```
    else:
        print("İkinci ismi yok")
```

In [102]:
```
ikinci_isim_yazdir(isim="Yavuz",ikinci_isim="Selim",soyisim="Baştemur")
```

```
Selim
```

In [103]:
```
ikinci_isim_yazdir(isim="Alperen",soyisim="Bahat")
```

```
İkinci ismi yok
```

### map, filter ve lambda Expressions

#### map

In [105]:
```
def kareAl(x):
    return x**2
```

In [106]:
```
kareAl(5)
```

Out [106]: 25

In [110]:
```
sayilar=range(1,6)
map(kareAl, sayilar) #bize oluşturulduğuna dair bir çıktı veriyor ancak okunaklı değil
                     #bu yüzden [* ] broadcast ile çıktıyı bir listeye dönüştürüp okuyabiliriz
```

Out [110]: <map at 0x7fc6f9a38670>

In [111]:
```
sayilar=range(1,6)
[*map(kareAl, sayilar)] #map ile map(neyi, neye) uygulayacağımızı belriliyoruz
                        #sırasıyla verilen fonksiyon verilen listedeki her elemana uygulanır
```

Out [111]: [1, 4, 9, 16, 25]

In [123]:
```
# map olmadan da şöyle yapılabilirdi
sayilar=[*range(1,6)]
print(sayilar)
print([*range(len(sayilar))])
for index in range(len(sayilar)): # böylece 0dan 4e kadar index işaretlenmiş olur çünkü 5 eleman 0-4 eder belirtmek zorunda
    sayilar[index]=kareAl(sayilar[index]) #indexi girip sayıyı sorgulatıp onun karesini aldırıyoruz
print(sayilar)
```

```
[1, 2, 3, 4, 5]
[0, 1, 2, 3, 4]
[1, 4, 9, 16, 25]
```

#### filter

In [129]:
```
def ciftSayi_filtresi(x):
    if x%2==0:
        return x
    else:            # bu kısımdan sonrasına gerek yok ancak yine de yazalım olmasa da çalışır
        return None
```

In [130]:
```
ciftSayi_filtresi(3)
```

In [131]:
```
ciftSayi_filtresi(4)
```

Out [131]: 4

In [132]:
```
def ciftSayi_filtresi2(x):
    return x if x%2==0 else None # daha pythonista bir yazım şekli
```

In [133]:
```
ciftSayi_filtresi2(3)
```

In [134]:
```
ciftSayi_filtresi2(4)
```

Out [134]: 4

In [139]:
```
sayilar=[*range(1,6)]
[*map(ciftSayi_filtresi2, sayilar)] #filter ile yapacağımızı map ile de yaparız ama çıktısı bulunmayan sonuçlar verimizi ki
```

Out [139]: [None, 2, None, 4, None]

In [141]:
```
sayilar=[*range(1,6)]
[*filter(ciftSayi_filtresi2, sayilar)] #filter çıktısı bulunmayan sonuçları eler, aynı map gibi listelemek gerektirir
```

Out [141]: [2, 4]

### lambda

In [142]:
```
sayilar=[*range(1,6)]
[*map(lambda x: x**2, sayilar)] #lambda ile fonksiyon yazmadan sofistike olmayan fonksiyonları map veya filter içinde...
                                #...istenilen listeye uygulayabiliriz
```

Out [142]: [1, 4, 9, 16, 25]

In [144]:
```
sayilar=[*range(1,6)]
[*filter(lambda x: x if x%2==0 else None, sayilar)] #bu da filter ile uygulanması
```

Out [144]: [2, 4]

### Kullanici Girdisi

In [145]:
```
input("Bir sayı girin:")
```

```
Bir sayı girin: 15
```

Out [145]: '15'

In [146]:
```
girdi=input()
type(girdi) #input girdileri hep stringtir eğer mecbursak tipini değiştirebiliriz
```

```
 5
```

Out [146]: str

In [147]:
```
girdi=input()
type(int(girdi))
```

```
 5
```

Out [147]: int

In [148]:
```
girdi=input()
type(int(girdi)) # böylece yazı yazarsak hata verir
```

```
 deneme
```

```
---------------------------------------------------------------------------ValueError                                Traceback (most recent call last)Cell In[148],
      1 girdi=input()
----> 2 type(int(girdi))
ValueError: invalid literal for int() with base 10: 'deneme'
```

In [176]:
```
def tek_miCift_mi():
    sayi=input("Bir sayı giriniz:")
    tekmiCiftmi=input("Girdiğiniz sayı çift ise 0, tek ise 1 yazınız:")
    return "Doğru!" if (int(sayi)%2)==int(tekmiCiftmi) else "Yanlış!"
```

In [177]:
```
tek_miCift_mi()
```

```
Bir sayı giriniz: 23
Girdiğiniz sayı çift ise 0, tek ise 1 yazınız: 0
```

Out [177]: 'Yanlış!'

In [178]:
```
tek_miCift_mi()
```

```
Bir sayı giriniz: 36
Girdiğiniz sayı çift ise 0, tek ise 1 yazınız: 0
```

Out [178]: 'Doğru!'

### Kullanici Girdisini Onaylamak

In [179]:
```
def sayiMi():
    sayi=input("Sayi giriniz:")
    if sayi.isdigit(): # .isdigit() komutu digit ise true değilse false yanıtını vererek mantıksal çıktılara katılırlar
        return "Başarılı."
    else:
        return "Başarısız."
```

In [180]:
```
sayiMi()
```

```
Sayi giriniz: e
```

Out [180]: 'Başarısız.'

In [181]:
```
sayiMi()
```

```
Sayi giriniz: 2
```

Out [181]: 'Başarılı.'

In [192]:
```
def sayiMiLoop():
    sayi=input("Sayı giriniz:")
    while sayi.isdigit():
        return "Başarılı."
```

```
    else:
        print("Başarısız, tekrar deneyniz.")
        sayi=input("Sayı giriniz:")
```

In [186]:
```
sayiMiLoop()
```

```
Sayı giriniz: 2
```

Out [186]: 'Başarılı.'

In [190]:
```
sayiMiLoop() # doğru çalışmaz çünkü while kısmı sonsuza dek dönebilirken else ile döngüden çıkar,
             # tekrar girdiyi while kısmına eklemeliyiz
```

```
Sayı giriniz: e
Başarısız, tekrar deneyniz.
Sayı giriniz: e
```

In [195]:
```
def sayiMiLoop2():
    sayi=input("Sayı giriniz:")
    while not sayi.isdigit(): # not kullanarak mantığımızı tersine çevirip while döngüsüne sokuyoruz
        print("Başarısız, tekrar deneyniz.")
        sayi=input("Sayı giriniz:")
    else:
        return "Başarılı."
```

In [194]:
```
sayiMiLoop2()
```

```
Sayı giriniz: e
Başarısız, tekrar deneyniz.
Sayı giriniz: e
Başarısız, tekrar deneyniz.
Sayı giriniz: e
Başarısız, tekrar deneyniz.
Sayı giriniz: 2
```

Out [194]: 'Başarılı.'

In [196]:
```
def emailKontrol(): #emaillerin ortak yani @ ve . içermeleri
    email=input("E-posta adresinizi giriniz:")
    while not ("@" and ".") in email:
        print("Başarısız, tekrar deneyiniz:")
        email=input("E-posta adresinizi giriniz:")
    else:
        return "E-posta adresiniz başarıyla kaydedildi."
```

In [198]:
```
emailKontrol()
```

```
E-posta adresinizi giriniz: asdasd
Başarısız, tekrar deneyiniz:
E-posta adresinizi giriniz: 3214e2w
Başarısız, tekrar deneyiniz:
E-posta adresinizi giriniz: dasda@asdfasd
Başarısız, tekrar deneyiniz:
E-posta adresinizi giriniz: asdas@asdas.asda.asd
```

Out [198]: 'E-posta adresiniz başarıyla kaydedildi.'

### Try, Except ve Finally Komutlari

In [199]:
```
round(1.6)
```

Out [199]: 2

In [215]:
```
def tamSayi_yuvarla():

    while True: #sonsuz döngü oluşturmak için while true kullanılabilir
        girdi = input("Yuvarlamak istediğiniz sayıyı giriniz:")
        status= " "
        try: # okuma sırasına göre ilk bunu deneyip başarılı sonuç alınırsa çıktı verecek
            girdi=float(girdi)
            print("Yuvarlama işleminin sonucu {}.".format(round(girdi)))
            status="başarılı."
            break #başarılı sonuç ile sonsuz while true döngüsünden çıkılır
        except: # başarısız sonuç alınacak harf girersek hata vermeyip okunaklı bir çıktı almamızı sağladı
            print("Girdiğiniz sayı geçersiz.")
            status="başarısız."
            pass # başarısız sonuç ile döngüye geri girilir
        finally: # try veya except ile seçilecek herhangi bi yola ek olarak her türlü yapılacak komuttur
            print("Çevirim {}".format(status))
```

In [216]:
```
tamSayi_yuvarla()
```

```
Yuvarlamak istediğiniz sayıyı giriniz: a

Girdiğiniz sayı geçersiz.
Çevirim başarısız.

Yuvarlamak istediğiniz sayıyı giriniz: 3.6

Yuvarlama işleminin sonucu 4.
Çevirim başarılı.
```

**Exception Tipleri**

https://docs.python.org/3/tutorial/errors.html

In [217]:
```
5 + 'a'
```

```
---------------------------------------------------------------------------TypeError                                 Traceback (most recent call last)Cell In[217],
----> 1 5 + 'a'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [221]:
```
try:
    5 + "a"
except IndexError:
    print("Seçtiğiniz eleman listede ulaşılamıyor.")
except TypeError:
    print("Yapılan işlem için geçersiz bir tipte girdide bulundunuz.")
except:
    print("Kod düzgün çalışmıyor.")
```

```
Yapılan işlem için geçersiz bir tipte girdide bulundunuz.
```

In [223]:
```
liste = []
liste[4]
```

```
---------------------------------------------------------------------------IndexError                                Traceback (most recent call last)Cell In[223],
      1 liste = []
----> 2 liste[4]
IndexError: list index out of range
```

In [224]:
```
liste = []
try:
    liste[4]
except IndexError:
    print("Seçtiğiniz eleman listede ulaşılamıyor.")
except TypeError:
    print("Yapılan işlem için geçersiz bir tipte girdide bulundunuz.")
except:
    print("Kod düzgün çalışmıyor.")
```

```
Seçtiğiniz eleman listede ulaşılamıyor.
```

In [219]:
```
vatandas = {
    'AD': 'Oguz',
    'TC_NO': 123123
}

vatandas['PASS_NO']
```

```
---------------------------------------------------------------------------KeyError                                  Traceback (most recent call last)Cell In[219],
      1 vatandas = {
      2     'AD': 'Oguz',
      3     'TC_NO': 123123
      4 }
----> 6 vatandas['PASS_NO']
KeyError: 'PASS_NO'
```

In [225]:
```
vatandas = {
    'AD': 'Oguz',
    'TC_NO': 123123
}
try:
    vatandas['PASS_NO']
except IndexError:
    print("Seçtiğiniz eleman listede ulaşılamıyor.")
except TypeError:
    print("Yapılan işlem için geçersiz bir tipte girdide bulundunuz.")
except:
    print("Kod düzgün çalışmıyor.")
```

```
Kod düzgün çalışmıyor.
```

In [226]:
```
vatandas = {
    'AD': 'Oguz',
    'TC_NO': 123123
}
try:
    vatandas['PASS_NO']
except IndexError:
```

```
    print("Seçtiğiniz eleman listede ulaşılamıyor.")
except TypeError:
    print("Yapılan işlem için geçersiz bir tipte girdide bulundunuz.")
except KeyError:
    print("Aramaya çalıştığınız anahtar kayıtlarda bulunmuyor.")
except:
    print("Kod düzgün çalışmıyor.")
```

```
Aramaya çalıştığınız anahtar kayıtlarda bulunmuyor.
```

In [ ]: