In [13]:
```
print("Hello world")
```

```
Hello world
```

In [1]:
```
print("Benim adim \nYavuz")
```

```
Benim adim
Yavuz
```

In [5]:
```
print("Benim adim \t Yavuz")
```

```
Benim adim	 Yavuz
```

.format() kullanımı

In [8]:
```
print("Benim adim {}".format("Yavuz"))
```

```
Benim adim Yavuz
```

In [9]:
```
print("Benim adim {}, yasim {}".format("Yavuz","24"))
```

```
Benim adim Yavuz, yasim 24
```

In [10]:
```
print("Benim adim {1}, yasim {0}".format("24","Yavuz"))
```

```
Benim adim Yavuz, yasim 24
```

In [14]:
```
print("Benim adim {ad}, yasim {yas}".format(yas=24,ad="Yavuz")) #text yazdırmak için "Yavuz" tırnak içinde yazılmalı
```

```
Benim adim Yavuz, yasim 24
```

Değişkenler

Kabul edilmeyen degisken isim atama senaryolari: • sayi ile baslamak • bosluk icermek • "*(/ gibi semboller icermek • rezerve edilmis isimleri kullanmak

In [15]:
```
Yavuz Selim BAŞTEMURsayi = 10
print(sayi)
```

```
10
```

In [16]:
```
sayi = 11
print(sayi)
```

```
11
```

In [17]:
```
sayi = sayi + 1
print(sayi)
```

```
12
```

In [19]:
```
sayi_ilk = 10
print(sayi_ilk)
```

```
10
```

In [33]:
```
#1say = 10
#print(1say) hata verir
```

In [14]:
```
sayi1 = 10
print(sayi1)
```

```
10
```

In [34]:
```
#sayi ilk = 10
#print(sayi ilk) hata verir
```

In [15]:
```
_=11
print(_)
```

```
11
```

In [40]:
```
#print="abc"
#print çalışır ama sonra sıkıntı verebilir
```

In [41]:
```
#print="abc"
#print(print) Hata verir
```

Veri Tipleri

Integer int 3, 10 Float float 1.5, 7.0 String str "Slm", "2" Boolean bool True, False List list [1,2,True,"a",1] Set set {1,2,True} Dictionary dict {"isim":"Yavuz", "yas":32} Tuple tup (1,2,True)

In [17]:
```
print(5)
3
```

```
5
```

Out [17]: 3

In [18]:
```
4+7
```

Out [18]: 11

In [25]:
```
type(6)
```

Out [25]: int

In [26]:
```
type(1.5)
```

Out [26]: float

In [27]:
```
5%3 #modüler
```

Out [27]: 2

In [28]:
```
5/3 #bölme
```

Out [28]: 1.6666666666666667

In [29]:
```
3.0*5
```

Out [29]: 15.0

Yazılar

In [16]:
```
strvar="Python"
print(strvar)
```

```
Python
```

In [2]:
```
strvar[0]
```

Out [2]: 'P'

In [4]:
```
strvar[-5]
```

Out [4]: 'y'

In [5]:
```
strvar[1]
```

Out [5]: 'y'

[] #tek bir eleman alınır [:] #başlangıç ve bitiş arasındaki elemanlar alınır (sonuncu alınmaz) [::]#başlangıç ve bitiş arasındaki elemanlar üçüncü değere göre atlanarak alınır

In [6]:
```
strvar[1:3] #3 dahil değil
```

Out [6]: 'yt'

In [7]:
```
strvar[0:5:2] #0'dan 5'e 2'şer atlayarak yazdır
```

Out [7]: 'Pto'

In [8]:
```
strvar[:] #tamamını alır
```

Out [8]: 'Python'

In [9]:
```
strvar[::3] #baştan sona 3'er atla
```

Out [9]: 'Ph'

In [10]:
```
len(strvar) #uzunluğu verir
```

Out [10]: 6

In [17]:
```
strvar = strvar + " öğren! "
print(strvar)
```

```
Python öğren!
```

In [18]:
```
print(strvar*5)
```

```
Python öğren! Python öğren! Python öğren! Python öğren! Python öğren!
```

strvar. nokta koyup tab tuşuna basınca fonksiyonlara ulaşılabilir, seçip () ortasında shift+tab yapınca instruction verir

In [20]:
```
strvar.upper()
```

Out [20]:
```
'PYTHON ÖĞREN! '
```

In [21]:
```
strvar.lower()
```

Out [21]:
```
'python öğren! '
```

In [22]:
```
strvar.split()
```

Out [22]:
```
['Python', 'öğren!']
```

In [24]:
```
strvar.split("o") #o'dan ayır
```

Out [24]:
```
['Pyth', 'n öğren! ']
```

In [25]:
```
strvar1="Pyhton ogren!"
strvar1.split(sep="o",maxsplit=1) #kaç split yapılacağı da belirlenebilir
```

Out [25]:
```
['Pyht', 'n ogren!']
```

Boolean

In [27]:
```
a = True
print(type(a))
b=False
print(a)
print(b)
c="True" #string olup aynı işlevi görmez ama çıktı olarak aynı görünür dikkat
print(type(c))
print(c)
```

```
<class 'bool'>
True
False
<class 'str'>
True
```

In [28]:
```
yas1 = 20
yas2 = 18
print(yas1>18)
print(yas2>18)
```

```
True
False
```

In [30]:
```
yas1 == 18
```

Out [30]:
```
False
```

In [31]:
```
yas2 == 18 #== eşit midir?
```

Out [31]:
```
True
```

In [32]:
```
yas2 != 18 #!= eşit değil midir?
```

Out [32]:
```
False
```

In [33]:
```
not yas1 > 18 #başa not koymak da olumsuzluğu verir
```

Out [33]:
```
False
```

List & Set

In [44]:
```
liste = ["a", "b", "c", "d", "e", "a"]
print(liste)
```

```
['a', 'b', 'c', 'd', 'e', 'a']
```

In [45]:
```
#liste = liste + "f" çalışmaz
liste = liste +["f"]
print(liste)
```

```
['a', 'b', 'c', 'd', 'e', 'a', 'f']
```

In [46]:
```
liste[0]
```

Out [46]:
```
'a'
```

In [47]:
```
liste.append("g") #ekler
liste
```

Out [47]: ['a', 'b', 'c', 'd', 'e', 'a', 'f', 'g']

In [48]:
```
liste.pop() #son elemanı atar
liste
```

Out [48]: ['a', 'b', 'c', 'd', 'e', 'a', 'f']

In [49]:
```
liste.pop(5) #5. indexli elemanı atar
liste
```

Out [49]: ['a', 'b', 'c', 'd', 'e', 'f']

In [50]:
```
sayilar = [123,12321,312,4535,335,345,1,1]
sayilar.sort() #küçükten büyüğe sıralama
sayilar
```

Out [50]: [1, 1, 123, 312, 335, 345, 4535, 12321]

In [51]:
```
sayilar.reverse() #tersine çevirme
sayilar
```

Out [51]: [12321, 4535, 345, 335, 312, 123, 1, 1]

In [53]:
```
set(sayilar) #sete aktarma, tekrarlar kalkar, sıralanır
```

Out [53]: {1, 123, 312, 335, 345, 4535, 12321}

In [54]:
```
sayilar = [123,12321,312,4535,335,345,1,1]
set(sayilar)
```

Out [54]: {1, 123, 312, 335, 345, 4535, 12321}

Tuple

In [1]:
```
liste = ["a", "b", "c", "d", "e", "a"]
print(liste)
tup = ("a", "b", "c", "d", "e", "a")
print(tup)
```

```
['a', 'b', 'c', 'd', 'e', 'a']
('a', 'b', 'c', 'd', 'e', 'a')
```

In [2]:
```
liste[0] = 12312
liste
```

Out [2]: [12312, 'b', 'c', 'd', 'e', 'a']

In [3]:
```
tup[0] = 12312
tup #çalışmaz, değiştirilemez immutable
```

```
---------------------------------------------------------------------------TypeError                                 Traceback (most recent call last)Cell In[3], l
----> 1 tup[0] = 12312
      2 tup
TypeError: 'tuple' object does not support item assignment
```

In [4]:
```
tup.count("a")
```

Out [4]: 2

In [5]:
```
tup.count("d")
```

Out [5]: 1

In [6]:
```
tup.count(True)
```

Out [6]: 0

In [7]:
```
tup.index("a") #ilki verir
```

Out [7]: 0

In [8]:
```
tup.index("d")
```

Out [8]: 3

In [9]:
```
tup.index(True) #hata verir
```

```
---------------------------------------------------------------------------ValueError                                Traceback (most recent call last)Cell In[9], l
----> 1 tup.index(True)
ValueError: tuple.index(x): x not in tuple
```

Çok Boyutlu Veri Tipleri DICTIONARY

In [12]:
```
dict1 = {"isim": "Yavuz","yas": 24, "lokasyon": "Pendik"}
```

```
dict1
```

Out [12]:
```
{'isim': 'Yavuz', 'yas': 24, 'lokasyon': 'Pendik'}
```

In [13]:
```
dict1 = {
    "isim": "Yavuz", #= kullanılmıyor, : kullanılıyor dikkat!
    "yas": 24,
    "lokasyon": "Pendik"
}
dict1
```

Out [13]:
```
{'isim': 'Yavuz', 'yas': 24, 'lokasyon': 'Pendik'}
```

In [17]:
```
dict2 = {
    "isim": "Yavuz",
    "yas": 24,
    "lokasyon": {
        "yasadigi_sehir" : "Londra",
        "dogdugu_sehir" : "İstanbul"
    }
}
dict2
```

Out [17]:
```
{'isim': 'Yavuz',
 'yas': 24,
 'lokasyon': {'yasadigi_sehir': 'Londra', 'dogdugu_sehir': 'İstanbul'}}
```

In [19]:
```
dict2["yas"]
```

Out [19]:
```
24
```

In [22]:
```
dict2["yasadigi_sehir"] #çalışmaz
```

```
---------------------------------------------------------------------------KeyError                                  Traceback (most recent call last)Cell In[22],
----> 1 dict2["yasadigi_sehir"]
KeyError: 'yasadigi_sehir'
```

In [23]:
```
dict2["lokasyon"]["yasadigi_sehir"]
```

Out [23]:
```
'Londra'
```

In [25]:
```
dict2.get("lokasyon").get("yasadigi_sehir")
```

Out [25]:
```
'Londra'
```

In [26]:
```
dict2.get("lokasyon")
```

Out [26]:
```
{'yasadigi_sehir': 'Londra', 'dogdugu_sehir': 'İstanbul'}
```

In [27]:
```
dict2.keys()
```

Out [27]:
```
dict_keys(['isim', 'yas', 'lokasyon'])
```

In [28]:
```
dict2.values()
```

Out [28]:
```
dict_values(['Yavuz', 24, {'yasadigi_sehir': 'Londra', 'dogdugu_sehir': 'İstanbul'}])
```

In [29]:
```
dict2.items()
```

Out [29]:
```
dict_items([('isim', 'Yavuz'), ('yas', 24), ('lokasyon', {'yasadigi_sehir': 'Londra', 'dogdugu_sehir': 'İstanbul'})])
```

In [ ]:
```

```