Rukovanje modulima

Sve funkcije i promenljive koje kreiramo dok pokrećemo Python okruženje nestaju prilikom izlaska iz njega. Ako želimo da nam program „duže traje", te definicije funkcija i promenljivih smeštamo u poseban fajl ili skriptu. Takav fajl, u zavisnosti od namene, možemo nazvati i modul. Drugim rečima, modul je fajl koji sadrži Python naredbe i definicije.

Moduli se koriste kako bi se kod organizovao. Na primer, metode i funkcije koji se tiču povezivanja sa bazama podataka se smeštaju u modul za rukovanje bazama podataka i slično. Veći programi pisani u Pythonu mogu biti podeljeni na više modula, dok se moduli dalje grupišu zajedno po paketima.

Imena Python modula

Fajl modula čine ime i .py ekstenzija. Kada imamo fajl calculator.py – imamo modul po imenu calculator. Promenljiva __name__ sadrži ime modula na koji se odnosi. Glavni program koji se pokreće je takođe modul sa posebnim imenom __main__. U radnom direktorijumu je potrebno napraviti fajl po imenu calculator.py i test_calculator.py. Kod fajla za modul calculator.py je:

Primer koda za modul calculator.py

```
_version = 1.0
def addition(a,b):
    return a+b
def subtraction(a,b):
    return a-b
def multiplication(a,b):
    return a*b
def division(a,b):
    return a/b
def square(a,b):
    return a**b
def division_with_rounding(a,b):
    return a//b
def test_function():
    print('Test function!')
if __name__ == '__main__':
    test_function()
```

Ako želimo da iskoristimo ovaj modul, možemo ga uvesti u glavni program:

Primer uvođenja modula

```
import calculator
import sys

print(__name__)
print(calculator.__name__)
print(sys.__name__)
```

Što će dovesti do ispisa:

```
__main__
Calculator
sys
```

Ime modula koji se izvršava je uvek __main__, dok su ostali moduli nazvani po imenima fajlova u kojim se nalaze. Takođe, moduli se mogu uvesti u druge module naredbom import.

Traženje Python modula

Kada je modul uvezen, prevodilac prvo traži ime tog modula u ugrađenim modulima. Ako tu nije pronađen, potraga se nastavlja u listi putanja obezbeđenih u `sys.path` promenljivoj. Ova promenljiva sadrži listu stringova koja sadrži putanje u kojima Python prevodilac treba da traži module. Sastoji se od trenutnog radnog direktorijuma (putanje odakle se trenutni skript pokreće), putanja iz PYTHONPATH promenljive okruženja (environmental variable) i par dodatnih lokacija na disku. Ako prevodilac ne može da nađe ime module, vratiće ImportError grešku u program. Izvršavanje naredbe `sys.path` na svežoj instalaciji Win10 operativnog sistema daće sledeći ispis:

Radno okruženje

```
import sys
print(sys.path)
```

Import naredba

Postoji par načina korišćenja ove naredbe.

```
from module import *
```

Ovom naredbom ćemo uvesti sve definicije iz datog modula, sa jednim izuzetkom – objekti koji počinju jednostrukom donjom crtom (_) neće biti uvezeni u glavni program. Objekti definisani na takav način su namenjeni za internu upotrebu u tom modulu. Ovakav način uvoženja se ne preporučuje, jer može doći do zagušenja u imenskom prostoru i preklapanja objekta iz drugih modula sa istim imenom.

Primer upotrebe import naredbe gde se unose svi objekti

```
from math import *
print(pi)
```

Promenljivu pi nismo nigde definisali, ali kako smo uvezli sve objekte i definicije iz modula math, dobili smo i pristup njenoj vrednosti, koja je definisana upravo u tom modulu.

Kao što smo rekli, objekti definisani sa donjom crtom su objekti koji se ne uvoze u imenski prostor glavnog modula. Na sledećem primeru ćemo videti šta se sve tačno uvozi tom naredbom:

Primer

```
from calculator import *
print(locals())
```

Kao što se vidi iz ispisa funkcije `locals()` koja nam daje sve definicije u trenutnom modulu, nigde nema promenljive `_version`. Nije uvezena u trenutni imenski prostor, ali ostatak definicija i objekata jeste.

Kako bismo rešili problem uvoženja viška definicija i objekata koji nastaje izvršavanjem prethodne naredbe za uvoz modula, možemo tačno zadati koje objekte želimo i to sledećom sintaksom:

Primer upotrebe import naredbe sa odabranim objektima

```
from module import variable, function, variable
```

Ovo pravilo možemo primeniti na raniji primer:

Primer unošenja objekta iz modula

```
from math import pi
print(pi)
```

Takođe, na ovaj način je moguće uvesti i definicije koje počinju donjom crtom, ali se to smatra lošom praksom.

„Najčistiji" način rukovanja modulima se ogleda u pozivu:

```
import modul
```

Ovo podrazumeva uvoženje samo imena modula, bez uvoženja dodatnih definicija u imenski prostor. Definicijama i promenljivama u tom modulu pristupamo pomoću operatora (.). Ovo nam takođe omogućava da definišemo i promenljive, koje će imati isto ime kao i neke od definicija iz uvezenog modula. Zato je sledeći primer potpuno validan:

Primer

```
import math
pi = 3.14
print(pi)
print(math.pi)
```

Objašnjenje:

Možemo ručno definisati promenljivu pi ili iskoristiti konstantu pi iz modula math.

Dodeljivanje alijasa

Prilikom uvoženja modula, moguće je njegovo ime zameniti drugim – tačnije, dodati mu alijas i to na sledeći način:

Primer dodavanja alijasa

```
import math as m
print(m.pi)
```

Vežba

Importujte math modul i ispišite koren broja 4. Koren se izračunava metodom sqrt() modula math.

Radno okruženje

Rešenje vežbe možete naći na kraju lekcije.

Pitanje

Ako smo modul uvezli naredbom import modul, da li će funkcija locals() ispisati imena i definicije i iz tog modula?

- Da
- **Ne**

Objašnjenje:

*Funkcija locals() neće ispisati imena i definicije iz tako uvezenog modula jer na taj način smo uvezli samo njegovo ime, a definicijama u njemu pristupamo uz operator (.). Ako bismo želeli da funkcijom locals() izlistamo i objekte iz tog modula, naredbu od malopre treba promeniti u from modul import *.*

Izvršavanje modula

Moduli se mogu uvesti u druge module ili se oni sami mogu pokrenuti i izvršiti. Izvršavanje samih modula je obično ostavljeno za testiranje njegove stabilnosti i funkcionalnosti od strane autora. Ako se željeni modul izvršava kao skripta, onda njegov atribut __name__ postaje __main__.

Ako bismo naš početni primer (calculator.py) pokrenuli iz komandne linije komandom: python calculator.py, dobili bismo samo ispis (' Test function!') jer se samo test_function() i pokreće na ovaj način:

```
if __name__ == '__main__':
    test_function()
```

Tim delom koda smo rekli programu da, ako je modul koji pokreće calculator.py fajl imena __main__, pokrene testnu funkciju.

Isti taj modul (calculator.py) možemo i uvesti u glavni program i koristiti ga kao i do sada.

Rešenje vežbe

Rešenje

```
import math
x=4
y=math.sqrt(x)
print(y)
```

Ispis

```
2.0
```

Objašnjenje:

Metoda sqrt vraća kvadratni koren broja u obliku realnog broja.

Rezime

- Moduli se u naš program uvoze naredbom import. Ovom naredbom možemo uvesti čitav modul ili precizirati tačno koje promenljive, fukcije ili klase želimo.
- Kada ne želimo da omogućimo da se sve promenljive, funkcije i klase iz modula mogu uvesti u drugi program, ispred njihovih imena ćemo dodati donju crtu (_): _variable.
- Za proveru imena trenutnog modula u kojem se program nalazi koristimo promenljivu __name__.
- Alijasi su druga, proizvoljna imena modula koje sami zadajemo u formatu import module as alias.