Dekorátor

Jó leírások: [[1]], [[2]]

A **dekorátorok** módosítják függvények vagy osztályok kódját!

Scope (hatókör), névtér

Minden új függvény fölépíti saját névterét, egy szótár formájában. Íme a globális és egy lokális névtér:

```
>>> x = 2
>>> print globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', 'x': 2, '__doc__': Non
>>> def fn():
...     y = 3
...     print locals()
...
>>> fn()
{'y': 3}
```

Ha egy név lokálisan nem lett létrehozva, a tartalmazó névterekben keresi egyre kijjebb haladva (itt az x nem lett lokálisan létrehozva, de elérhet?):

```
>>> def fn():
...     y = 3
...     print "x, y:", x, y
...     print locals()
...
>>> fn()
x, y: 2 3
{'y': 3}
```

Kérdés: Mi történik az alábbi kód hatására? magyarázzuk meg, mi történik:

```
>>> def fn():
...     y = 3
...     print "x, y:", x, y
...     x = 5
...     print locals()
...
>>> fn()
```

És mi történik, ha a 3. és 4. sort fölcseréljük?

A névtér törl?dik a függvény lefutása után, magasabb szinten az alacsonyabb szint? névterek nem érhet?k el:

```
>>> def fn():
...     z = 1
...
>>> print z
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'z' is not defined
```

A **függvény lezárása** a Pythonnak azt a képességét jelenti, hogy a nem globális névtérben definiált függvény **emlékszik** a definiálás pillanatában érvényes bennfoglaló névterekre:

```
>>> def kulso(x):
```

```
...     def belso():
...         print x
...     return belso
...
>>> f1 = kulso(3)
>>> f2 = kulso(5)
>>>
>>> f1()
3
>>> f2()
5
```

Dekorálunk

A küls? függvény az argumentumában átadott függvény eredményét duplázza, és ezt a duplázó függvényt adja vissza. Ezzel **dekorálunk** két különböz? függvényt!

```
>>> def kulso(fn):
...     def belso(*args):
...         print "belül vagyunk"
...         return 2*fn(*args)
...     return belso
...
>>> dekoralt = kulso(fv)
>>> dekoralt(5)
belül vagyunk
10
>>> def fadd(a, b):
...     return a + b
...
>>> dekoralt = kulso(fadd)
>>> dekoralt(4, 5)
belül vagyunk
18
```

A `fv = kulso(fv)` rövidítése a `@kulso` a fv függvénydefiníció elé írásával:

```
>>> @kulso
... def fmul(a, b):
...     return a * b
...
>>> fmul(3, 5)
belül vagyunk
30
```

Dekorátor lehet olyan osztály is, mely **meghívható**, azaz amelyben létezik `__call__` függvény:

```
>>> class Dekorator(object):
...     def __init__(self, f):
...         print "Dekorátor konstruktor"
...         print "meghívjuk a fv-t:", f()
...     def __call__(self):
...         print "Dekorátor meghívása"
...
>>> @Dekorator
... def fn():
...     print "az fn meghívása"
...
Dekorátor konstruktor
meghívjuk a fv-t: az fn meghívása
```

```
None
>>> fn()
Dekorátor meghívása
```