Lambda izrazi

Lambda operator ili funkcija je način kreiranja malih, takozvanih anonimnih funkcija, koje nose takav naziv jer nemaju ime. Ove funkcije se pišu samo u jednoj liniji i prestaju da postoje u programu nakon izvršenja, osim ako se dodele promenljivoj pa se na taj način mogu pozivati ponovo. Kreiraju se pomoću ključne reči lambda.

Ime lambda potiče iz lambda računa, kompletno zasnovanog na anonimnim funkcijama, koji je 1936. godine osmislio američki matematičar Alonzo Čerč (Alonzo Church). Kad je reč o programskim jezicima, lambda funkcija je prvi put popularnost doživela u programskom jeziku Lisp. Korisnost ovih funkcija u Pythonu se najviše vidi u njihovom korišćenju zajedno sa funkcijama filter(), map(), kao i funkcijama za sortiranje.

Sintaksa lambda funkcije

Sintaksa ove funkcije izgleda ovako:

Primer

```
lambda list_of_arguments: expression
```

Lista argumenata se sastoji od imena promenljivih odvojenih zarezom, dok je izraz (expression) zapravo bilo koji aritmetički izraz koji koristi argumente iz prethodne liste. Može postojati samo jedan izraz. Ovu funkciju takođe možemo dodeliti promenljivoj kako bismo joj „dali ime".

Primer

```
f = lambda x, y: x + y
a=f(1,2)
print(a)
```

Ovde smo definisali lambda funkciju koja sabira vrednosti koje se nalaze u listi argumenata, a nakon karaktera dve tačke (:) nastavlja se izraz koji će ova funkcija izvršiti. Na kraju, ova funkcija se smešta u promenljivu f. Takođe, poređenja radi, isti rezultat bismo dobili koristeći standardno definisanu funkciju:

Primer

```
def addition(x,y):
    return x + y
```

Lambda i map() funkcije

Kao što smo rekli, ove anonimne funkcije najveći potencijal dostižu kada se koriste uz nekoliko već definisanih funkcija. Jedna od njih je i map() funkcija, koja primenjuje prosleđenu funkciju na svaki element date sekvence.

```
m = map(func, seq)
```

Prvi argument `map()` funkcije je funkcija, a drugi sekvenca. U prethodnim verzijama Pythona ova funkcija je vraćala listu gde je svaki element bio rezultat funkcije `func` nad njegovim parom u prosleđenoj sekvenci. U Python verziji 3, `map()` funkcija vraća iterator.

Primer 1 Lambda i map()	
Kod	**Rezultat**

`def addition(x):`
`  return x + x`
 	
`numbers= [1,2,3,4,5]`
`print(list(map(addition, numbers)))`
`print(list(map(lambda x:x+x, numbers)))`
	`[2, 4, 6, 8, 10]`
	`[2, 4, 6, 8, 10]`

Tabela 16.1. Lambda funkcija i map() funkcija

Ovde smo implementirali dva različita načina kako bismo videli razliku u pristupu korišćenja lambda funkcije i definisanja zasebne funkcije. Kao što se vidi, rezultat obe funkcije je isti. Iako `map()` funkcija vraća iterator, taj objekat smo prebacili u tip list radi lakše manipulacije.

Funkciji `map()` je moguće proslediti i više od jedne sekvence, ali je poželjno da sve budu iste dužine, jer ako nisu, `map()` funkcija će prestati sa izvršavanjem kada dođe do poslednjeg indeksa najkraće liste, pa će tako i njen rezultat biti iste veličine kao i ta lista:

Primer 2 Lambda i map()	
Kod	**Rezultat**

`x = [1,2,3,4]`
`y = [2,4,6,8]`
`z = [-1,-2,-3]`
`print(list(map(lambda a,b,c:a+b+c, x,y,z)))`
	`[2, 4, 6]`

Tabela 16.2. Lambda funkcija i map() funkcija

Nakon posmatranja ispisa, uviđamo logiku map() funkcije; parametar a preuzima vrednosti elemenata liste x, parametar b liste y, a parametar c liste z, koja je ujedno i najkraća, pa čini da rezultat bude iste dužine.

Pitanje

Naredbom `type(lambda x, y: x + y)` dobićemo objekat tipa:

- <class>
- <class lambda>
- **<function>**

Objašnjenje:

Iako se ne koristi ključna reč def, lambda je istog tipa kao i sve ostale funkcije. Ono što je razlikuje jeste to što anonimne funkcije sadrže samo jednu naredbu. Dakle, omogućava nam da ispišemo određenu funkcionalnost bez definisanja zasebne funkcije.

Funkcija filter()

Sintaksa funkcije `filter()` je:

```
filter(func, seq)
```

Ova funkcija nam omogućava filtriranje elemenata sekvence na osnovu `True/False` rezultata prosleđene funkcije. Dakle, svaki element prosleđene sekvence će proći kroz logiku prosleđene funkcije koja će na osnovu zadatog uslova vratiti `bool` vrednost. Povratna vrednost funkcije `filter()` je takođe iterator. U sledećem primeru opet vidimo razliku između implementacije sa lambda funkcijom i implementacije uz definiciju zasebne funkcije:

Primer funkcije filter():

Radno okruženje

```
def filter_func(x):
   if x % 2:
      return True

   else:
      return False

numbers= list(range(0,10))

print(list(filter(filter_func, numbers)))

print(list(filter(lambda x: x % 2, numbers)))
```

Vežba

Napisati lambda funkciju koja prima argument x i vraća njegov kvadrat i rezultat ispisati na ekranu.

Rešenje

```
x = 10

result = lambda n: n**2
print(result(x))
```

Objašnjenje:

Lambda funkciji smo prosledili parametar x, čiji smo kvadrat izračunali pomoću (**) operatora.

Sortiranje

Iako liste imaju sopstvenu metodu za sortiranje u vidu `list.sort()`, bavićemo se funkcijom `sorted()`, koja se češće koristi, a po argumentima je gotovo identična `list.sort()` pozivu, i koja je primenjiva i na ostale sekvence. Sintaksa `sorted()` funkcije je:

```
sorted(seq, [key=None, reverse=False])
```

Povratna vrednost je lista bez obzira na to kog tipa je prosleđena sekvenca, dok su parametri `key i reverse` opcioni. Parametar `reverse` prihvata `bool` vrednost (`True/False`) i određuje hoće li smer sortiranja ići po obrnutom redosledu, dok nam parametar `key` omogućava da lambda funkcija dođe do izražaja. Naime, u `key` parametar se prosleđuje funkcija po čijoj logici će se vršiti sortiranje. Kod funkcije koja se prosleđuje je bitno da svaki element sekvence može pretvoriti u numeričku vrednost.

Primer

```
s = "String"
print(sorted(s, key=lambda x: x))
```

U ovom slučaju je lambda funkcija svaki element prosleđene sekvence pretvorila u numeričku vrednost; konkretno, pretvorila je svaki karakter u njegov ASCII kod i po tome obavila sortiranje.

Kod sortiranja je bitno i to da svi elementi prosleđene sekvence budu istog tipa. Postoje dva ograničenja pri korišćenju parametra `key`:

- broj argumenata funkcije koja se prosleđuje mora biti jedan;
- funkcija koja se koristi kod ovog parametra mora biti tako kodirana da može obraditi svaki element; na primer, ako je sekvenca koju sortiramo tipa `string`, funkcija će morati svaki karakter da prebaci u tip int; u suprotnom, sortiranje neće uspeti.

Uslovi u Lambda funkciji

Lambda funkcija u svojoj sintaksi dozvoljava i korišćenje `if-else` konstrukcije, i to na sledeći način:

Primer

```
lambda x: True if uslov else False
```

Dakle, ako je uslov tačan, u program će se vratiti vrednost na mestu `True`, a u suprotnom vratiće se vrednost na mestu False.

Primer

```
lambda x: 'Positive number' if x > 0 else 'Negative number'
```

Ovaj primer će pri pozivu, u zavisnosti od vrednosti argumenta x, vratiti u program string Positive number ili Negative number.
Takođe je moguće korišćenje više uslova upotrebom `and` ključne reči, kao i korišćenje ugnežđenih `if-else` konstrukcija. Jedan takav primer bi izgledao ovako:

```
lambda x : x^2 if x < 5 else (x^3 if x > 5 else x)
```

Naravno, ovo svojstvo lambda funkcije je primenljivo i kada se koristi kao pomoćna funkcija u radu sa `map(), filter(), sorted()` i sličnim.

Vežba

Koristeći lambda funkciju, datu listu čiji su uređeni parovi n-torke ('city_name', postal_code) sortirati po poštanskom broju; rezultat smestiti u listu i ispisati na ekranu.

```
cities =[("New York",10001),
       ("Paris",75000),
       ("Moscow", 101000),
       ("Tokyo",100000)]
```

Radno okruženje

```
cities =[("New York",10001),

     ("Paris",75000),

     ("Moscow", 101000),

     ("Tokyo",100000)]

cities = sorted(cities, key=lambda x: x[1])
print(cities)
```

Objašnjenje:

Za ovaj problem nam je najlakše koristiti sorted() funkciju. Prosleđujemo joj sekvencu koju želimo da sortiramo i ključ po kojem je sortiramo. Ključ je u ovom slučaju lambda funkcija sa parametrom x, koji se odnosi na svaki element date liste gradovi. Kako je svaki element ove liste n-torka sa dva elementa, potrebno je indeksirati drugi element n-torke (pozivni broj) kako bismo listu sortirali upravo po pozivnom broju.

Rezime

- Lambda funkcija u Pythonu je poseban tip funkcije poznat kao anonimna funkcija.
- Lambda funkcija se definiše koristeći lambda ključnu reč propraćenu argumentima odvojenim zarezom (,), karakterom dve tačke (:) i jednom naredbom.
- Lambda funkcija podržava if-else naredbu.
- Filter funkcija nam omogućava filtriranje elemenata date sekvence koristeći pomoćnu funkciju kao ključ.
- Map funkcija nam omogućava da na svaki element prosleđene sekvence primenimo željenu funkciju.
- Sorted funkcija nam omogućava sortiranje date sekvence na osnovu unapred zadate funkcije. Povratna vrednost ove funkcije je lista. Prosleđena sekvenca mora imati elemente istog tipa.