Functions/Fonksiyonlar

Tanım

Biz şimdiye dek karşılaştığımız print(), len(), type() ve open() gibi örnekler sayesinde ‘fonksiyon’ denen şeyi az çok tanıdığımızı söyleyebiliriz. Dolayısıyla fonksiyonun ne demek olduğunu şeklen de olsa biliyoruz ve hatta fonksiyonları kodlarımız içinde etkili bir şekilde kullanabiliyoruz.Fonksiyonların görevi, karmaşık işlemleri bir araya toplayarak, bu işlemleri tek adımda yapmamızı sağlamaktır. Fonksiyonlar çoğu zaman, yapmak istediğimiz işlemler için bir şablon vazifesi görür. Fonksiyonun özelliklerini madde madde inceleyelim ve genel bir fonsiyon tanımız

1) Her fonksiyonun bir adı bulunur ve fonksiyonlar sahip oldukları bu adlarla anılır.

print() fonksiyonu,

open()fonksiyonu,

type() fonksiyonu

input()fonksiyonu,

len() fonksiyonu en bilinenleri sayabiliriz bunların haricinde daha fazla fonksiyon bulunmaktadır.

2) Foksiyonların yanına eklenen parantez işaretlerinin içine, fonksiyonlara işlevsellik kazandıran bazı parametreler yazılır.

open(dosya\_adı,

print("Merhaba Dünya!"),

len("İstanbul")

3) Fonksiyonlar farklı sayıda parametre alabilir.

Örneğin print() fonksiyonu toplam 256 adet parametre alabilirken, input() fonksiyonu yalnızca tek bir parametre alır.

4) Fonksiyonların, isimli ve isimsiz parametreleri dışında, bir de varsayılan değerli parametreleri vardır.

Örneğin print() fonksiyonunun sep, end ve file parametreleri varsayılan değerli parametrelere birer örnektir. Eğer bir parametrenin varsayılan bir değeri varsa, o parametreye herhangi bir değer vermeden de fonksiyonu kullanabiliriz. Python bu parametrelere, belirli değerleri kendisi atayacaktır. Tabii eğer istersek, varsayılan değerli parametrelere kendimiz de başka birtakım değerler verebiliriz.

Defining a Function / Bir Fonksiyon Bildirmek

Bir fonksiyon yaptığımızda buna fonksiyon bildirme diyoruz. Örneği inceleyelim ve fonksiyonu çağıralım.

def function\_name():

    codes

## Function without Parameters / Parametresiz Fonksiyon

def add\_information():

    name='büşra'

    second\_name='fatma'

    lst\_name=' akkaya'

    sapce=' '

    information=name +sapce + second\_name + sapce+ lst\_name

    print(information)

add\_information

## Function Returning a Value-1/ Değer Döndüren Fonksiyon-1

Fonksiyon ayrıca değer döndürebilir. Eğer bir fonksiyonun dönüş ifadesi yoksa işlevin değeri yoktur. Yukarıdaki fonksiyonları return kullanarak yeniden yazalım. Artık fonksiyonu çağırdığımızda ve yazdırdığımızda bir fonksiyondan bir değer alıyoruz.

def add\_two\_numbers ():

    num\_one = 2

    num\_two = 3

    total = num\_one + num\_two

    return total

print(add\_two\_numbers())

# 5

## Function with Parameters/Paremetrelerle Fonksiyon

Bir fonksiyonda farklı veri tiplerini (sayı, string, boolean, list, tuple, dct veya set) parametre olarak iletebiliriz.

* Tek Parametre: Eğer fonksiyonumuz bir parametre alıyorsa, fonksiyonumuzu bir argüman ile çağırmalıyız.

Örnek:

def greetings (name):

    message = name + ', welcome to Python for Everyone!'

    return message

print(greetings('Büşra'))

#Büşra, welcome to Python for Everyone!

Örnek:

def sum\_square(x):

    return x\*x + x\*x

print(sum\_square(2))

#8

Örnek:

def square\_number(x):

    return x \* x

print(square\_number(2))

#4

Örnek:

def area\_of\_circle (r):

    PI = 3.14

    area = PI \* r \*\* 2 #(100\*3,14)

    return area

print(area\_of\_circle(10))

def sum\_of\_numbers(n):

    total = 0

    for i in range(n+1):

        total+=i

    print(total)

print(sum\_of\_numbers(10)) # n=10 , 55

print(sum\_of\_numbers(100)) # n=100 , 5050

* İki Parametre: Bir fonksiyonun bir parametresi veya parametresi olabilir veya olmayabilir. Bir fonksiyon ayrıca iki veya daha fazla parametreye sahip olabilir. İki parametreli bir fonksiyonu kontrol edelim:

Genel gösterimi:

def function\_name(para1, para2):

    codes

    codes

  print(function\_name(arg1, arg2))

Örnek:

 def generate\_all (first\_name, last\_name):

    space = ' '

    full\_name = first\_name + space + last\_name

    return full\_name

 print('ad-soyad: ', generate\_all('Büşra','Akkaya'))

#ad-soyad:  Büşra Akkaya

Örnek:

def sum\_two\_numbers (num\_one, num\_two):

    sum = num\_one\*num\_one + num\_two\*num\_two

    return sum

print('Sum of square: ', sum\_two\_numbers(5, 8))

Örnek:

def calculate\_age (current\_year, birth\_year):

    age = current\_year - birth\_year

    return age;

print('Now Age: ', calculate\_age(1870, 1819))

#Now Age:  51

## Passing Arguments with Key and Value/Anahtar ve Değer ile Bağımsız Değişkenleri Geçme

Genel Gösterim

def function\_name(para1, para2):

    codes

    codes

print(function\_name(para1 = 'John', para2 = 'Doe'))

Örnek:

def print\_fullname(firstname, lastname):

    space = ' '

    full\_name = firstname  + space + lastname

    print(full\_name)

print(print\_fullname(firstname = 'Büşra', lastname = 'Akkaya'))

# Büşra Akkaya

Örnek:

def add\_two\_numbers (num1, num2):

    total = num1 + num2

    print(total)

print(add\_two\_numbers(num2 = 38, num1 = 8))

#46

## Function Returning a Value-2 / Değer Döndüren Fonksiyon-2

Bir fonksiyonla bir değer döndürmezsek, fonksiyonumuz varsayılan olarak None değerini döndürür. Bir işlevle bir değer döndürmek için, geri döndürdüğümüz değişkenin ardından return anahtar sözcüğünü kullanırız. Bir fonksiyondan her türlü veri tipini döndürebiliriz. Farklı örnekleri inceleyelim.

Örnek:

def print\_name(firstname):

    return firstname

print\_name('Büşra')

def print\_full\_name(firstname, lastname):

    space = ' '

    full\_name = firstname  + space + lastname

    return full\_name

print\_full\_name(firstname='Büşra', lastname='Akkaya')

# 'Büşra Akkaya'

Örnek:

def add\_two\_numbers (num1, num2):

    total = num1 + num2

    return total

print(add\_two\_numbers(2, 3))

def a\_age (year, birth\_year):

    age = year - birth\_year

    return age;

print('Age: ', a\_age(2019, 1819))

#5

Age:  200

Örnek:

def find\_even\_numbers(n):

    evens = []

    for i in range(n + 1):

        if i %5 == 0:

            evens.append(i)

    return evens

print(find\_even\_numbers(10))

#[0, 5, 10]

## Function with Default Parameters/Varsayılan Parametrelerle Fonksiyon

Fonksiyonu çağırdığımızda bazen varsayılan değerleri parametrelere iletiriz. Fonksiyonu çağırırken argümanları iletmezsek, onların varsayılan değerleri kullanılacaktır.

Genel Gösterim:

def function\_name(param = value):

    codes

function\_name()

function\_name(arg)

## Arbitrary Number of Arguments/Argümanların Keyfi Sayısı

Eğer fonksiyonumuza aktardığımız argüman sayısını bilmiyorsak, parametre isminin önüne \* ekleyerek isteğe bağlı sayıda argüman alabilen bir fonksiyon oluşturabiliriz.

Genel Gösterim:

def function\_name(\*args):

    codes

function\_name(param1, param2, param3,..)

Örnek:

def sum\_all\_nums(\*nums):

    total = 0

    for num in nums:

        total += num

    return total

print(sum\_all\_nums(14, 3, 5))

#22

Örnek:

def sum\_all\_nums(\*nums):

    total = 10

    for num in nums:

        total += num

    return total

print(sum\_all\_nums(14, 3, 5))

#32

Yukarıdaki iki örnekte benzerdir total değerlerin önemi vurgulamak için iki örneği de incelemenizi tavsiye ederim.

## Default and Arbitrary Number of Parameters in Functions/Fonksiyonlarda Varsayılan ve Keyfi Parametre Sayısı

def generate\_groups (team,\*args):

    print(team)

    for i in args:

        print(i)

print(generate\_groups('Ekip','Büşra','Burak','Deniz','Eyüp'))

#Ekip

#Büşra

#Burak

#Deniz

#Eyüp

## Function/Başka Bir Fonksiyonun Parametresi Olarak Fonksiyon

def  do\_something ( f , x ):

    return  f ( x )

def  square\_number ( n ):

     return  n  \*  n

print (do\_something ( square\_number , 3)) # 27