Lambda İfadeleri

Bu konuda **lambda ifadelerini(expression)** öğrenmeye çalışacağız. **lambda ifadeleri** fonksiyonlarımızı oluşturmak için Pythonda bulunan pratik bir yöntemdir ve gerektiği yerlerde bu ifadeleri kullanabiliriz. Biliyorsunuz listelerimizi oluşturmak için **List Comprehension** yöntemini kullanabiliyorduk. İsterseniz **List Comprehension** yöntemini hatırlayalım.

In [1]:
```
liste1 = [1,2,3,4,5]
liste2 = list()
for i in liste1: # Bu klasik liste oluşturma yöntemi
    liste2.append(i*2)
print(liste2)
```

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

In [4]:
```
liste3 = [1,2,3,4,5]
liste4 = [i * 2 for i in liste3] # List Comprehension
print(liste4)
```

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

Aynı buradaki gibi bir fonksiyonu da tek satır halinde **lambda ifadeleriyle** oluşturabiliriz. İlk önce yapısına bakalım sonra örneklerimize geçelim.

```
etiket = lambda parametre1,parametre2.... :  İşlem
```

Bu yapıdan henüz bir şey anlamamış olabiliriz. İsterseniz örneklerimizle ***lambda ifadelerini*** anlamaya çalışalım. Bir tane iki ile çarpma görevini yerine getiren fonksiyon yazalım.

In [7]:
```
def ikiyleçarp(x): # Klasik fonksiyon tanımlama
    return x * 2
```

In [8]:
```
print(ikiyleçarp(2))
```

```
4
```

In [9]:
```
# Şimdi de bu fonksiyonu lambda ifadelerini kullanarak tek satırda yazalım.

ikiyleçarp = lambda x : x * 2 # x parametre x* 2 return ifadesi ve ikiyleçarp değeri
 de bir etikettir(değişken gibi düşünelim)
```

In [11]:
```
ikiyleçarp(3) # Buradaki 3 argümanı lambda ifadesindeki x'in yerine geçiyor.
```

Out[11]:
```
6
```

In [12]:
```
def toplama(a,b,c):
    return a + b + c
```

In [13]:
```
print(toplama(3,4,5))
```

```
12
```

In [14]:
```
topla = lambda x,y,z : x + y + z
```

In [15]:
```
print(topla(3,4,5))
```

```
12
```

In [16]:
```
# Stringi ters çevirme
def terscevir(s):
    return s[::-1]
```

In [17]:
```
print(terscevir("Python Programlama"))
```

```
amalmargorP nohtyP
```

In [18]:
```
ters =  lambda s : s[::-1]
```

In [19]:
```
print(ters("Python Programlama"))
```

```
amalmargorP nohtyP
```

In [25]:
```
# çift mi

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

In [26]:
```
print(çiftmi(12))
```

```
True
```

In [27]:
```
print(çiftmi(13))
```

```
False
```

In [28]:
```
çifttek = lambda sayı :  sayı % 2 == 0
```

In [29]:
```
çifttek(34)
```

Out[29]:
```
True
```

In [30]:
```
çifttek(13)
```

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

İşte lambda ifadesini bu şekilde küçük fonksiyonlar için kullanabiliriz. **lambda ifadelerini** özellikle kısa bir fonksiyonu **def** ifadesiyle yazmanın zahmetli olduğu zamanlarda kullanılabilir.