04b uvažanje modulov

January 28, 2024

0.1 Uvažanje modulov

Doslej smo spoznali le prgišče funkcij - `input`, `len`, `sqrt`. V resnici pa skupaj s Pythonom dobimo tisoče funkcij za vse mogoče reči, od pošiljanja pošte in kriptografije do predvajanja zvoka in brskanja po datotekah .zip. Še večja gora funkcij obstaja na spletu: karkoli si zamislite napisati, skoraj gotovo je nekdo - če le gre za dovolj splošno reč - že potreboval kaj takšnega ter to tudi napisal in vam dal na razpolago.

Da se ne bi izgubili v gori funkcij (in drugih reči), jih moramo nekako urediti. Funkcije pridejo lepo zapakirane v škatlice – včasih celo tako, da so znotraj škatlice nove škatlice. Škatlici rečemo *modul*.

Doslej smo uporabljali, na primer, funkcije iz modula `math`, tako da smo jih uvozili z `from math import *`. Pri poučevanju programiranja se izogibam temu, da bi morali tipkati kakšne fraze, ki jih ne razumete (kar je eden od glavnih razlogov, zakaj se učimo programiranja v Pythonu in ne v jeziku, kjer bi morali na prvem predavanju definirati razrede in pisati `public static void main(String args[])`). `from math import *` je bila ena redkih in čas je, da izvemo, za kaj gre. Ter tudi čas, da se te fraze odvadimo in se naučimo pisati kot se spodobi.

0.1.1 Uvažanje modula

Modul z matematičnimim funkcijami, `math`, uvozimo tako:

[17]:
```
import math
```

S tem v bistvu dobimo novo spremenljivko, `math`.

[18]:
```
math
```

[18]:
```
<module 'math' from '/Users/janez/miniconda3/envs/prog/lib/python3.7/lib-
dynload/math.cpython-37m-darwin.so'>
```

Tole je spremenljivka najbolj čudnega tipa doslej. Ni preprosto števil (`int`, `float`) ali niz (`str`) ali seznam ali terka (`list`, `tuple`), temveč spremenljivka vrste "modul" ("module").

Modulov ne moremo seštevati ali množiti. Niti izpisati bolj ali manj ne. Kaj pa potem znajo? Kaj je njihov superpower? To, da vsebujejo stvari. Modul `math` vsebuje funkcije `sin`, `radians` in `sqrt`, poleg tega pa tudi kakšno konstanto, na primer `pi`. Do vseh teh stvari pridemo tako, da k spremenljivki, ki predstavlja modul, dodamo piko in ime stvari. Tako je `math.sin` funkcija, ki izračuna sinus in `math.pi` konstanta *pi*.

[19]:
```
math.pi
```

[19]:
```
3.141592653589793
```

[20]:
```
math.sin(math.pi / 4)
```

[20]:
```
0.7071067811865475
```

Uvozimo še en modul, `os`.

[21]:
```
import os
```

V `os` so funkcije, s katerimi lahko ustvarimo direktorij ali pobrišemo datoteko. Poleg tega pa vsebuje še nekaj zanimivega: modul. Modul `path` je modul znotraj modula `os`. Vsebuje, recimo, funkcijo `splitext`, ki za podano ime datoteke vrne osnovo in končnico.

[22]:
```
osnova, koncnica = os.path.splitext("nek_film.avi")
```

[23]:
```
osnova
```

[23]:
```
'nek_film'
```

[24]:
```
koncnica
```

[24]:
```
'.avi'
```

Tako se uvaža module.

0.1.2 Uvažanje posamičnih funkcij

Če funkcije določenega modula kličemo velikokrat in predvsem, če to počnemo v aritmetičnih izrazih, to postane nepregledno.

```
x = 2 * math.sin(math.radians(phi) + math.pi / 2) - 2 * math.cos(math.radians(alpha))
```

Boljše bi bilo imeti te funkcije pri roki brez ponavljajočega se `math`. Uvozimo jih z

[25]:
```
from math import sin, cos, radians, pi
```

[26]:
```
cos(pi / 4)
```

[26]:
```
0.7071067811865476
```

`from math import sin, cos, radians, pi` uvozi naštete funkcije in postanejo, pod temi imeni, dostopne programu. Imeli bomo torej imena `sin`, `cos`, `radians` in `pi`, ne pa tudi `tan` in `log`. Predvsem pa ne `math`. Se pravi:

- če uvozimo `import math`, imamo `math` in, recimo `math.cos`, ne pa `cos`. Uvozili smo pač `math`, ne `cos`.

- če uvozimo `from math import cos` pa imamo `cos` in ne `math` ali `math.cos`. Uvozili smo pač `cos` in ne `math`.

Obstaja še "preprostejši" način uvažanja.

[27]:
```
from math import *
```

Ta je podoben prejšnjemu, le da funkcij ne naštevamo. `*` predstavlja *vse*, kar je v modulu.

Prvi način, uvažanje modula, ima to prednost, da je v vsakem klicu funkcije jasno, odkod ta funkcija prihaja. Slabost so precej daljši izrazi.

Drugi način, uvažanje posamičnih funkcij skrajša izraze. Odkod prihaja kakšna funkcija, še vedno vidimo, vendar je potrebno za to pogledati na začetek programa, v `import`-e.

Tretji način je v splošnem slab. Izrazi so krajši, vendar za posamične funkcije ne vemo, odkod so prišle. Na ta način uvažamo kvečjemu modul `math` in nobenega drugega. Posebej v večjih projektih.

1 Kje uvažamo

Vedno na začetku programa. Ne kar tako, vmes.

Izjemoma: na začetku funkcije. To storimo, recimo, kadar ni jasno, ali bo modul možno uvoziti oziroma je uvažanje počasno, potrebujemo pa ga le v neki funkciji. Druga situacija, kjer bi to prišlo prav, je, če se dva modula uvažata vzajemno, eden drugega. To razrešimo tako, da določen modul uvozimo šele, ko ga potrebujemo.

To ni zakon, samo dogovor.

Uvažanje modulov se v resnici zgodi samo enkrat. Če petkrat napišemo `import math`, se bo modul v resnici uvozil le prvič.