Fonksiyonlarda Parametre Türleri

Bu derste fonksiyonlara parametre vermenin farklı yollarını öğrenmeye çalışacağız. İsterseniz hemen başlayalım.

Parametrelerin Varsayılan Değerleri

Biliyorsunuz önceki konularda şöyle bir fonksiyon tanımlamıştık.

In [3]:
```
def selamla(isim):
    print("Selam",isim)
```

In [4]:
```
selamla("Murat")
```

```
Selam Murat
```

In [5]:
```
selamla("Serhat")
```

```
Selam Serhat
```

In [6]:
```
selamla() # Böyle bir kullanım hata verecektir.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-8fe4760a9706> in <module>()
----> 1 selamla() # Böyle bir kullanım hata verecektir.

TypeError: selamla() missing 1 required positional argument: 'isim'
```

Ancak biz eğer bir parametrenin değerini varsayılan olarak belirlemek istersek, bunu varsayılan değerler ile yapabiliriz. Varsayılan değerleri anlamak için **selamla** fonksiyonunu varsayılan parametre değeri ile yazalım.

In [7]:
```
def selamla(isim = "İsimsiz"):
    print("Selam",isim)
```

In [9]:
```
selamla() # Hiç bir değer göndermedik. "isim" parametresinin değeri varsayılan olarak "İsimsiz" olarak belirlendi
```

```
Selam İsimsiz
```

In [10]:
```
selamla("Serhat") # Değer verirsek varsayılan değerin yerine bizim verdiğimiz değer g eçer.
```

```
Selam Serhat
```

İşte bu kadar ! Peki birçok parametreye sahip olursak ne olacak ? Bir fonksiyon daha tanımlayalım

In [12]:
```
def bilgilerigöster(ad = "Bilgi Yok",soyad = "Bilgi Yok",numara =  "Bilgi Yok"):
    print("Ad:",ad,"Soyad:",soyad,"Numara:",numara)
```

In [13]:
```
bilgilerigöster() # Bütün parametreler varsayılan değerle ekrana basılacak.
```

```
Ad: Bilgi Yok Soyad: Bilgi Yok Numara: Bilgi Yok
```

In [14]:
```
bilgilerigöster("Mustafa Murat","Coşkun") # ad ve soyad değerini verdik ancak numara
 parametresi varsayılan değer oldu.
```

```
Ad: Mustafa Murat Soyad: Coşkun Numara: Bilgi Yok
```

Ancak böyle bir durumda argümanları gönderirken değerleri **sıralı** vermemiz gerekiyor. Peki sadece **numara** parametresine değer vermek istersek ne yapacağız ?

In [15]:
```
bilgilerigöster(numara = "123456") # numara parametresini özel olarak belirtiyoruz.
```

```
Ad: Bilgi Yok Soyad: Bilgi Yok Numara: 123456
```

In [16]:
```
bilgilerigöster(ad = "Mustafa Murat",numara = "123456")
```

```
Ad: Mustafa Murat Soyad: Bilgi Yok Numara: 123456
```

Aslında biz varsayılan değerleri kursumuzun en başlarında görmüştük. **print fonksiyonunun sep parametresini hatırlayalım.**

In [17]:
```
print("Mustafa","Murat","Coşkun") # sep parametresine değer vermeyince varsayılan ola
rak boşluk karakteri verildi.
```

```
Mustafa Murat Coşkun
```

In [18]:
```
print("Mustafa","Murat","Coşkun",sep = "/") # sep parametresine özel olarak değer ata
dık.
```

```
Mustafa/Murat/Coşkun
```

İsterseniz print fonksiyonun nasıl yazıldığına **help** fonksiyonu sayesinde bakalım.

In [20]:
```
help(print)  # sep parametresine varsayılan olarak boşluk değeri verildiğini görebili
yoruz.
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Esnek Sayıda Değerler

Biliyorsunuz bir fonksiyon yazıldığında özel olarak kaç tane parametresi olacağını önceden belirtmemiz gerekiyor. Örneğin, bir **toplama** fonksiyonu yazalım.

In [23]:
```
def toplama(a,b,c):
    print(a+b+c)
```

In [24]:
```
toplama(3,4,5)
```

```
12
```

In [26]:
```
toplama(3,4,5,6) # 4 tane argüman veremeyiz.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-aeb23284c8f3> in <module>()
----> 1 toplama(3,4,5,6) # 4 tane argüman veremeyiz.

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

Eğer bu fonksiyonu 4 argüman alacak şekilde tanımlamak istersek, tekrardan tanımlamamız gerekiyor.

In [28]:
```
def toplama(a,b,c,d):
    print(a+b+c+d)
```

In [29]:
```
toplama(3,4,5,6)
```

```
18
```

In [30]:
```
toplama(3,4,5) # Ancak bu sefer de 3 argüman veremiyoruz.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-35c9fb4ed348> in <module>()
----> 1 toplama(3,4,5) # Ancak bu seferde 3 argüman veremiyoruz.

TypeError: toplama() missing 1 required positional argument: 'd'
```

Peki ben bir fonksiyonu esnek sayıda argümanla kullanmak istersem ne yapacağım ? Bunun için de **Yıldızlı Parametre** kullanmam gerekiyor. Kullanımı şu şekildedir;

In [39]:
```
def toplama(*parametreler): # Artık parametreler değişkenini bir demet gibi kullanabilirim.
    toplam =  0
    print("Parametreler:",parametreler)
    for i in parametreler:
        toplam += i
    return toplam
```

In [40]:
```
print(toplama(3,4,5,6,7,8,9,10))
```

```
Parametreler: (3, 4, 5, 6, 7, 8, 9, 10)
52
```

In [41]:
```
print(toplama())
```

```
Parametreler: ()
0
```

In [38]:
```
print(toplama(1,2,3))
```

```
Parametreler: (1, 2, 3)
6
```

print fonksiyonunu tekrar hatırlayacak olursak aslında **print** fonksiyonu bu şekilde tanımlanmış bir fonksiyondur. Çünkü biz print fonksiyonuna istediğimiz sayıda argüman gönderebiliyorduk.

In [42]:
```
print(3,4,5,6)
```

```
3 4 5 6
```

In [43]:
```
print("Elma","Armut")
```

```
Elma Armut
```

Bu konuda gördüğümüz gibi Python'ın bu özelliklerini kullanarak daha esnek fonksiyonlar yazabiliriz. Bir sonraki konuda **Global ve Yerel değişkenleri** öğrenmeye çalışacağız.

In [ ]:
```

```