Naučite Python 3

Brzi kurs programiranja

Ashwin Pajankar

Agencija Eho
www.infoelektronika.net

ISBN 978-86-80134-39-0

Naučite Python 3

Naslov originala: Kickstart to Python 3
Autor: Ashwin Pajankar

Prevod: Biljana Tešić

Izdaje i štampa: Agencija Eho, Niš
e-mail: redakcija@infoelektronika.net

Tiraž: 200

Godina izdanja: 2022

CIP - Каталогизација у публикацији
Библиотеке Матице српске, Нови Сад

004.43PYTHON

АШВИН, Пајанкар
Naučite Python 3 : brzi kurs programiranja / Ashwin Pajankar ; prevod Biljana Tešić. - Niš : Agencija Eho, 2022 (Niš : Agencija Eho). - 206 str. : ilustr. ; 24 cm

Prevod dela: Kickstart to Python 3. - Tiraž 200. - Bibliografija.

ISBN 978-86-80134-39-0

a) Програмски језик "Python"

COBISS.SR-ID 64694793

Poglavlje 1 - Uvod u Python

Nadam se da ste pogledali sadržaj knjige. Ako niste, onda vas molim da ga pogledate, jer će vam dati predstavu o sadržaju ovog poglavlja. Ako niste potpuni početnik u Python-u, možda će vam ovo poglavlje biti osnovo. Međutim, ako ste novi u Python-u ili računarskom programiranju, ovo poglavlje će vam biti veoma korisno.

Počećemo naše putovanje pomoću malih i jednostavnih koraka u ovom poglavlju. U ovom poglavlju ćemo naučiti sledeće teme:

- Istorija programskog jezika Python
- Instaliranje Python-a na različitim platformama
- IDLE
- Režim script za Python
- Python IDE-ovi
- Python implementacije i distribucije
- Python Package Index

Kada pročitate ovo poglavlje u celosti, naučićete osnove Python-a i pokretanje jednostavnih Python programa.

1.1 Istorija programskog jezika Python

Python je interpretirani programski jezik visokog nivoa i opšte namene. Napravljen je sa namerom da kod bude lako čitljiv. Python kod se može okarakterisati sintaksom nalik engleskom jeziku. Lak je za čitanje i razumevanje osobi koja je tek počela da uči programiranje. Mnoge funkcije su „pozajmljene“ iz drugih programskih jezika.

Programski jezik Python je pod velikim uticajem ABC-a koji je razvijen u kompaniji **„Centrum Wiskunde & Informatica (CWI)“**. Sam ABC je bio pod velikim uticajem SETL-a i ALGOL-a 68.

Glavni autor Pythona je Guido Von Rossum. Radio je sa programskim jezikom ABC u kompaniji „CWI“. Python zajednica mu je dodelila titulu **Benevolent Dictator for Life (BDFL)**.

Ideja o programskom jeziku Python začeta je kasnih osamdesetih godina prošlog veka kao o nasledniku ABC programskog jezika. Guido je takođe pozajmio sistem modula od Modula-3. Počeo je da primenjuje jezik 1989. godine. Verzija 0.9.0 je objavljena na alt.sources-u u februaru 1991. godine. Verzija 1.0 je objavljena 1994. godine. Python 2.0 je objavljen u oktobru 2000. godine.

U decembru 2008. godine objavljena je nova (i unazad nekompatibilna) verzija programskog jezika Python, poznata kao Python 3. Podrška za Python 2 je prestala 1. januara 2020. godine i više se ne razvija. Sada je jedina verzija Python 3. Ona je u aktivnom razvoju i podržao ju je **Python Software Foundation**. Za sve demonstracije u knjizi koristimo Python 3. Kad god pomenem Python, mislim Python 3.

1.2 Instaliranje Python-a na različitim platformama

U ovom odeljku ćemo naučiti kako da instaliramo Python 3 na Windows-u i Linux-u. Python podržava veliki broj operativnih sistema. Međutim, najčešće korišćeni operativni sistemi za razvoj pomoću Python-a su Windows i Linux. Stoga ću objasniti Python 3 pomoću ove dve platforme.

1.2.1 Instalacija na Linux-u

Skoro sve glavne distribucije Linux-a se isporučuju sa instaliranim Python-om 2 i 3. Python 2 interpreter je binarni izvršivi **python**, a Python 3 interpreter je binarni izvršivi **python3**. Koristim Debian Linux varijantu **Raspberry Pi OS** sa Raspberry Pi-em 4 kao mojom omiljenom Linux platformom za Python razvoj. Otvorite emulator terminala vaše Linux distribucije i pokrenite sledeću komandu:

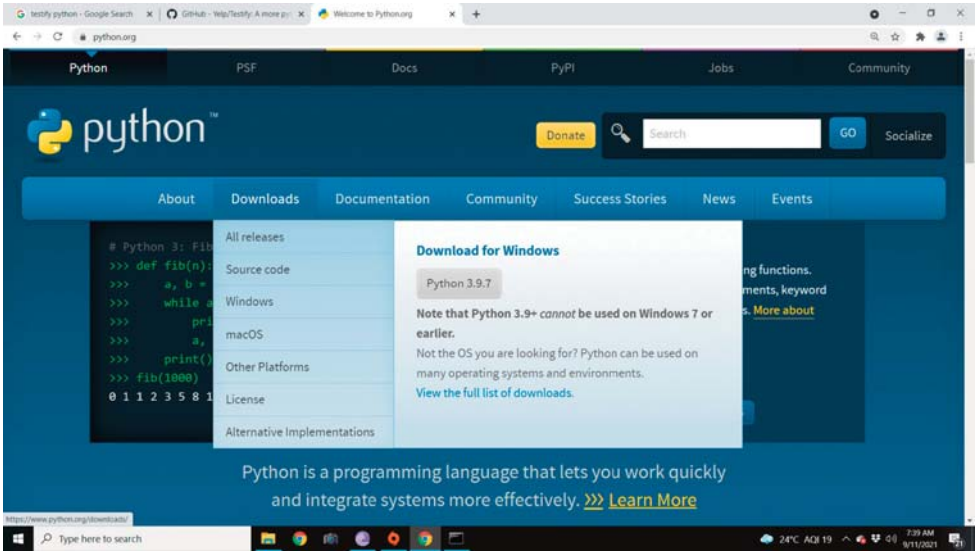
```
pi@raspberrypi:~ $ python3 -V
```

Biće prikazan sledeći rezultat,

Python 3.7.3

1.2.2 Instalacija na Windows-u

Veoma je lako i jednostavno instalirati Python na Windows računaru. Moramo da pristupimo Python početnoj stranici na internetu koja se nalazi na adresi www.python.org. Ako zadržite pokazivač miša iznad odeljka **Downloads**, veb sajt automatski otkriva operativni sistem i prikazuje odgovarajuću datoteku za preuzimanje. Kliknite na dugme sa natpisom **Python 3.9.7**. Verzija programa u momentu čitanje knjige će se promeniti, ali je proces isti. Kada kliknete na dugme, biće preuzeta windows izvršiva instalaciona datoteka u korisničkom direktorijumu **Downloads** na vašem računaru. On otkriva arhitekturu vašeg računara i preuzima relevantnu datoteku. Na primer, imam x86 64-bitni Windows računar. Preuzeo je datoteku **python-3.9.7-amd64.exe**. Snimak ekrana veb sajta www.pithon.org je prikazan na slici 1-1 na sledeći način:



Slika 1-1: Preuzimanje Python instalacije za Windows

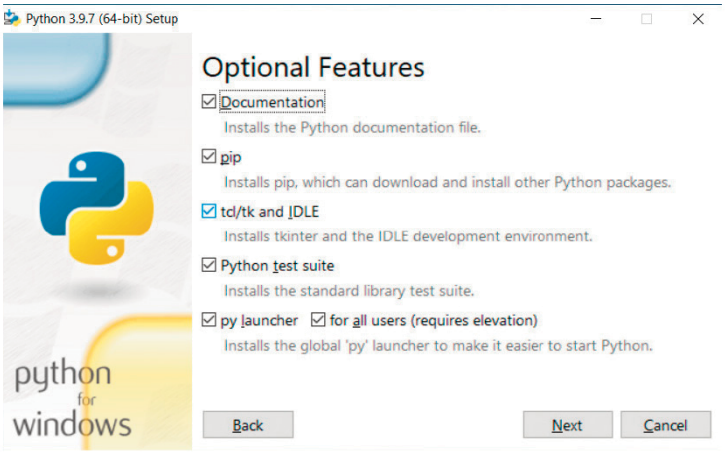
Sada otvorite instalacionu datoteku i ona će pokrenuti instalacioni program.

NAPOMENA: Datoteku možete otvoriti klikom na preuzetu datoteku u vašem pregledaču. Takođe možete pronaći fizičku lokaciju datoteke, tako što ćete pregledati opcije preuzimanja u vašem pretraživaču. Ukazaće na lokaciju u direktorijumu **Downloads** vašeg Windows korisnika.

Prozor instalacionog programa je prikazan na snimku ekrana na slici 1-2:

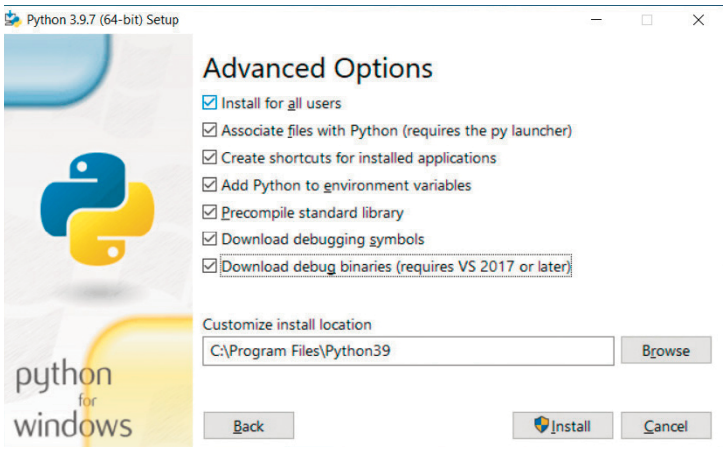
Slika 1-2: Python instalacioni program

Označite sva polja za potvrdu i kliknite na opciju **Customize installation**. Otvoriće se sledeći ekran za instalaciju kao na slici 1-3:



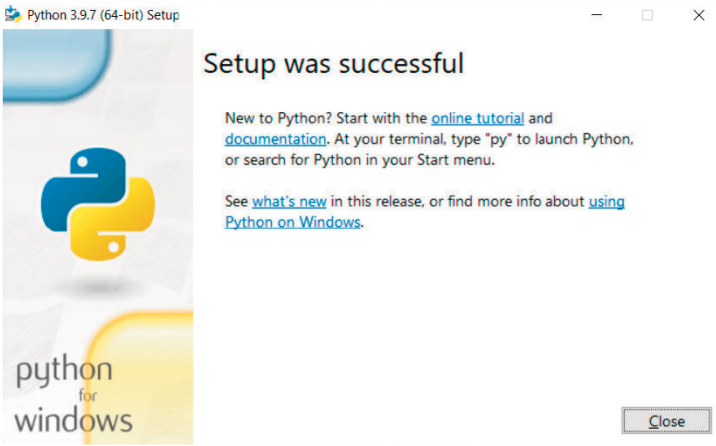
Slika 1-3 - Opcije za instalaciju

Kliknite na dugme **Next** i odvešće vas do ekrana prikazanog na slici 1-4:



Slika 1-4 - Opcije za instalaciju

Proverite sve opcije i kliknite na dugme **Install**. Zatim će od vas biti zatražene administratorski akreditivi. Unesite akreditive i biće započeto instaliranje Python-a i drugih relevantnih programa na vašem Windows računaru. Kada se instalacija završi, biće prikazan sledeći ekran (slika 1-5):



Slika 1-5 - Poruka o uspehu instalacije

Pre nego što počnemo da slavimo, moramo da proverimo nekoliko „stvari". Otvorite komandnu liniju prozora (cmd) i pokrenite sledeću komandu:

```
C:\Users\Ashwin>python -V
```

Rezultat je sledeći,

Python 3.9.7

Čestitam! Instalirali ste Python 3 na vaš Windows računar.

1.3 IDLE

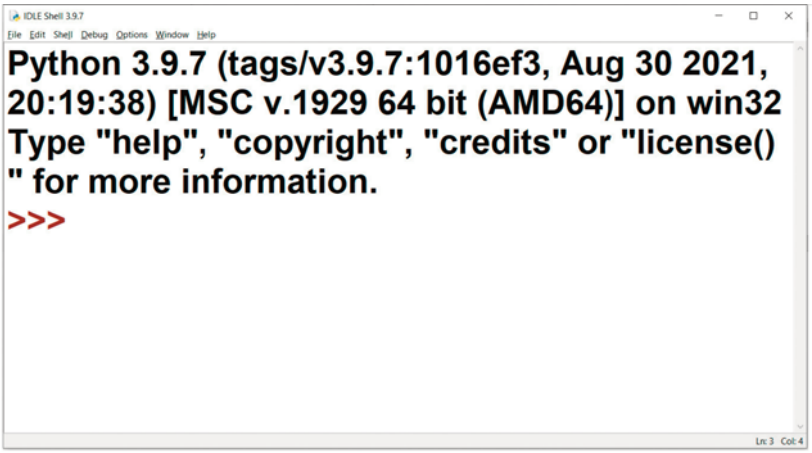
Python Softvare Foundation je razvio integrisano razvojno okruženje (IDE - integrated development environment) za Python. Nazvan je „IDLE", što je skraćenica za **Integrated Development and Learning Environment**. Isporučuje se sa Python instalacijom kada ga instaliramo na Windowsu. Na Linux distribucijama, moramo ga zasebno instalirati. Za Debian i derivate, pokrenite sledeću komandu na komandnoj liniji (emulatoru terminala):

```
pi@raspberrypi:~ $ sudo apt-get install idle -y
```

Biće instaliran IDLE na vašoj Linux distribuciji.

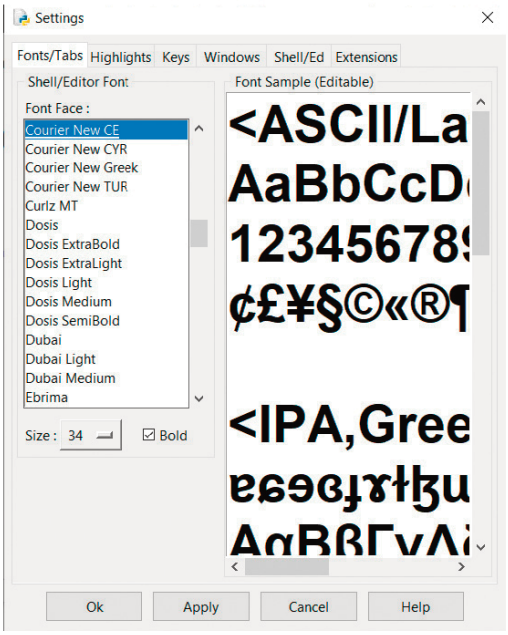
Sada ćemo da koristimo IDLE. Možemo ga pronaći, tako što ćemo upisati IDLE u Windows traku za pretragu. Možemo ga pronaći u Linux meniju. U Linux-u, možemo ga pokrenuti iz komandne linije pomoću sledeće komande,

```
pi@raspberrypi:~ $ idle &
```



Slika 1-6 - IDLE

U traci menija, ispod stavke menija Options, nalazimo opciju **Configure IDLE** u kojoj možemo podesiti veličinu fonta i druge detalje (slika 1-7),



Slika 1-7 - Opcije konfiguracije IDLE-a

Promenite font i veličinu po svom izboru i kliknite na dugme **Apply**, a zatim na dugme **OK**. Sada ćemo pokušati da razumemo interaktivni režim Python-a. Kada se pozove, Windows IDLE prikazuje interaktivni režim. Možemo ga koristiti za direktno pokretanje Python iskaza, a da pri tom ne moramo da ih sačuvamo kao datoteku.

Python iskazi se šalju direktno u interpretator i rezultat se odmah prikazuje u istom prozoru. Ako ste koristili komandnu liniju OS-a, onda znate da je ovo skoro isto. Interaktivni režim se obično koristi za pokretanje pojedinačnih iskaza ili kratkih blokova koda. Sada ćemo pokušati da pokrenemo jednostavan iskaz:

```
>>> print(“Hello World!”)
```

Ovo daje sledeći rezultat i ispisuje ga u istom prozoru:

```
Hello World!
```

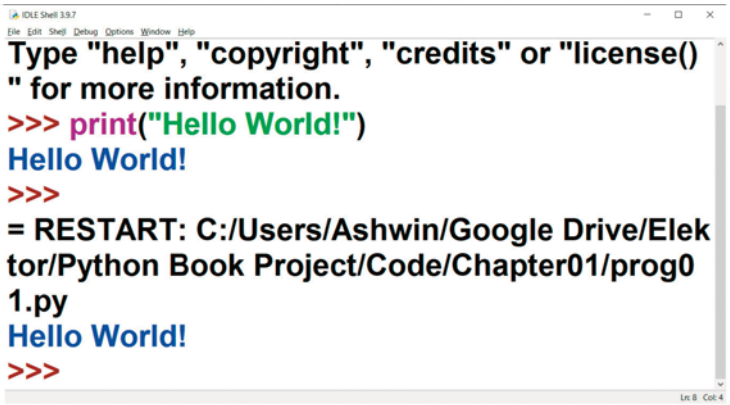
Pokazivač se vraća na komandnu liniju i spreman je da preuzme novu komandu od korisnika. Na ovaj način smo pokrenuli naš prvi Python iskaz. Možemo izaći pokretanjem komande **exit()** u interpretatoru. Alternativno, možemo pritisnuti CTRL + D da izađemo. Možemo takođe pozvati interaktivni režim iz komandne linije, tako što ćemo pokrenuti komande **python** na Windows-u i **python3** na Linux komandnoj liniji.

1.4 Režim script za Python

Python interaktivni režim je dobar za jednolinijske iskaze i male blokove koda. Međutim, ne čuva iskaze kao programe. Ovo se može uraditi u režimu script. U Python interpretatoru IDLE-a ispod menija **File** na traci menija izaberite **New file**. Unesite sledeći kod u novu datoteku:

```
print(“Hello World!”)
```

U meniju **File** na traci menija izaberite **Save**. Otvoriće se prozor **Save as**. Sačuvajte datoteku na lokaciji po vašem izboru. IDLE automatski dodaje ekstenziju ***.py** na kraj naziva datoteke. Zatim kliknite na meni **Run** na traci menija, a zatim kliknite na **Run Module**. Ovo će izvršiti program i prikazati rezultat u prozoru interaktivnog režima IDLE-a kao što je prikazano ispod (slika 1-8).



Slika 1-8 - Rezultat Python skripta

Drugi način da izvršite program je da ga pokrenete iz komandne linije. Da biste pokrenuli program u komandnoj liniji, idite do direktorijuma u kojem je program sačuvan, a zatim pokrenite sledeću komandu u Windowsu:

```
C:\ Python Book Project\Code\Chapter01>python prog01.py
```

Ovo daje sledeći rezultat:

Hello World!

U Linuxu se koristi sledeća komanda:

```
pi@raspberrypi:~ $ python3 prog01.py
```

Možete koristiti i sledeću komandu u Linuxu da biste pokrenuli program:

```
pi@raspberrypi:~ $ which python3
```

Ovo daje lokaciju Python 3 interpretera kao rezultat:

/usr/bin/python3

Sada dodajemo sledeću liniju koda u Python datoteku koju smo kreirali:

```
#!/usr/bin/python3
```

Stoga će cela datoteka koda izgledati ovako:

```
#!/usr/bin/python3
print(“Hello World!”)
```

Sada ćemo da promenimo dozvole za Python programske datoteke. Ako smo sačuvali datoteku sa nazivom **prog01.py**, pokrenimo sledeću komandu:

```
pi@raspberrypi:~ $ chmod +x prog01.py
```

Na ovaj način smo promenili dozvole datoteke i učinili je izvršivom. Možemo pokrenuti datoteku na sledeći način:

```
pi@raspberrypi:~ $ ./prog01.py
```

Imajte na umu da smo nazivu datoteke dodali prefiks **./**. To proizvodi sledeći rezultat:

Hello World!

Dodali smo prvi red tako da shell operativnog sistema zna koji interpreter da koristi za pokretanje programa.

Evo nekoliko načina na koje možemo da pokrenemo Python programe.

1.5 Python IDE-ovi

Do sada smo koristili IDLE za Python programiranje. Takođe možemo koristiti druge uređivače i IDE-ove. U Linux komandnoj liniji možemo koristiti programe za uređivanje kao što su **vi**, **vim** i **nano**. **Vi** uređivač se isporučuje sa većinom Linux distribucija. Ostala dva možemo da instaliramo na Debianu (i izvedenim distribucijama), pomoću sledeće komande:

```
pi@raspberrypi:~ $ sudo apt-get install vim nano -y
```

Takođe možemo da koristimo uređivač otvorenog teksta kao što je **Notepad** na Windows-u ili **Leafpad** na Linux-u. Možemo da instaliramo Leafpad uređivač na Debianu i drugim distribucijama, pomoću sledeće komande:

```
pi@raspberrypi:~ $ sudo apt-get install leafpad -y
```

Raspberry Pi OS (moj omiljeni Debian derivat) se isporučuje sa IDE-ovima **Thonny**, **Geany** i **Mu**. Možemo da ih instaliramo na druge Debian derivate pomoću sledeće komande:

```
pi@raspberrypi:~ $ sudo apt-get install thonny geany mu-editor -y
```

Ako vam je jednostavniji Eclipse, postoji lep plugin poznat kao **Pydev**. On se može instalirati sa **Eclipse Marketplace-a**.

1.6 Python implementacije i distribucije

Program koji interpretira i pokreće Python program poznat je kao Python interpreter. Onaj koji podrazumevano dolazi sa Linuxom i koji obezbeđuje Python Softvare Foundation je poznat kao CPython. Druge organizacije su kreirale Python interpretere koji se pridržavaju Python standarda. Ovi interpreteri su poznati kao Python implementacije. Baš kao što C i C++ imaju mnogo kompajlera, Python ima mnogo implementacija interpretera. U knjizi ćemo koristiti standardni CPython interpreter koji se podrazumevano isporučuje uz Linux. Sledi delimična lista drugih popularnih alternativnih implementacija Python interpretera:

- IronPython
- Jython
- PyPy
- Stackless Python
- MicroPython

Mnoge organizacije povezuju Python interpreter po svom izboru sa mnogim modulima i bibliotekama i distribuiraju ga. Ovi paketi su poznati kao Python distribucije. Listu Python implementacija i distribucija možemo videti na sledećim URL-ovima:

https://www.python.org/download/alternatives/
https://wiki.python.org/moin/PythonDistributions
https://wiki.python.org/moin/PythonImplementations

1.7 Python Package Index

Python se isporučuje sa mnogo biblioteka. Ovo je poznato kao filozofija Python-a „uključene baterije". Mnogo više biblioteka razvijaju mnogi nezavisni programeri i organizacije. Na osnovu vašeg radnog profila, ove biblioteke mogu biti korisne u izvršavanju predviđenih zadataka. Sve takve biblioteke trećih strana su smeštene u **Python Package Index-u**. On se nalazi na stranici https://pypi.org/. Biblioteku možemo potražiti na toj stranici.

Python se isporučuje sa uslužnim programom poznatim kao **pip**. Pip je **obrnuti akronim**. To znači da proširenje pojma sadrži sam pojam. Pip znači **pip installs packages** ili **pip installs python**. To je alatka za upravljanje paketima koja instalira Python pakete, koji dolazi kao pomoćni program komandne linije. Njenu verziju možemo proveriti tako što ćemo pokrenuti sledeću komandu na komandnoj liniji (cmd-u i emulatoru terminala) operativnog sistema:

```
pi@raspberrypi:~ $ pip3 -V
```

Biće ispisana trenutno instalirana verziju pip-a. Ako želimo da vidimo listu trenutno instaliranih paketa, potrebno je da pokrenemo sledeću komandu:

```
pi@raspberrypi:~ $ pip3 list
```

Biće vraćena lista svih paketa koji su već instalirani.

NAPOMENA: Sve komande koje se odnose na pip su iste i na Windowsu i na Linuxu. Ako želimo da instaliramo novu biblioteku, možemo da je potražimo u PyPI-u. Možemo da instaliramo novi paket na sledeći način:

```
pi@raspberrypi:~ $ pip3 install numpy
```

Biće instalirana NumPy biblioteka na računaru. Koristićemo ovaj uslužni program u celoj knjizi za instaliranje potrebnih paketa trećih strana.

Rezime

U ovom poglavlju smo razmotrili istoriju programskog jezika Python i njegovu instalaciju na Windowsu i Linuxu. Takođe smo videli kako se koristi Python interpreter i kako se pišu i izvršavaju Python skriptovi na različite načine. Pregledali smo IDLE i takođe smo videli razne druge IDE-ove za Python. Na kraju, naučili smo kako da koristimo pip, tj. menadžer paketa za Python.

Sledeće poglavlje će biti mnogo praktičnije. Naučićemo kako da pišemo programe pomoću ugrađenih struktura podataka.

Poglavlje 2 - Ugrađene strukture podataka

U prethodnom poglavlju smo instalirali Python 3 na različite platforme. Napisali smo jednostavan uvodni program i naučili kako da ga pokrenemo na razne načine. Takođe smo naučili kako da koristimo interpretirani (interaktivni) režim. To poglavlje je bilo uvodno i nije bilo previše teško kada je reč o programiranju.

Ovo poglavlje je malo više fokusirano na programiranje (poznato i kao kodiranje). Upoznaćemo se sa različitim ugrađenim strukturama podataka u Python-u. Detaljno ćemo se fokusirati na sledeće teme:

- IPython
- liste
- n-torke
- skupovi
- rečnik

Nakon što pročitate ovo poglavlje u celosti, moći ćete da koristite IPython i ugrađene strukture podataka u Python-u.

2.1 IPython

IPython znači **Interactive Python Shell**. To je program koji nam obezbeđuje više mogućnosti od Python-ove ugrađenog interaktivnog shella. Moramo da ga instaliramo zasebno pomoću sledeće komande:

```
pi@raspberrypi:~ $ pip3 install ipython
```

Komanda je ista i za Windows i za macOS. Ako ga instalirate na Linux distribuciji (kao ja), možda ćete videti sledeću poruku u evidenciji instalacije:

> **Skriptovi iptest, iptest3, ipython i ipython3 su instalirane u direktorijumu '/home/pi/.local/bin' koji nije u PATH-u.**
>
> **Razmislite o dodavanju ovog direktorijuma u PATH-u, a ako želite da potisnete ovo upozorenje, upotrebite —no-warn-script-location.**

To znači da moramo da dodamo pomenutu lokaciju direktorijuma u datoteke **~/.bashrc** i **~/.bash_profile**. Moramo da dodamo sledeću liniju u obe datoteke (tako da radi za shell za prijavljivanje i za odjavljivanje):

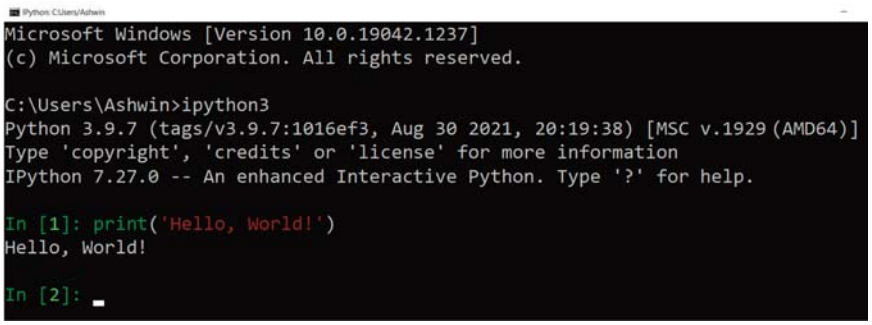
```
PATH=$PATH:/home/pi/.local/bin
```

Takođe će biti prikazana slična poruka za Windows. Moramo da dodamo putanju direktorijuma pomenutog u instalacionoj evidenciji **PATH** promenljivima (User i System) u Windows-u.

Kada promenimo putanju, moramo da zatvorimo i ponovo pozovemo uslužne programe komandne linije operativnih sistema. Nakon ovoga, pokrenite sledeću komandu:

```
pi@raspberrypi:~$ ipython3
```

Biće pokreunt IPython za Python 3 u komandnoj liniji. Komanda je ista i za Windows i za druge platforme. Na slici 1-2 je snimak ekrana IPython sesije u toku na Windows desktop računaru:



Slika 2-1 - IPython sesija u toku

Ovakav IPython možemo koristiti za pisanje malih programa. Dakle, krenimo.

2.2 Liste

Možemo da uskladištimo više od jedne vrednosti u listama. Liste su ugrađena funkcija Python-a. Ne moramo ništa da instaliramo ili uvozimo kada koristimo liste. Elementi liste su zatvoreni u uglastim zagradama i odvojeni zarezima. Ali, liste ni na koji način nisu linearne strukture podataka, jer možemo imati i listu lista. Videćemo više o listi lista kasnije kada budemo naučili osnove.

Liste su promenljive, što znači da ih možemo menjati. Sada ćemo videti nekoliko primera lista i povezanih ugrađenih rutina za liste. Otvorite IPython u komandnoj liniji vašeg operativnog sistema i pratite kod:

```
In [1]: fruits = ['babana', 'pineapple', 'orange']
```

Ovo će kreirati listu. Sada je možemo ispisati na konzoli na dva načina:

```
In [2]: fruits
```

Ovo proizvodi sledeći rezultat:

```
Out[2]: ['babana', 'pineapple', 'orange']
```

Takođe možemo da koristimo ugrađenu funkciju print na sledeći način:

```
In [3]: print(fruits)
```

Ovo je rezultat:

['banana', 'pineapple', 'orange']

Lista je uređena struktura podataka, što znači da se članovi lista skladište i preuzimaju određenim redosledom. Možemo to iskoristiti u našu prednost da bismo preuzeli elemente lista. Prvi element ima indeks 0. Ako je veličina liste **n**, poslednji element se indeksira kao **n-1**. Ovo je slično indeksima niza u jezicima C, C++ i Java. Ako ste ranije programirali na ovim programskim jezicima, ova šema indeksiranja će vam biti poznata.

Možemo da preuzmemo elemente lista korišćenjem kombinacije naziva liste i indeksa elementa. Evo primera:

```
In [4]: fruits[0]
Out[4]: 'banana'
In [5]: fruits[1]
Out[5]: 'pineapple'
In [6]: fruits[2]
Out[6]: 'orange'
```

Možemo koristiti i negativne indekse. Indeks -1 se odnosi na poslednji element, a -2 se odnosi na prethodni element. Evo primera:

```
In [7]: fruits[-1]
Out[7]: 'orange'
In [8]: fruits[-2]
Out[8]: 'pineapple'
```

Ako pokušamo da koristimo nevažeći indeks, videćemo sledeće rezultate:

```
       In [9]: fruits[3]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-9-7ceeafd384d7> in <module>
----> 1 fruits[3]

       IndexError: list index out of range
       In [10]: fruits[-4]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
```

```
<ipython-input-10-1cb2d66442ee> in <module>
----> 1 fruits[-4]

       IndexError: list index out of range
```

Možemo da dobijemo dužinu liste na sledeći način:

```
In [12]: len(fruits)
Out[12]: 3
```

Takođe možemo videti tip podataka liste na sledeći način:

```
In [13]: type(fruits)
Out[13]: list
In [14]: print(type(fruits))
<class 'list'>
```

Kao što vidimo u rezultatu, klasa promenljive je lista. To ćemo detaljno razmotriti u četvrtom poglavlju knjige.

Možemo koristiti konstruktor **list()** da bismo kreirali listu:

```
In [15]: fruits = list(('banana', 'pineapple', 'orange'))
In [16]: fruits
Out[16]: ['banana', 'pineapple', 'orange']
```

Možemo da preuzmemo opseg elemenata iz sledećih lista:

```
In [17]: SBC = ['Raspberry Pi', 'Orange Pi', 'Banana Pi', 'Banana Pro', 'NanoPi',
'Arduin
    ...: o Yun', 'Beaglebone']
In [18]: SBC[2:5]
Out[18]: ['Banana Pi', 'Banana Pro', 'NanoPi']
```

U ovom primeru preuzimamo elemente indeksirane sa 2, 3 i 4. Takođe, razmotrite sledeći primer:

```
In [19]: SBC[2:]
Out[19]: ['Banana Pi', 'Banana Pro', 'NanoPi', 'Arduino Yun', 'Beaglebone']
```

Na ovaj način možemo da preuzmemo elemente od indeksa 2, pa nadalje.

```
In [20]: SBC[:2]
Out[20]: ['Raspberry Pi', 'Orange Pi']
```

I na ovaj način možemo preuzeti sve elemente pre indeksa 2. Takođe možemo da koristimo negativne indekse da bismo preuzeli više elemenata na sledeći način:

```
In [21]: SBC[-4:-1]
Out[21]: ['Banana Pro', 'NanoPi', 'Arduino Yun']
```

Takođe možemo koristiti **if** konstrukciju da bismo proverili da li element postoji na listi na sledeći način:

```
In [23]: if 'Apple Pie' in SBC:
    ...:     print('Found')
    ...: else:
    ...:     print('Not Found')
    ...:
       Not Found
```

Stavku na listi možemo promeniti na sledeći način:

```
In [25]: SBC[0] = 'RPi 4B 8GB'
```

Takođe možemo da ubacimo stavku u listu u indeksu na sledeći način:

```
In [36]: SBC.insert(2, 'Test Board')
```

Stavka na indeksu 2 u ovoj listi je pomerena za jednu poziciju unapred. Isto važi i za ostale stavke.

Možemo dodati stavku u listu na sledeći način,

```
In [38]: SBC.append('Test Board 1')
```

Ovo će dodati stavku na kraj liste. Takođe možemo koristiti proširene operacije u listama. Ovo dodaje jednu listu na kraj druge liste.

```
In [39]: list1 = [1, 2, 3]; list2 = [4, 5, 6];
In [40]: list1.extend(list2)
In [41]: list1
Out[41]: [1, 2, 3, 4, 5, 6]
```

Možemo ukloniti stavku sa liste na sledeći način:

```
In [43]: SBC.remove('Test Board')
```

Možemo koristiti dva različita pristupa da bismo uklonili stavku na određenom indeksu. Oba pristupa su prikazana u nastavku:

```
In [44]: SBC.pop(0)
Out[44]: 'RPi 4B 8GB'
In [46]: del SBC[0]
```

Ako ne navedemo indeks, iskočiće (što znači ukloniti i vratiti) poslednja stavka:

```
In [47]: SBC.pop()
Out[47]: 'Test Board 1'
```

Možemo ukloniti sve elemente sa liste na sledeći način:

```
In [48]: SBC.clear()
```

Takođe možemo da izbrišemo celu listu na sledeći način,

```
In [49]: del SBC
```

Ako sada pokušamo da pristupimo listi, ona će vratiti grešku na sledeći način:

```
In [50]: SBC
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-50-69ed78d7b4fc> in <module>
----> 1 SBC

NameError: name 'SBC' is not defined
```

Sada ćemo naučiti kako da koristimo liste sa petljama. Kreirajte listu na sledeći način:

```
In [51]: fruits = ['apple', 'banana', 'cherry', 'pineapple', 'watermelon',
'papaya']
```

Možemo koristiti konstrukciju petlje for na sledeći način,

```
In [52]: for member in fruits:
    ...:     print(member)
    ...:
apple
banana
cherry
pineapple
watermelon
papaya
```

Sledeći kod takođe daje isti rezultat:

```
In [53]: for i in range(len(fruits)):
    ...:     print(fruits[i])
    ...:
apple
banana
cherry
pineapple
watermelon
papaya
```

Takođe možemo koristiti petlju while na sledeći način:

```
In [54]: i = 0

In [55]: while i < len(fruits):
    ...:     print(fruits[i])
    ...:     i = i + 1
    ...:
apple
banana
cherry
pineapple
watermelon
papaya
```

Pre nego što nastavim dalje, želim da razmotrim jednu važnu funkciju. Koristili smo primere mnogih lista. Većina lista koje smo koristili su liste nizova znakova. Par su liste brojeva. Takođe možemo imati liste drugih tipova podataka. Evo primera:

```
In [56]: l1 = [1.2, 2.3, 3.4]

In [57]: l2 = ['a', 'b', 'c']

In [58]: l3 = [True, False, False, True, True, False]
```

Ovde smo koristili liste brojeva sa pokretnom tačkom, znakova i Bulovih vrednosti, tim redom. Takođe možemo kreirati listu mešovitih tipova na sledeći način:

```
In [59]: l4 = [1, 'Test', 'a', 1.2, True, False]
```

Listu možemo sortirati na sledeći način:

```
In [60]: fruits.sort()
In [61]: fruits
Out[61]: ['apple', 'banana', 'cherry', 'papaya', 'pineapple', 'watermelon']
```

Takođe možemo sortirati listu obrnutim redosledom na sledeći način:

```
In [62]: fruits.sort(reverse = True)
In [63]: fruits
Out[63]: ['watermelon', 'pineapple', 'papaya', 'cherry', 'banana', 'apple']
```

Kao vežbu, sortirajte numeričke i bulove liste.

Možemo da kopiramo jednu listu u drugoj na sledeći način:

```
In [64]: newlist = fruits.copy()
```

Ranije smo videli da rutina **extend()** može spojiti dve liste. Možemo koristiti operator sabiranja (+) da bismo spojili dve liste na sledeći način:

```
In [65]: l1 + l2
Out[65]: [1.2, 2.3, 3.4, 'a', 'b', 'c']
```

Možemo koristiti operator množenja sa listama na sledeći način,

```
In [66]: l1 * 3
Out[66]: [1.2, 2.3, 3.4, 1.2, 2.3, 3.4, 1.2, 2.3, 3.4]
```

2.3 N-torke

N-torke su slične listama. Moramo da koristimo zagrade dok ih kreiramo. Razlikuju se od lista po tome što su nepromenljive, što znači da kada se kreiraju, ne mogu se menjati. Pogledajmo jednostavan primer na sledeći način,

```
In [1]: fruits = ('apple', 'grape', 'mango')
In [2]: fruits
Out[2]: ('apple', 'grape', 'mango')
In [3]: print(type(fruits))
<class 'tuple'>
```

Indeksiranje, petlja i konkatenacija (operator +) za n-torke su isti kao i za liste.

Pošto su n-torke nepromenljive, ne možemo direktno da promenimo ni jednu informaciju uskladištenu u n-torkama. Međutim, možemo ih konvertovati u liste i zatim dodeliti izmenjenu listu bilo kojoj n-torci. Pogledajte sledeći primer:

```
In [4]: temp_list = list(fruits)

In [5]: temp_list.append('papaya')

In [6]: fruits = tuple(temp_list)

In [7]: fruits
Out[7]: ('apple', 'grape', 'mango', 'papaya')
```

U prethodnom primeru, demonstrirali smo upotrebu rutine **tuple()**. Na ovaj način možemo pametno da koristimo sve rutine lista za upotrebu u n-torkama.

Sada ćemo da vidimo demonstraciju metoda **count()** za brojanje koliko puta se određeni član elementa pojavljuje u n-torci:

```
In [8]: test_tuple = (2, 3, 1, 3, 1, 4, 5, 6, 3, 6)
In [9]: x = test_tuple.count(3)
In [10]: print(x)
3
```

2.4 Skupovi

Liste i n-torke su uređene strukture podataka i obe dozvoljavaju duple vrednosti. Skupovi se razlikuju od lista i n-torke, jer su nisu uređeni i stoga ne dozvoljavaju duple vrednosti. Skupovi se definišu pomoću vitičastih zagrada. Ovo je primer jednostavnih skupova:

```
In [12]: set1 = {'apple', 'banana', 'orange'}
In [13]: set1
Out[13]: {'apple', 'banana', 'orange'}
In [14]: set2 = set(('apple', 'banana', 'orange'))
In [15]: set2
Out[15]: {'apple', 'banana', 'orange'}
In [16]: print(type(set1))
<class 'set'>
```

Ne možemo koristiti indekse za preuzimanje elemenata bilo kog skupa, pošto skupovi nisu uređeni. Ali, možemo koristiti konstrukcije petlje for i while. Probajte ovo kao vežbu.

Možemo dodati nove stavke pomoću rutine **add()** na sledeći način:

```
In [17]: set1
Out[17]: {'apple', 'banana', 'orange'}
In [18]: set1.add('pineapple')
In [19]: set1
Out[19]: {'apple', 'banana', 'orange', 'pineapple'}
```

Možemo koristiti rutine **remove()** ili **discard()** da bismo uklonili stavku sa bilo koje liste na sledeći način:

```
In [20]: set1.remove('banana')
In [21]: set1.discard('apple')
```

Obe rutine izazivaju greške ako pokušamo da uklonimo nepostojeće stavke.

Hajde da vidimo nekoliko postavljenih metoda. Prvo ćemo videti kako da izračunamo uniju dva skupa. Sada ćemo da kreiramo skupove za to:

```
In [22]: set1 = {1, 2, 3, 4, 5}
In [23]: set2 = {3, 4, 5, 6, 7}
In [24]: set3 = set1.union(set2)
In [25]: set3
Out[25]: {1, 2, 3, 4, 5, 6, 7}
```

Ovde skladištimo uniju u novom skupu. Blagi alternativni pristup čuva uniju u prvom skupu na sledeći način,

```
In [29]: set1.update(set2)
In [30]: set1
Out[30]: {1, 2, 3, 4, 5, 6, 7}
```

Takođe možemo ukloniti sve elemente iz skupa na sledeći način:

```
In [31]: set3.clear()
```

Rutina **copy()** funkcioniše na sličan način kao lista. Sada ćemo da izračunamo razliku na sledeći način:

```
In [32]: set3 = set1.difference(set2)
In [33]: set3
Out[33]: {1, 2}
```

Ovaj primer vraća novi skup rezultata. Možemo ukloniti odgovarajuće elemente iz jednog od skupova koristeći ovo:

```
In [34]: set1.difference_update(set2)
In [35]: set1
Out[35]: {1, 2}
```

Možemo izračunati presek na sledeći način:

```
In [37]: set3 = set1.intersection(set2)
In [38]: set3
Out[38]: {3, 4, 5}
```

Možemo proveriti da li je skup podskup drugog skupa na sledeći način:

```
In [39]: set2 = {1, 2, 3, 4, 5, 6, 7, 8}
In [40]: set1.issubset(set2)
Out[40]: True
```

Slično tome, možemo proveriti da li je skup nadskup drugog skupa:

```
In [41]: set2.issuperset(set1)
Out[41]: True
```

Takođe možemo da proverimo da li su dva skupa disjunktna (nemaju ni jedan zajednički element) na sledeći način:

```
In [42]: set1 = {1, 2, 3}
In [43]: set2 = {4, 5, 6}
In [44]: set1.isdisjoint(set2)
Out[44]: True
```

Možemo izračunati simetričnu razliku između dva skupa na sledeći način:

```
In [45]: set1 = {1, 2, 3}
In [46]: set2 = {2, 3, 4}
In [47]: set3 = set1.symmetric_difference(set2)
In [48]: set3
Out[48]: {1, 4}
```

Takođe možemo izračunati uniju i presek pomoću operatora **|** i **s** na sledeći način:

```
In [49]: set1 | set2
Out[49]: {1, 2, 3, 4}
In [50]: set1 & set2
Out[50]: {2, 3}
```

2.5 Rečnici

Rečnici su uređeni, promenljivi i ne dozvoljavaju duplikate. Rečnici u Python-u 3.6 su neuređeni. Rečnici iz Python-a 3.7 su uređeni. Stavke se čuvaju u rečniku u parovima **key:value** i na njih se može pozvati korišćenjem naziva ključa. Sada ćemo da kreiramo jednostavan rečnik na sledeći način:

```
In [52]: test_dict = { "fruit": "mango", "colors": ["red", "green", "yellow"]}
```

Stavkama rečnika možemo pristupiti pomoću ključa (key) na sledeći način:

```
In [53]: test_dict["fruit"]
Out[53]: 'mango'
```

```
In [54]: test_dict["colors"]
Out[54]: ['red', 'green', 'yellow']
```

Možemo da preuzmemo ključeve (keys) i vrednosti (values) na sledeći način:

```
In [55]: test_dict.keys()
Out[55]: dict_keys(['fruit', 'colors'])

In [56]: test_dict.values()
Out[56]: dict_values(['mango', ['red', 'green', 'yellow']])
```

Možemo ažurirati vrednost na sledeći način:

```
In [60]: test_dict.update({"fruit": "grapes"})
In [61]: test_dict
Out[61]: {'fruit': 'grapes', 'colors': ['red', 'green', 'yellow']}
```

Možemo dodati u rečnik na sledeći način:

```
In [62]: test_dict["taste"] = ["sweet", "sour"]
In [63]: test_dict
Out[63]:
{'fruit': 'grapes',
 'colors': ['red', 'green', 'yellow'],
 'taste': ['sweet', 'sour']}
```

Takođe možemo izbaciti stavke:

```
In [64]: test_dict.pop("colors")
Out[64]: ['red', 'green', 'yellow']
In [65]: test_dict
Out[65]: {'fruit': 'grapes', 'taste': ['sweet', 'sour']}
```

Takođe možemo izbaciti poslednju umetnutu stavku:

```
In [66]: test_dict.popitem()
Out[66]: ('taste', ['sweet', 'sour'])
```

Takođe možemo izbrisati stavku,

```
In [67]: del test_dict["fruit"]
In [68]: test_dict
Out[68]: {}
```

Takođe možemo izbrisati rečnik na sledeći način:

```
In [69]: del test_dict
In [70]: test_dict
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-70-6651ddf27d40> in <module>
----> 1 test_dict

NameError: name 'test_dict' is not defined
```

Možemo da prođemo kroz rečnike, pomoću konstrukcije petlje for i while. Pokušajte da ovo uradite kao vežbu.

Rezime

U ovom poglavlju naučili smo osnove skupova, n-torki, lista i rečnika u Python-u. Zajedno, oni su poznati kao kolekcije u Python-u.

U sledećem poglavlju ćete naučiti osnove znakovnih nizova, funkcija i rekurzije.

Poglavlje 3 – Znakovni nizovi, funkcije i rekurzija

U prethodnom poglavlju smo istražili Python kolekcije koje su uključivale liste, n-torke, skupove i rečnike. Takođe smo počeli da pišemo male isečke koda. Naučili smo mnogo ugrađenih funkcija za kolekcije. Takođe smo istražili IPython konzolu.

U ovom poglavlju ćemo dublje „zaroniti“ u Python programiranje. Takođe ćemo početi da koristimo IDLE. Ovo je lista tema koje ćemo obraditi u ovom poglavlju:

- znakovni nizovi u Python-u
- funkcije
- rekurzija
- direktna i indirektna rekurzija

Nakon što pročitate ovo poglavlje u celosti, trebalo bi da razumete koncepte i programiranje znakovnih nizova i funkcija u Python-u.

3.1 Znakovni nizovi u Python-u

U ranijim poglavljima, dok smo koristili rutinu **print()** i kolekcije (liste, n-torke, skupovi i rečnici), intenzivno smo koristili i znakovne nizove. Nisam ovo spomenuo, jer sam želeo da to razmotrimo u posebnom poglavlju, jer je važno da razumete osnove. U ovom odeljku ćemo detaljno razmotriti znakovne nizove u Python-u: naučićete osnove i povezane rutine.

Sada ćemo otvoriti IPython u komandnoj liniji i početi da učimo. Možemo kreirati znakovni niz, koristeći par jednostrukih ili dvostrukih navodnika. Evo primera:

```
In [1]: 'Python'
Out[1]: 'Python'
In [2]: "Python"
Out[2]: 'Python'
```

Kao što vidimo, možemo koristiti par jednostrukih ili dvostrukih navodnika, ali ne i kombinaciju oba. Sledi primer:

```
In [3]: 'Python"
  File "<ipython-input-3-580e07628eb0>", line 1
    'Python"
            ^
SyntaxError: EOL while scanning string literal
```

Podaci o znakovnim nizovima se čuvaju na isti način bilo da koristimo par jednostrukih ili dvostrukih navodnika. U sledećem primeru se upoređuju znakovni nizovi koji su kreirani pomoću para jednostrukih i dvostrukih navodnika: