Opsezi i prostori imena

Kako naši programi postaju sve veći i zahtevaju sve više funkcija i modula, potrebno je znati kako Python koristi promenljive da izvrši operacije.

Prostor imena je tipa rečnik, u kojem Python pretražuje potrebno ime promenljive i pomoću kojeg dobavlja objekat na koji pokazuje. Ovim se imena promenljivih mapiraju na objekte.

Za dobavljanje liste imena kojima Python može pristupiti koristimo dir([variable]) funkciju. Ako se pozove bez argumenata, ta funkcija daje listu svih promenljivih u našem imenskom prostoru. Ako po pokretanju Pythona izvršimo funkciju `print(dir())`, dobijamo:

```
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__',
'__package__', '__spec__']
```

Ovo predstavlja listu objekata dostupnih iz „svežeg" Python terminala.

Tipovi prostora imena

Od najvišeg ka najnižem, prostori imena u Pythonu mogu biti:

- ugrađeni imenski prostor (`built-in namespace`) – ovaj imenski prostor sadrži funkcije koje su ugrađene u Python;
- globalni imenski prostor (`global namespace`) – ovaj imenski prostor sadrži imena iz modula koji su uvezeni u skriptu/projekat; kreiran je kada uvezemo dati modul i postoji dok se skripta ne završi;
- lokalni imenski prostor (`local namespace`) – imena koja su definisana u datoj funkciji čine lokalni imenski prostor; ovaj imenski prostor je kreiran kada se funkcija pozove i završava kada funkcija vrati vrednost (završi izvršavanje).

Opsezi

Razlog zašto želimo da znamo više o našem imenskom prostoru jeste zato što želimo da znamo koje promenljive su nam dostupne u datom trenutku. U kontekstu imenskog prostora, opseg je kolekcija imena pridružena trenutnom okruženju. Python radno okruženje čine sledeći opsezi (od najvišeg do najnižeg):

- ugrađeni opseg (`built-in scope`);
- opseg na nivou modula (globalni) (`module scope`);
- obuhvatni opseg (nelokalni) (`enclosing scope`);
- lokalni opseg (`local scope`).

Kada je ime referencirano u Pythonu, prevodilac to ime traži u imenskom prostoru počevši od najnižeg opsega (lokalnog opsega) idući ka najvišem dokle god ili ne pronađe to ime ili ne prijavi `NameError` grešku.

Ugrađeni opseg (Built-in Scope)

Ugrađeni opseg sadrži sve funkcije koje su ugrađene u osnovu Pythona. Ovo podrazumeva funkcije kao što su `print()`, `dir()` i druge. Lista ugrađenih funkcija se može naći ovde. Kad god se pozove neka od ugrađenih funkcija, Python prevodilac će je tražiti u opsegu ugrađenih imena, pa je tako ovaj opseg i „najširi", što znači da će se, ako bi se ime koje postoji u ovom opsegu definisalo ponovo u opsegu nižeg nivoa, koristiti to ime iz nižeg nivoa opsega, jer, kao što smo ranije rekli, prevodilac ime traži počevši od najnižeg nivoa (lokalni opseg).

Primer

```
dir = 1
print(type(dir))
print(dir(dir))
```

Pri definisanju promenljive sa istim imenom kao i ugrađena funkcija `dir()` izgubili smo mogućnost da koristimo istu funkciju, ali nismo uklonili tu funkciju. U ovom slučaju prevodilac je u naredbi `dir(dir)` prvo pretražio lokalni opseg i pronašao dato ime. Ako želimo i dalje da koristimo ugrađeno ime `dir()`, moramo definisati u kom se opsegu nalazi. Ovo se rešava koristeći `builtins` modul (modul koji sadrži sve ugrađene funkcije, koji se mora ručno uvesti u program).

Primer

Radno okruženje

```
import builtins

dir = 1
print(builtins.dir(dir))
```

Opseg na nivou modula

Jedan korak ispod već ugrađenog opsega je opseg na nivou modula. Ovaj opseg se sastoji od imena definisanih u datom modulu koji se pokreće, ili u tom terminalu, izvan svih klasa i funkcija. Ovaj opseg je deo imenskog prostora na koji mislimo kada definišemo promenljive i dodajemo im vrednosti. Ovo znači da objekti definisani unutar modula imaju pristup ovom imenskom prostoru ako nisu deo obuhvatnog opsega (`enclosing scope`):

Primer

```
x = 1
def f():
    print(x)
f()
```

U ovom primeru promenljiva x nije definisana u okviru funkcije `f()` pa nam se može činiti da bi prevodilac prikazao `NameError` grešku. Međutim, zbog preklapajućih opsega, promenljiva x je tražena u višim nivoima opsega – u ovom slučaju je to opseg na nivou modula i vrednost je vraćena u funkciju.

Iako je ovo sintaksički sasvim moguće i nema greške, ovaj pristup je loša praksa i treba ga izbegavati, jer promenljiva istog imena može postojati u višim opsezima i može doći do problema višeznačnosti. Zato je savet da se za ulazne vrednosti funkcije koriste samo argumenti funkcije.

Obuhvatni (enclosing) i lokalni opseg

Lokalni opseg je imenski prostor „trenutnog" nivoa programa. Dakle, reč je o imenskom prostoru trenutne funkcije ili klase ili uvezenog modula koji nije glavni modul programa.

Primer

```
x = 1
def f():
    x = 1
```

U prvoj liniji koda promenljiva x je definisana na nivou glavnog modula programa, dok je u trećoj liniji koda ta promenljiva definisana kao deo lokalnog imenskog prostora funkcije `f()`.

Treba imati u vidu da najniži nivo zapravo nije lokalni imenski prostor, već nivo na kojem se kod pokreće.

Primer

Radno okruženje

```
x = "0"  # x is defined as part of the namespace of the main module

def example():

    x = 1 # x is defined as part of the local namespace of the function
example()

    def method():

        x = 2 # x is defined as part of the local namespace of the function
method()

        def function():

            x = 3 # x is defined as part of the local namespace function()

            print("Scope of function(): ", x)
```

```
        function()

        print("Scope of method(): ", x)

    method()

    print("Scope of example(): ", x)

example()

print("Scope of main module: ", x)
```

Objašnjenje:

U primeru je prikazano više vrsta opsega promenljive x kao I prikaz ispisa.

Imenski prostori između lokalnog opsega i opsega glavnog modula se nazivaju obuhvatni opsezi. U prethodnom kodu, `method()` funkcija sadrži promenljivu x definisanu kao 2 u svom, lokalnom, opsegu. Ista ta promenljiva u funkciji koja je obuhvata – `example()`, ima vrednost 1 i na kraju, ta promenljiva ima vrednost 0 na nivou glavnog modula. Opseg funkcije `method()` nema pristup promenljivoj x postavljenoj na vrednost 3 u opsegu funkcije `function()`.

Pitanje

Označite imenski prostor koji ne postoji:

- built-in namespace
- global namespace
- **module namespace**
- local namespace

Objašnjenje:

Imenski prostor module ne postoji. Sam modul koji se piše može imati i globalni i lokalni imenski prostor.

Naredbe za promenu opsega

U nekim situacijama može biti jako bitno da pomerimo opseg promenljive izvan njenog lokalnog imenskog prostora. U prošlom primeru nismo mogli da pristupimo promenljivoj x u funkciji function() jer je ona bila u najnižem opsegu. Pravilan način za prosleđivanje promenljivih kroz različite nivoe opsega je njihovo prosleđivanje funkcijama u svojstvu njihovih argumenata. Ponekad ovo nije moguće, pa zato u Pythonu imamo dve naredbe: `global` i `nonlocal`.

Python global naredba

U bilo kom imenskom polju, ako je promenljiva deklarisana sa naredbom `global`, prevodilac će joj pristupiti iz opsega glavnog modula. Iskoristićemo prethodni primer:

Primer global naredbe

Radno okruženje

```
x = "0"  # x is defined as part of the namespace of the main module

def example():

    x = 1 # x je definisan kao deo lokalnog imenskog prostora function
example()

    def method():

        global x # x will be defined as part of the namespace of the main
module
        x = 2 # x is defined as part of the local namespace function
method()

        def function():

            x = 3 # x is defined as part of the local namespace f-is
function ()
            print("Scope of function(): ",x)

        function()

        print("Scope of method(): ",x)

    method()

    print("Scope of example(): ",x)

example()
```

U okviru imenskog prostora funkcije `method()`, deklarisali smo promenljivu x kao globalnu promenljivu ili promenljivu koja je u okviru imenskog prostora glavnog modula. Nakon te naredbe, promene napravljene nad promenljlivom x su uticale i na tu promenljivu u opsegu glavnog modula (prva linija koda), pa se tako na poslednjoj `print()` naredbi vidi da promenljiva x ponovo ima vrednost 2.

Python nonlocal naredba

Kako naredba global služi pri deklarisanju promenljive u okviru opsega glavnog modula, naredba `nonlocal` nam omogućava da deklarišemo promenljivu koja će važiti u imenskom prostoru sledećeg obuhvatnog modula. Na primeru je ovo jasnije:

Primer nonlocal naredbe

Radno okruženje

```
x = "0"  # x is defined as part of the namespace of the main module

def example():

    x = 1 # x is defined as part of the local namespace function example()

    def method():

        nonlocal x # x will be defined as part of the namespace of the
example () function

        x = 2 #x is defined as part of the local namespace function method
        def function():

            x = 3 # x is defined as part of the local namespace function
function ()

            print("Scope of function(): ",x)

        function()

        print("Scope of method(): ", x)

    method()

    print("Scope of example(): ", x)

example()

print("Scope of main module: ", x)
```

Možemo videti da nam je `nonlocal` naredba omogućila deklarisanje promenljive x u okviru imenskog prostora funkcije `method()`.

Bitno je napomenuti da će `nonlocal` naredba vratiti grešku ako je naredni obuhvatni opseg zapravo opseg glavnog modula.

Rezime

- Funkcijom dir() listamo imena dostupna u trenutnom Python okruženju.
- Podržani prostori imena u Pythonu su: ugrađeni, globalni i lokalni.
- Prostor imena je zapravo tipa rečnik, u kojem Python pretražuje potrebno ime promenljive i pomoću kojeg dobavlja objekat na koji pokazuje.
- Podržani opsezi imena u Pythonu su: opseg na nivou modula, obuhvatni opseg i lokalni opseg.
- Python prevodilac referencirano ime traži od najnižeg opsega ka najvišem.
- NameError greška nastaje ako referencirano ime nije pronađeno ni u jednom objektu.
- Opseg promenljive možemo menjati pomoću ključnih reči global i nonlocal.
- Naredbom global ime promenljive postavljamo u globalni imenski prostor.
- Naredba nonlocal nam omogućava da u nižim imenskim prostorima koristimo promenljive već definisane u višim.