**List Comprehension(Liste Anlama)**

Python'da liste anlama, bir diziden liste oluşturmanın daha kolay bir yoludur.

* Örneğin, bir dizeyi bir karakter listesine dönüştürmek istiyorsanız. Birkaç yöntem kullanabilirsiniz. Bunlardan bazılarını görelim:

Örnek:1

language = 'Python'

lst = list(language)

print(type(lst))

print(lst)

# <class 'list'>

# ['P', 'y', 't', 'h', 'o', 'n']

Bir bal şeklide aynı liste kavraması gerçekleştirelim.

Örnek:2

lst = [i for i in language]

print(type(lst)) # list

print(lst) # ['P', 'y', 't', 'h', 'o', 'n']

* Bir sayı listesi oluşturmak istiyorsak

Örnek:

numbers = [i for i in range(13)]

print(numbers)

#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Örnek:

numbers = [(i, i \* i) for i in range(13)]

print(numbers)

#[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81), (10, 100), (11, 121), (12, 144)]

* Liste anlama if ifadesi ile birleştirilebilir

Örnek:1

even\_numbers = [i for i in range(40) if i % 2 == 0]

print(even\_numbers)

#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]

Örnek:2

list\_of\_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

flattened\_list = [ number for row in list\_of\_lists for number in row]

print(flattened\_list)

#[1, 2, 3, 4, 5, 6, 7, 8, 9]

Örnek:3

odd\_numbers = [i for i in range(41) if i % 2 != 0]

print(odd\_numbers)

#[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39]

## **Lambda Function**

Lambda işlevi, adı olmayan küçük bir anonim işlevdir. Herhangi bir sayıda argüman alabilir, ancak yalnızca bir ifadeye sahip olabilir.

## **Creating a Lambda Function**

Bir lambda işlevi oluşturmak için lambda anahtar sözcüğünü, ardından bir parametre(ler) ve ardından bir ifade kullanırız.

Örnek:

def add\_two\_nums(a, b):

    return a + b

print(add\_two\_nums(2, 3)) #5

Kendi kendini çağıran lambada işlevi

Örnek:

add\_two\_nums  =  lambda  a , b : a  +  b

print ( add\_two\_nums ( 2 , 3 ))  #5

Örnek:

(lambda a, b: a + b)(2,3) #5

Örnek:

multiple\_variable = lambda a, b, c: a \*\* 2 - 3 \* b + 4 \* c

print(multiple\_variable(5, 5, 3))#22

## **Lambda Function Inside Another Function(Lambda İşlevi Başka Bir İşlev İçinde)**

Başka bir işlevin içinde bir lambda işlevi kullanma.

def power(x):

    return lambda n : x \*\* n

cube = power(2)(3)

print(cube) #2\*2\*2 8

two\_power\_of\_five = power(2)(5)

print(two\_power\_of\_five)#2\*2\*2\*2\*2 32