Decorators(dekorativlər)

May 23, 2022

1 Dekoratorlar

Bundan əvvəlki bəhsdə funksiyalar və modulları öyrəndik.Modullar və funksiyalar haqqında ümumi təsəvvürünüz yarandı.İn isə funksiyalarda istifadə olunan dekoratorları öyrənək.

1.0.1 Dekoratorlar nədir!

Dekorativ funksiyalar, Pythonda istifadəçiyə quruluşunu dəyişdirmədən mövcud bir obyektə yeni funksionallıq əlavə etməyə imkan verən dizayn nümunəsidir.Dekorativlər adətən təmin edəcəyimiz funksiyadan əvvəl çağırılır.Pythonda funksiyalar birinci sinif vətəndaşlardır(first class citizens).Bu o deməkdir ki, arqument kimi ötürülərək, bir funksiyadan geri qaytarılmış, dəyişdirilmiş və dəyişənə tapşırılan kimi əməliyyatlara dəstək olurlar. Bu Python dekorativlərini yaratmağı öyrənməzdən əvvəl başa düşmək üçün əsas bir anlayışdır.Dekorativlərə Flask framework -də daha çox istifadə olunur.Uzan uzadı kod sətirləri yazmaq yerinə,dekorativ funksiyalardan istifadə edə bilərsiniz.

[57]:
```
def add_num(var):
    return var + 1
add_one = add_num
add_one(4)
```

[57]: 5

Yuxarıdakı funksiyada yeni kod nümunəsi yoxdur.add_num funksiyası hazırladıq və ardından bir (var) arqumenti almasını təmin edib həmin dəyəri return ilə aldıq

Və ya funksiya daxilində funksiya hazırlayaq

[58]:
```
#Defining Functions Inside other Functions
def add_num(var):
    def add_one(var):
        return var + 1


    result = add_one(var)
    return result
add_num(4)
```

[58]: 5

Yuxarıdakı add_num əsas add_one funksiyası isə aparıcıdır.

Və ya iki ayrı funksiya tərtib edib funksiya parametrlərini digərinə ötürmək

[59]:
```
#Passing Functions as Arguments to other Functions
def add_num(var):
    return var + 1

def func_call(function):
    var = 4 #sabit dəyər
    return function(var)
func_call(add_num)
```

[59]: 5

Yuxarıda add_num funksiyası tərtib edib var dəyişəni üzərinə 1ədədini əlavə edirik.func_call funksiyası vasitəsilə ilk funksiya üçün dəyişənə 4 rəqəmini mənimsətdik daha sonra return vasitəsilə ilk funksiyamizdan aldığımız toplama əməlini yerinə yetiririk

Həmçinin bir funksiya yeni bir törəmə funksiya hazırlaya bilər

[60]:
```
#Functions Returning other Functions
i = 4
def some():
    def var():
        return i*2
    return var
k = some()
k()
```

[60]: 8

[61]:
```
#Creating Decorators
def _decorator(function):
    def wrapper():
        func = function()
        _make = func.upper()
        return _make
    return wrapper
```

[62]:
```
def uppercase():
    return 'azerbaijan'
decorate =_decorator(uppercase)
decorate()
```

[62]: 'AZERBAIJAN'

Yuxarıdakı nümunədə dekorativ funksiyanı hazırladıq._decorator funksiyası daxilində wrapper funksiyası hazırlayıb daha sonra _decorator funksiyasından gələn function arqumentini func dəy-

işəninə əlavə etdik.Aldığımız parametr sətir tipində olduğu üçün sətir tipi metodlarından rahatlıqla istifadə edə bilərik.Və əsas funksiyamız wrapper funksiyasını return ifadəsinə ötürürük.

Və ya başqa yazılışla @ - işarəsindən istifadə olunur.və dekorativ funksiyanı istifadə etmək üçün tətbiq edəcəyimiz funksiyadan əvvəl yazılmalıdır.

[63]:
```
def _decorator(function):
    def wrapper():
        func = function()
        _make = func.upper()
        return _make
    return wrapper

@_decorator #dekorativ funksiyamızı çağırırıq
def uppercase():
    return 'azerbaijan'
uppercase()
```

[63]:
```
'AZERBAIJAN'
```

Yuxarıdakı nümunədə @_decorator - dekorativ funksiya olub özündən sonra gələn istənilən funksiyaya tətbiq oluna bilinir

[64]:
```
def _decorator(function):
    def wrapper():
        func = function()
        _make = func.upper()
        return _make
    return wrapper
def _decorator2(function):
    def wrapper():
        func = function()
        split = func.split()
        return split
    return wrapper

@_decorator2
@_decorator

def test():
    return 'machine learning'
test()
```

[64]:
```
['MACHINE', 'LEARNING']
```

[ ]:
```

```

[1]:
```
def Fahrenheit(temprature):
    """
    funksiya selsi cinsindən temperaturu fahrenheyt cinsinə çevirir
    """

    def wrapper(celsius):
        F = (celsius * 9/5) + 32
        temprature(celsius)
        return ('Converted :{:.2f}F'.format(F))
    return wrapper

def Celsius(temprature):
    """
    funksiya fahrenheit cinsindən temperaturu selsi cinsinə çevirir
    """
    def wrapper(fahrenheit):
        C = (fahrenheit - 32) * 5/9
        temprature(fahrenheit)
        return ('Converted :{:.2f}C'.format(C))
    return wrapper
```

[2]:
```
@Celsius
def convert(t):
    return t
convert(12)
```

[2]: 'Converted :-11.11C'

[3]:
```
@Fahrenheit
def convert(t):
    return t
convert(10)
```

[3]: 'Converted :50.00F'

[ ]: