04c pisanje modulov

January 28, 2024

0.1 Kako pišemo module

Napišimo program, ki bo vseboval konstanto `odgovor`, ki bo imela vrednost 42, in funkcijo, ki računa Fibonaccijeva števila.

```
odgovor = 42

def fibonacci(n):
    a = b = 0
    for i in range(n):
        a, b = b, a+b
    return a
```

Program shranimo pod imenom fibo.py.

To je to. V drugem programu lahko rečemo

```
import fibo
print("Odgovor je", fibo.odgovor)
print("Deseto Fibonaccijevo število pa je", fibo.fibonacci(10))
```

Tudi vse ostale finte, na primer, `from fibo import odgovor`, delujejo.

Modul ni nič drugega kot program, ki ga uvozimo.

Tudi vsi drugi programi, ki ste jih napisali doslej, so hkrati moduli: lahko jih uvozite. Pazite le na tole: ko modul uvozimo, se ta v resnici izvede, čisto tako, kot bi se izvajal program. Vse, kar se v tem programu-modulu definira, ostane definirano in se nahaja v modulovem imenskem prostoru. Vendar se zgodi tudi vse ostalo, kar piše v modulu: če pri izvajanju naleti na print(“Foo”) se bo ob uvozu modula izpisalo Foo. Če program-modul vsebuje klic funkcije (in ne le definicij), se bo ta funkcija poklicala tudi ob uvozu.

Kje Python išče module? Navadno v trenutnem direktoriju, poleg tega pa še v drugih. Na Windowsih v, recimo, c:-packages. Več o tem si lahko preberete v dokumentaciji.

Tudi sicer ne bomo rinili prav globoko v module, samo še eno stvar omenimo, da vas ne bo presenetila. Ko boste prvič uvozili modul, se bo poleg datoteke s končnico .py pojavila še ena, z enakim imenom, a končnico .pyc. Python si vanjo “prevede” (pravzaprav bi smeli pisati brez narekovajev, temu se v resnici reče prevajanje) v obliko, v kateri ga bo lahko naslednjič hitreje uvozil. Pri manjših modulih se to ne pozna, pri velikem številu velikih modulov pa. Če vas moti, jo lahko pobrišete; drugič se bo pač spet pojavila ponovno.

[ ]: