Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Kratki uvod u Python

Renata Vlahović Kruc

22. siječnja 2020.

Sadržaj

1 Uvod

Skripta *Kratki uvod u Python* napisana je u svrhu vježbi iz kolegija **Osnove algoritama** (https://web.math.pmf.unizg.hr/nastava/oa/index.php) koji je predviđen kao izborni kolegij na drugoj godini preddiplomskog studija Matematika, smjer nastavnički na Prirodoslovno-matematičkom fakultetu u Zagrebu.

2 Povijest Pythona

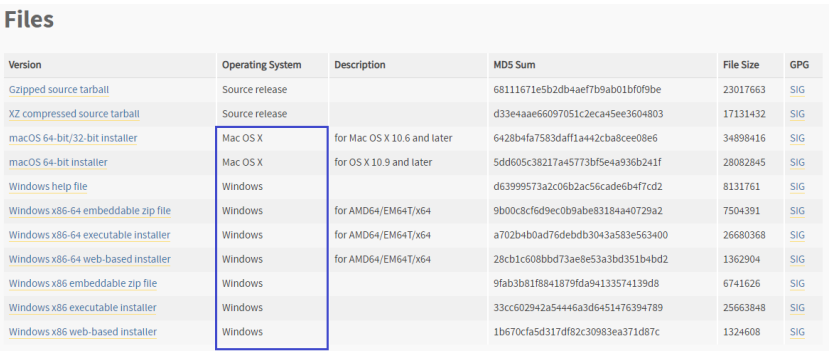
Programski jezik Python stvorio je Guido Van Rossum u kasnim osamdesetima tražeći hobi koji će ga zabaviti tijekom božićnog odmora. Svoju prvu verziju 0.9.0. objavio je u prosincu 1991. godine. Programski jezik je dobio ime po britanskoj humorističnoj seriji *Monty Python's Flying Circus*, čiji je veliki obožavatelj bio Van Rossum. Do sada su objavljenje tri verzije Pythona. Python 1.0 objavljen je 1994. godine, Python 2.0 objavljen je tek 16 godina kasnije, a Python 3.0 objavljen je 2008. godine.

3 Instalacija Pythona 3.7

Za rješavanje programskih zadataka zadanih na vježbama potrebno je instalirati programski jezik Python 3.7. Instalacija je besplatna i dostupna na stranici

https://www.python.org/downloads/release/python-374/.

Prilikom preuzimanja datoteke za instalaciju, potrebno je obratiti pozornost na operacijski sustav računala na kojem želite instalirati programski jezik Python (Slika 1).



Files

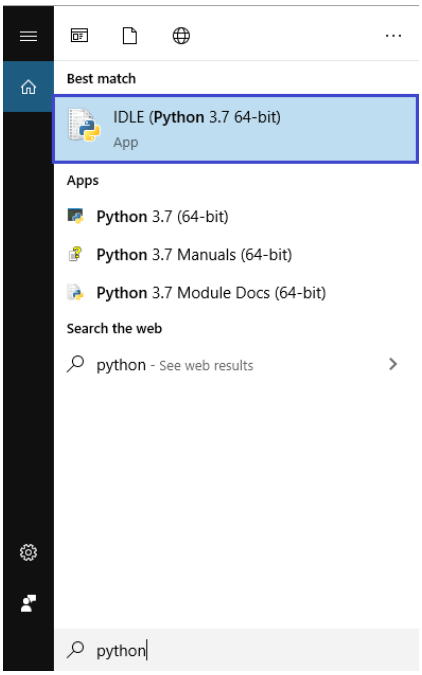
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda9413574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

Slika 1: Instalacija Pythona 2.3

4 Pokretanje programa

Pokretanje programa u Pythonu može se izvoditi na dva načina: korištenjem interaktivnog sučelja *Python Shell-a* ili pozivanjem Python interpretera[1] za program napisan u nekom tekstualnom editoru (*Notepad, Notepad++,...*).

IDLE (*Integrated Development and Learning Environment*) je ujedno editor i interaktivno sučelje za interpreter Pythona. S obzirom na to, IDLE ima dvije glavne vrste prozora: Shell i Editor. Prilikom instalacije Pythona na Windowsima, automatski je instaliran IDLE GUI (*Graphical User Interface*). Kako biste započeli programiranje u Pythonu, otvorite IDLE na svojem računalu korištenjem izbornika `START` (Slika 2).

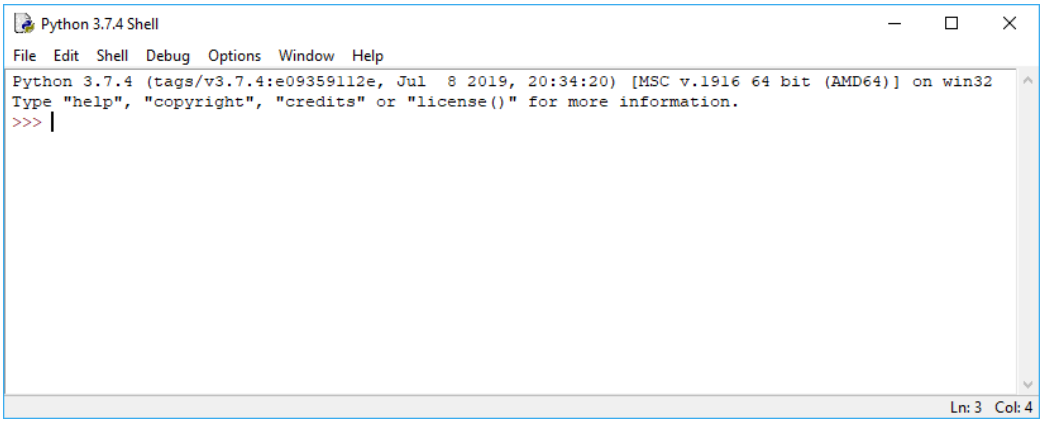


Slika 2: Otvaranje Python Shell-a

Ukoliko koristite operacijski sustav Linux, možda ćete morati dodatno instalirati IDLE. Pokretanje IDLE-a na Linuxu možete pomoću tekstualnog sučelja upisivanjem naredbe `idle` (ili `idle3`).

[1]Interpreter je računalni program koji izravno izvšava napisani kod u nekom programskom jezikom, bez da ga prije izvršavanja prevede na strojni jezik.

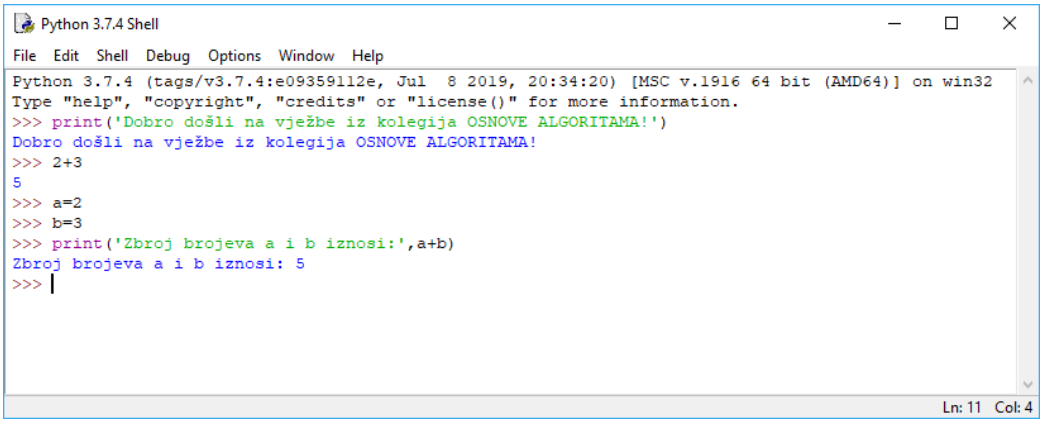
Nakon pokretanja IDLE-a, otvorit će se Python Shell (Slika 3).



Slika 3: Python Shell

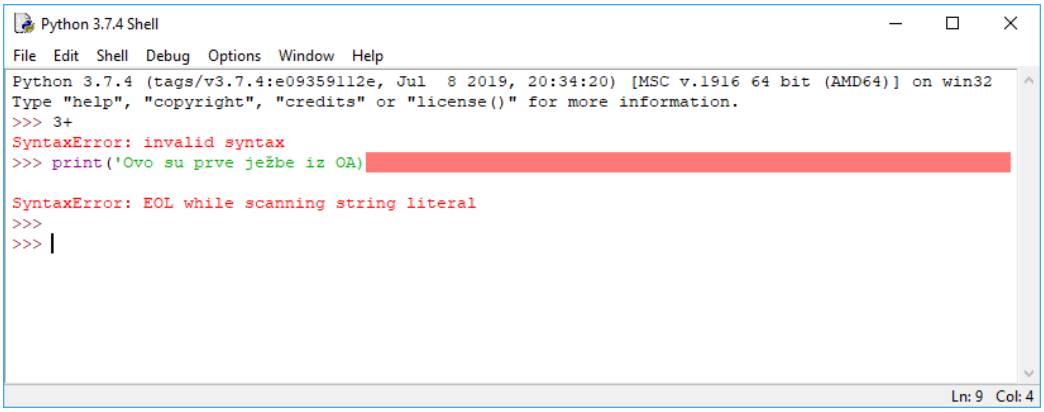
Python Shell je interaktivno sučelje u kojem možete pisati jednostavne programe. On vam može poslužiti za isprobavanje naredbi i nekih ideja.

Nakon unošenja naredbi, za pokretanje programa dovoljno je pritisnuti tipku `ENTER` nakon čega će se ispisati izlazni podaci (Slika 4).



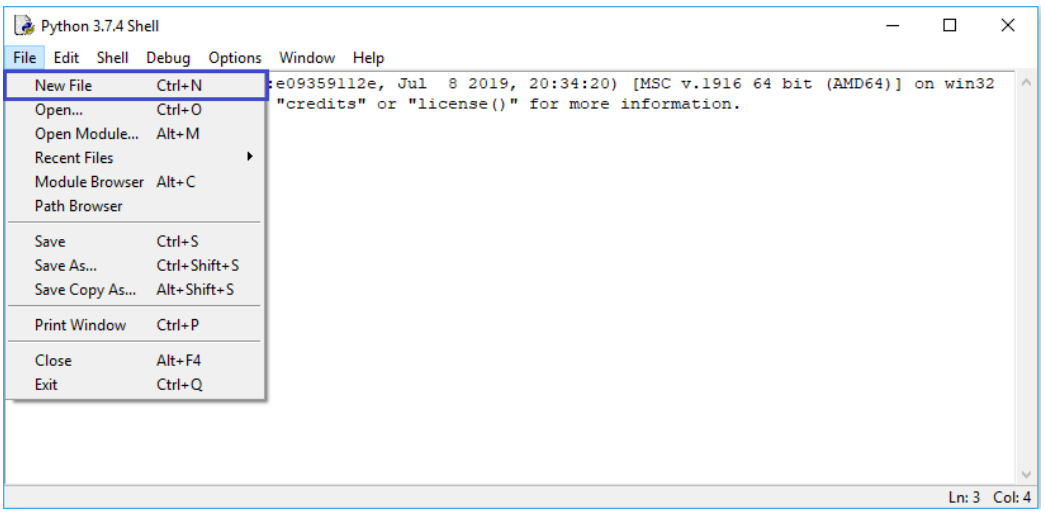
Slika 4: Pisanje programa u Python Shell-u

Ukoliko prilikom unošenja naredbi pogriješite, pri izvršavanja programa, javit će se obavijest o greški (Slika 5).



Slika 5: Greška u programu

Za složenije programe koji imaju više linija koda, potrebno je koristiti tekstualne editore. Jedan od načina je pokretanje editora unutar IDLE-a u izborniku `File` → `New File` (Slika 6).

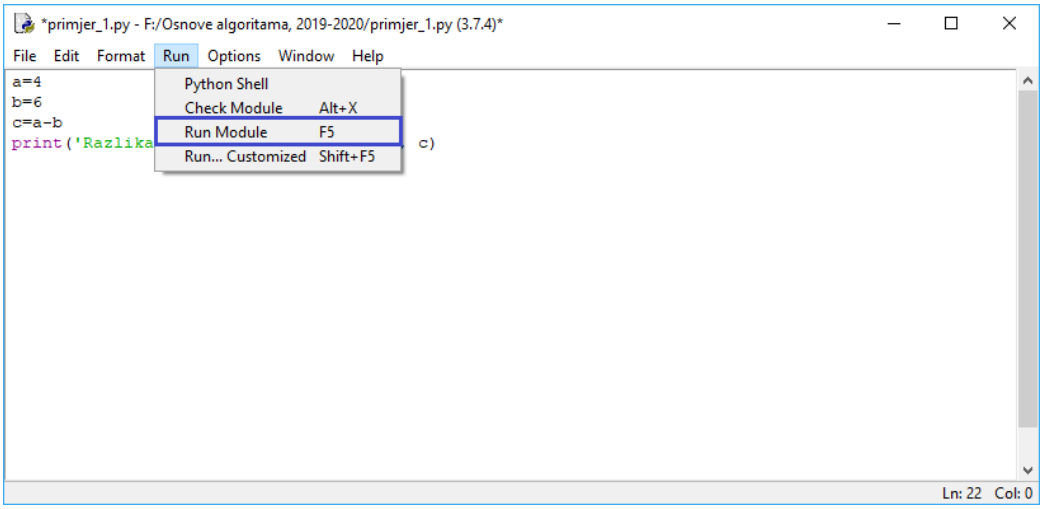


Slika 6: Pisanje programa u editoru

U tekstualni editor upisujete kod programa. Prije pokretanja programa, potrebno ga je spremiti koristeći izbornik `File` → `Save As`. Datoteku je potrebnu spremiti u formatu `ime.py` (Slika 7).

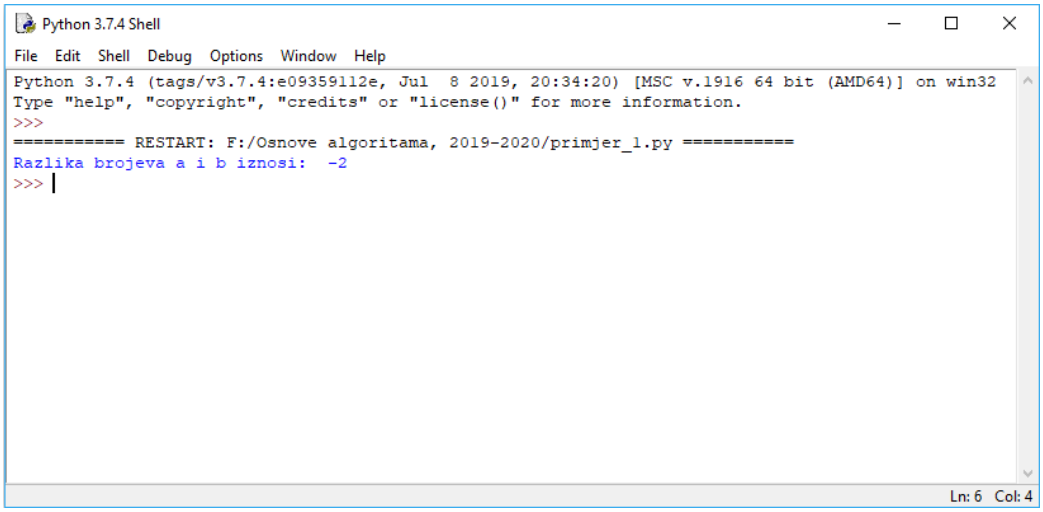
Slika 7: Spremanje programa napisanog u editoru

Kako bite pokrenuli izvršavanje napisanog programa u editoru, pritisnute u izborniku `Run` → `Run Module` ili `F5` (Slika 8).



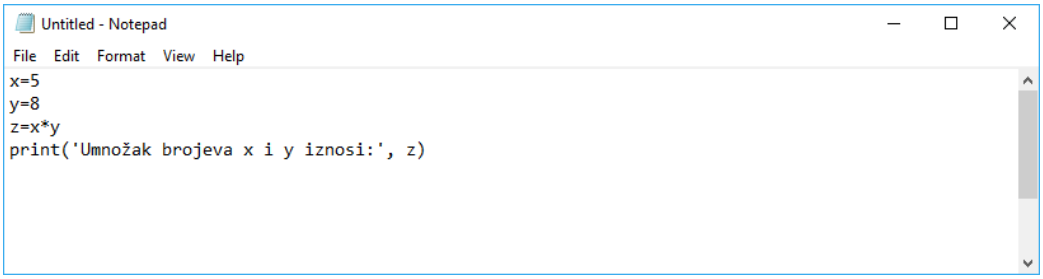
Slika 8: Pokretanje programa napisanog u editoru

Prilikom pokretanja programa na opisani način otvorit će se Python Shell u kojem će biti vidljivi izlazni podaci napisanog programa (Slika 9).



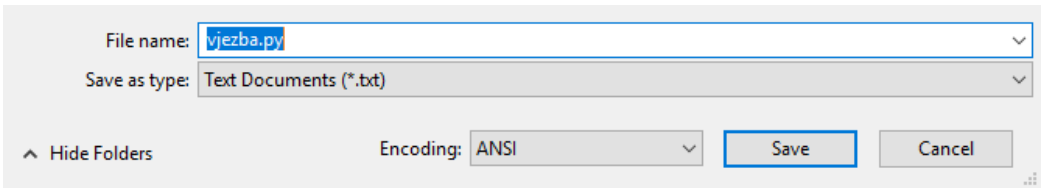
Slika 9: Rezultat programa ispisan u Python Shell-u

Osim korištenja editora unutar IDLE-a, za pisanje programa moguće je koristiti i mnoge druge tekstualne editore, na primjer `Notepad` (Slika 10).



Slika 10: Pisanje programa u Notepad-u

Kako biste pokrenuli napisani program, potrebno ga je prvo spremiti u formatu `ime.py` koristeći izbornik `File` → `Save As` (Slika 11).

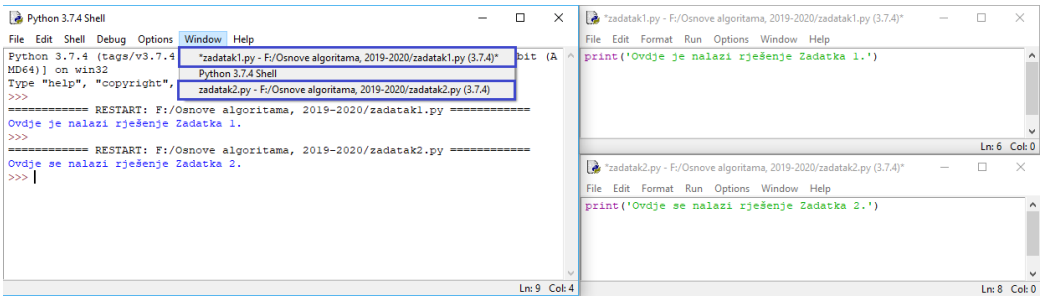


Slika 11: Spremanje programa napisanog u Notepad-u

Spremljenu datoteku potom je potrebno otvoriti u Python Shell-u koristeći izbornik `File` → `Open` i odabirom željene datoteke. Nakon toga po-

krenite izvršavanje programa na već opisan način `Run` → `Run Module`. Osim toga, program će se izvršiti i samim klikom na `.py` datoteku.

Istovremeno je moguće imati više otvorenih prozora za editiranje programa. Nove otvorene prozore za editiranje otvaramo u izborniku `File` → `New File`, a postojeće `File` → `Open`. Rezultati svih programa pojavit će se u istom Python Shell-u. Popis otvorenih datoteka možete vidjeti u izborniku `Window` (Slika 12).



Slika 12: Više otvorenih programa

5 Sintaksa Pythona 3.7

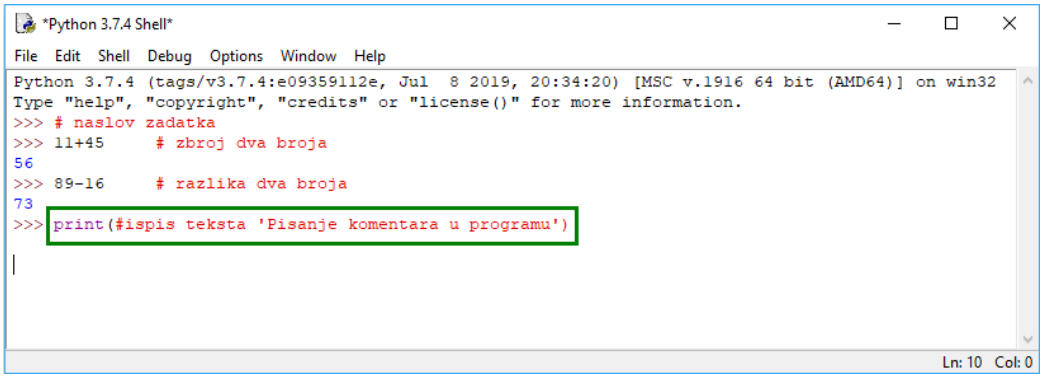
Postoji puno knjiga, priručnika, skripti i internetskih stranica na kojima možete pronaći materijale za učenje Pythona. Jedna od njih je *The Python Tutorial* [2].

U nastavku poglavlja su objašnjene naredbe koje su potrebne za rješavanje zadataka zadanih na vježbama.

5.1 Pisanje komentara

Pisanje komentara unutar programa je ponekad vrlo korisno. Programi koji su složeni često se sastoje od mnogo dijelova te je korisno napisati komentar na što se koji dio odnosi.

Komentari započinju sa znakom "#" te se protežu do kraja retka. Oni se mogu nalaziti na početku retka ili na kraju, nakon naredbi, ali ne i unutar naredbi (Slika 13).



Slika 13: Pisanje komentara

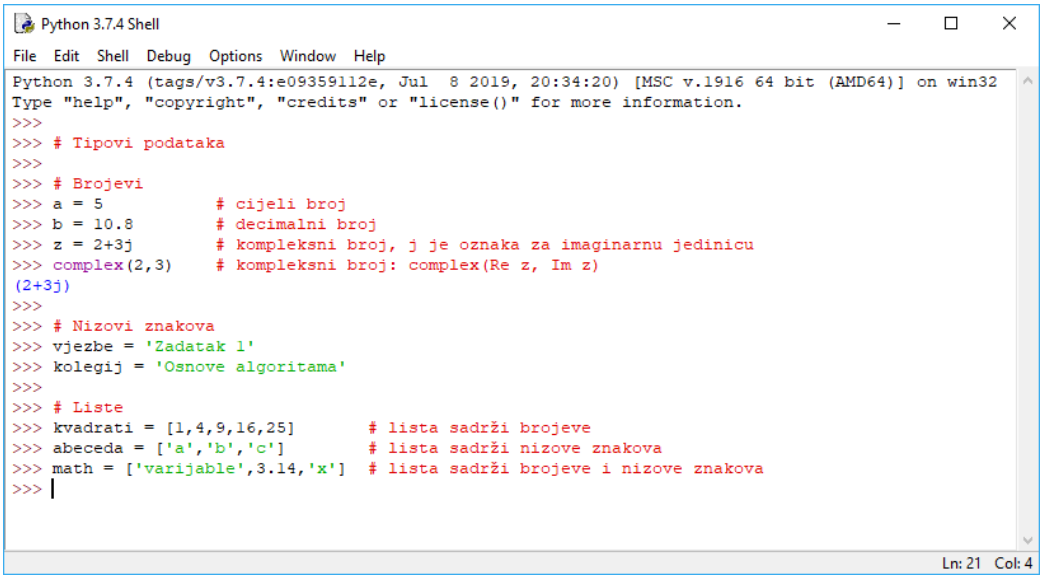
5.2 Definiranje varijabli

U programskom jeziku Python nije potrebno deklarirati varijable prije prve upotrebe, tj. odrediti koji će se tip podataka u njima spremati, već ih je dovoljno inicijalizirati, tj. pridružiti im neku vrijednost. Time se određuje tip podataka spremljen u pojedinoj varijabli[2].

Kako bismo inicijalizirali varijablu, koristimo znak "=". S lijeve strane znaka nalazi se ime varijable, a s desne strane vrijednost varijable.

[2]Više informacija o deklariranju i inicijaliziranju varijabli u programskim jezicima možete pronaći u skripti [1] na stranicama 21-22.

Python ima pet tipova podataka: brojevi (eng. *numbers*), niz znakova (eng. *string*), lista (eng. *list*), n-terac (eng. *tuple*) i rječnik (eng. *dictionary*). Na vježbama će se koristiti prva tri tipa podataka (Slika 14).



Slika 14: Tipovi podataka

5.3 Ulazni i izlazni podaci

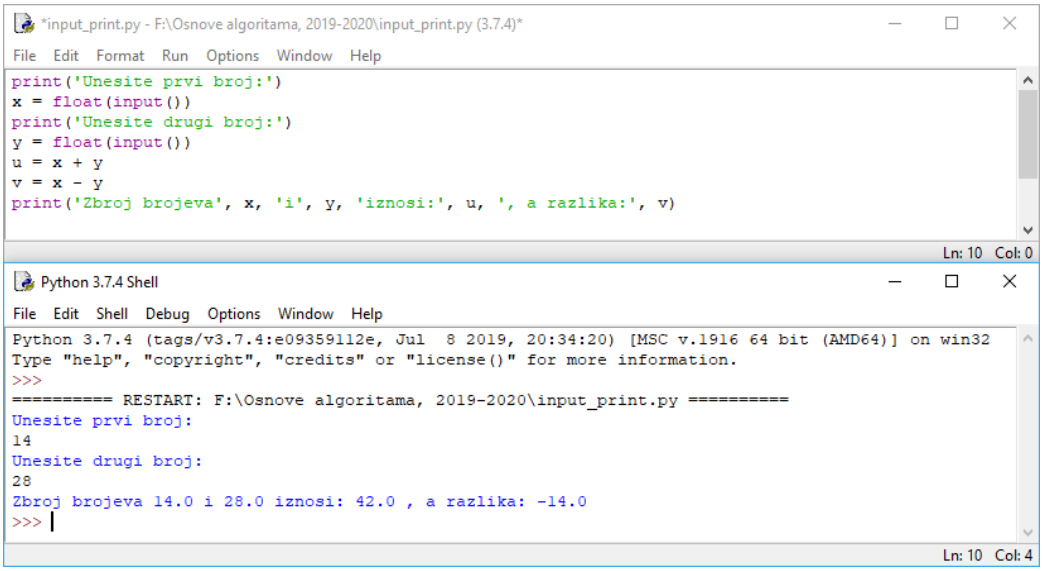
Svaki algoritam na početku uzima ulazne podatke, a na kraju vraća izlazne podatke koji predstavljaju rješenje problema.

Osim što se varijablama može pridružiti konstanta ili rezultat nekih računskih operacija, može joj se pridružiti i ulazni podatak pomoću naredbe `input()`. Kod korištenja naredbe `input()` potrebno je inicijalizirati tip podataka koji se unosi. U suprotnome, program će podatak koji unesete inicijalizirati kao niz znakova. Za unos cijeli brojeva potrebno je koristiti naredbu `int(input())`, a za unos decimalnih brojeva potrebno je koristiti naredbu `float(input())`.

Varijabli možemo također pridružiti i neki slučajno generirani realni broj iz intervala $[0.0, 1.0)$ pomoću naredbe `random.random()` ili neki slučajno generirani cijeli broj iz intervala $[a, b]$ pomoću naredbe `random.randint(a,b)`. Za korištenje navedenih naredbi potrebno je uključiti modul `random`. Naredba je `import random`.

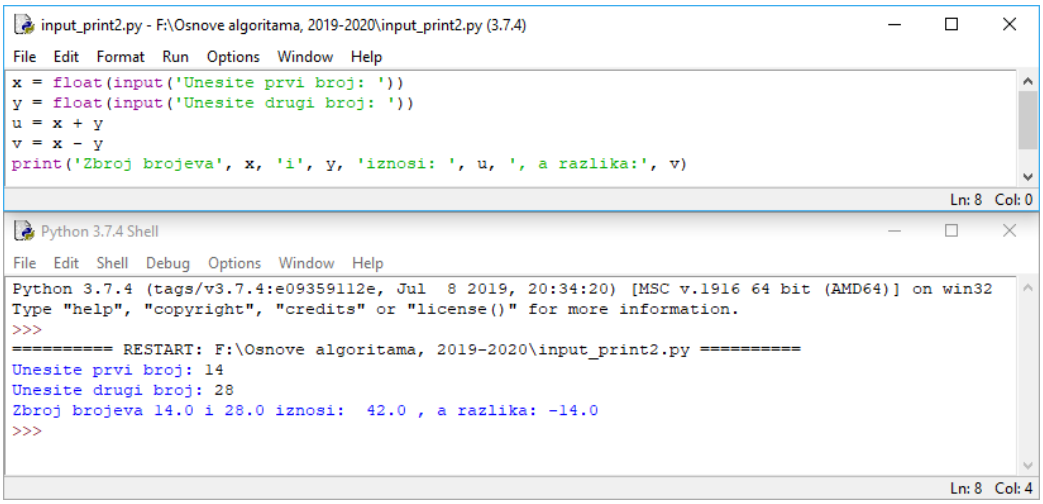
U prethodnim primjerima vidjeli smo da izlazne podatke možemo ispisati koristeći naredbu `print`. Ukoliko želimo ispisati određeni tekst potrebno ga je staviti u navodnike, dok ispisivanje vrijednosti varijabli ne stavljamo

u navodnike. Unutar jedne naredbe `print` možemo ispisati više izlaznih podataka, pri čemu ih je potrebno odvojiti zarezima (Slika 15).



Slika 15: Ulazni i izlazni podaci

Program na Slici 15 može se napisati i jednostavnije. Naredbe `print('tekst')` i `input()` mogu se ujediniti u jednu naredbu: `input('tekst')` (Slika 16).



Slika 16: Ulazni i izlazni podaci - 2

5.4 Brojevi

Programski jezik Python možemo koristiti kao kalkulator i pomoću njega možemo zbrajati, oduzimati, množiti, dijeliti, potencirati, korijenovati,... Python ima tri osnovna tipa brojeva: cijeli, decimalni i kompleksni. Kompleksni broj $z = x+yi$ prikazat ćemo pomoću naredbe `complex(x,y)`. Želimo li očitati realni, odnosno imaginarni dio nekog kompleksnog broja z, koristit ćemo naredbu `z.real`, odnosno `z.imag`.

Za pretvaranje vrijednosti nekog broja x u cijeli broj koristimo naredbu `int(x)`. Nadalje, za pretvaranje vrijednosti nekog broja x u decimalni broj koristimo naredbu `float(x)`, dok za pretvaranje vrijednosti nekog broja x u kompleksni broj koristimo naredbu `complex(x)`.

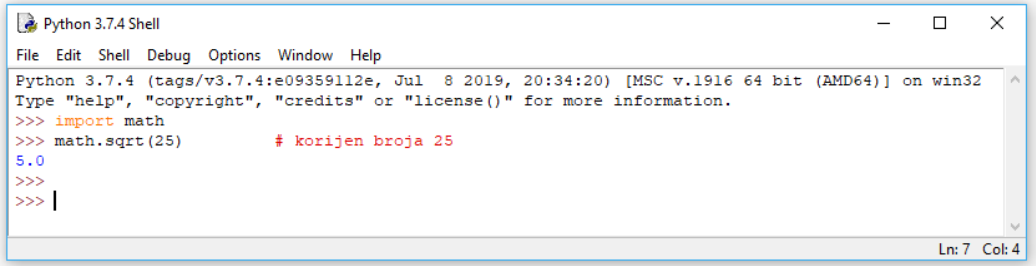
Osim toga, vrijednost nekog broja x možemo pretvoriti i u binarni, heksadekadski i oktalni zapis pomoću naredbi `bin(x)`, `hex(x)`, odnosno `oct(x)`.

Popis naredbi za neke računske operacije nalazi se u Tablici 1.

Računska operacija	**Rezultat**
`x + y`	zbroj brojeva x i y
`x - y`	razlika brojeva x i y
`x * y`	umnožak brojeva x i y
`x / y`	dijeljenje brojeva x i y
`x // y`	cjelobrojno dijeljenje brojeva x i y
`x % y`	ostatak pri dijeljenju broja x brojem y
`x ** y`	y-ta potencija broja x
`pow(x,y)`	y-ta potencija broja x
`abs(x)`	apsolutna vrijednost broja x
`round(x,n)`	zaokruživanje broja x na n decimala

Tablica 1: Računske operacije

Za računanje vrijednosti funkcija realnog broja potrebno je uključiti modul `math`, ili `cmath` za računanje s kompleksnim brojevima. Module uključujemo koristeći naredbe `import math`, odnosno `import cmath` (Slika 17).



Slika 17: Korištenje matematičkog modula

Popis naredbi za računanje vrijednosti nekih matematičkih funkcija nalazi se u Tablici 2.

Za računanje vrijednosti funkcija za kompleksne brojeve naredbe idu analogno s prefiksom `cmath`. Na primjer: `cmath.pow(x,y)`.

Matematičke funkcije	**Objašnjenje**
`math.pi`	broj π
`math.e`	broj e
`math.exp(x)`	x-ta potencija broja e
`math.floor(x)`	najveće cijelo broja x
`math.log(x)`	prirodni logaritam broja x
`math.log10(x)`	logaritam broja x po bazi 10
`math.log(x,b)`	logaritam broja x po bazi b
`math.pow(x,y)`	y-ta potencija broja x
`math.sqrt(x)`	drugi korijen broja x
`math.sin(x)`	sinus realnog broja x
`math.cos(x)`	kosinus realnog broja x
`math.tan(x)`	tangens realnog broja x
`math.degrees(x)`	pretvaranje kuta iz radijana u stupnjeve
`math.radians(x)`	pretvaranje kuta iz stupnjeva u radijane

Tablica 2: Matematičke funkcije

Osim navedenih naredbi, postoje još mnoge koje može naći na internetskim stranicama [3] i [4].

5.5 Niz znakova

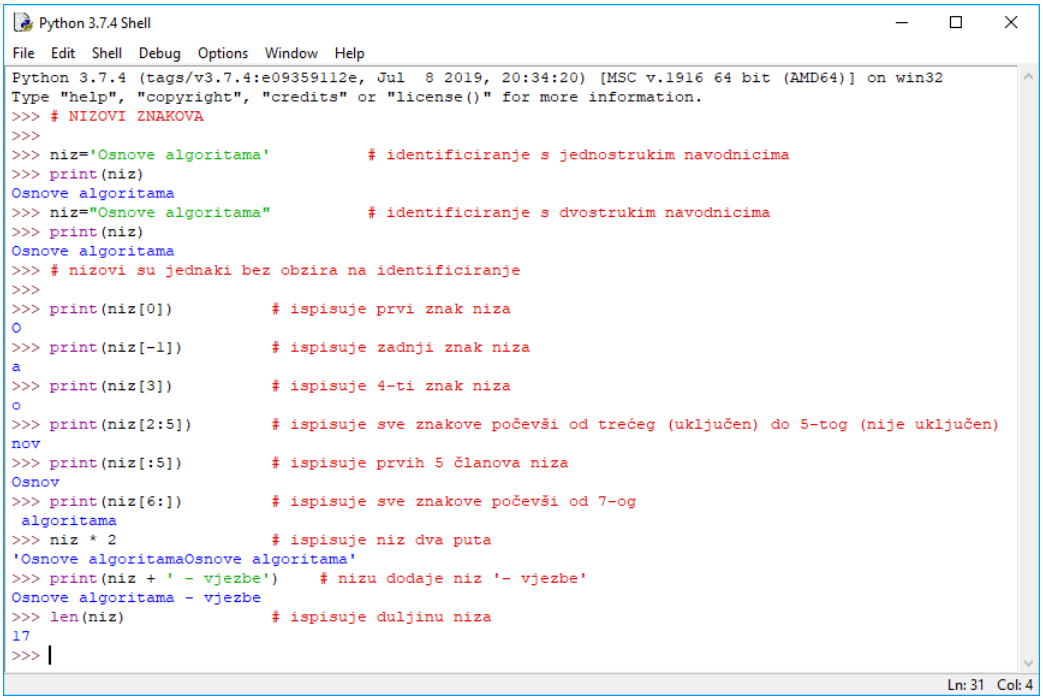
Niz znakova (eng. *string*) u Pythonu identificiramo kao skup znakova unutar jednostrukih `'...'` ili dvostrukih navodnika `"..."`. Ukoliko niz znakova sadrži jednostruke navodnike, za njegovo identificiranje potrebno je korištenje dvostrukih navodnika, i obrnuto.

Svaki niz znakova može se indeksirati na način da prvi znak u nizu ima indeks 0, drugi indeks 1, itd. Nizove možemo indeksirati i negativnim brojevima na način da zadnji znak u nizu ima indeks -1, predzadnji -2, itd.

Indeksiranje nizova znakova omogućava ispis podskupa nekog niza pomoću operatora `[ ]`, ako želimo ispisati samo jedan znak niza, ili pomoću operatora `[ : ]`, ako želimo ispisati više znakova niza.

Nizovi se mogu povezivati pomoću operatora `+`, ponavljati pomoću operatora `*`, a duljina niza može se izračunati koristeći naredbu `len()`.

Primjeri korištenja nizova znakova u Pythonu prikazani su na Slici 18.



Slika 18: Nizovi znakova

5.6 Liste

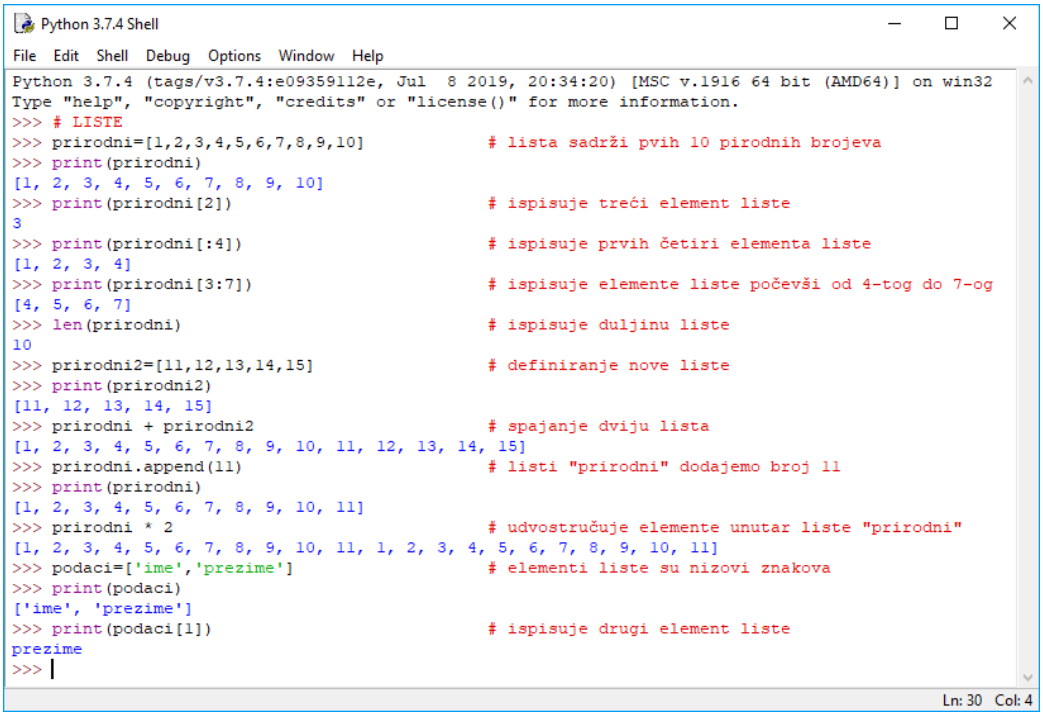
Liste (eng. *lists*) u Pythonu identificiramo koristeći uglate zagrade `[ ]`. Lista može sadržavati brojeve i nizove znakova istovremeno. Elemente liste odvajamo zarezima.

Slično kao i kod nizova znakova, prvi element liste može se indeksirati indeksom 0, drugi indeksom 1, i tako dalje; ili zadnji indeksom -1, predzadnji indeksom -2, i tako dalje. Također, za ispisivanje određenih elemenata liste koristi se operator `[ ]`, ukoliko želimo ispisati samo jedan element liste, ili `[ : ]`, ukoliko želimo ispisati više elemenata liste.

Liste se mogu spajati pomoću operatora `+` , ponavljati pomoću operatora `*` , a duljina liste može se izračunati pomoću naredbe `len()`.

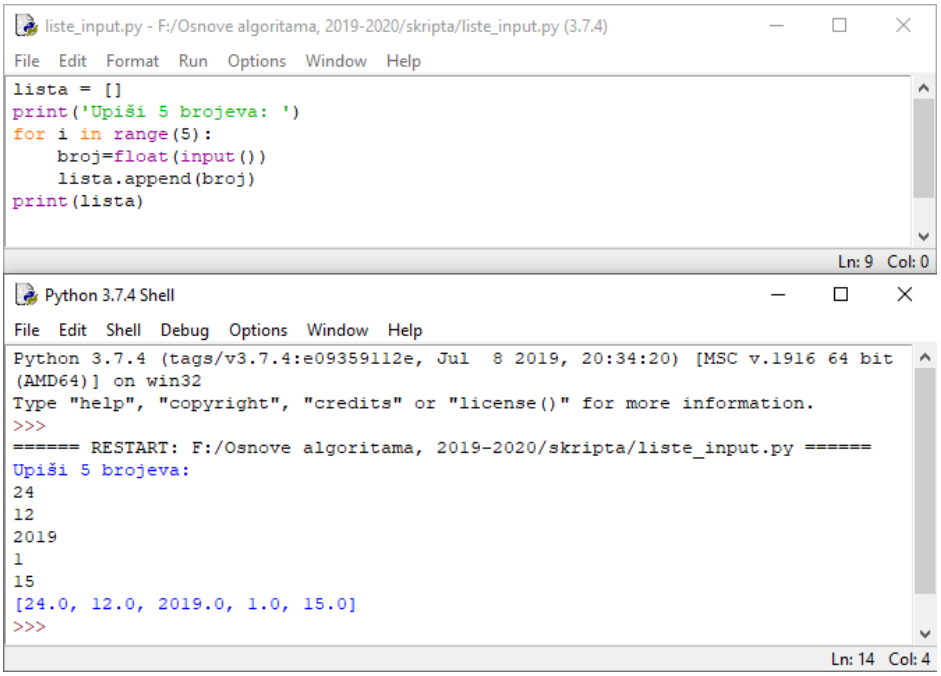
U neku listu se mogu dodati i pojedinačni elementi pomoću naredbe `imeliste.append()`.

Primjeri korištenja nizova znakova u Pythonu prikazani su na Slici 19.

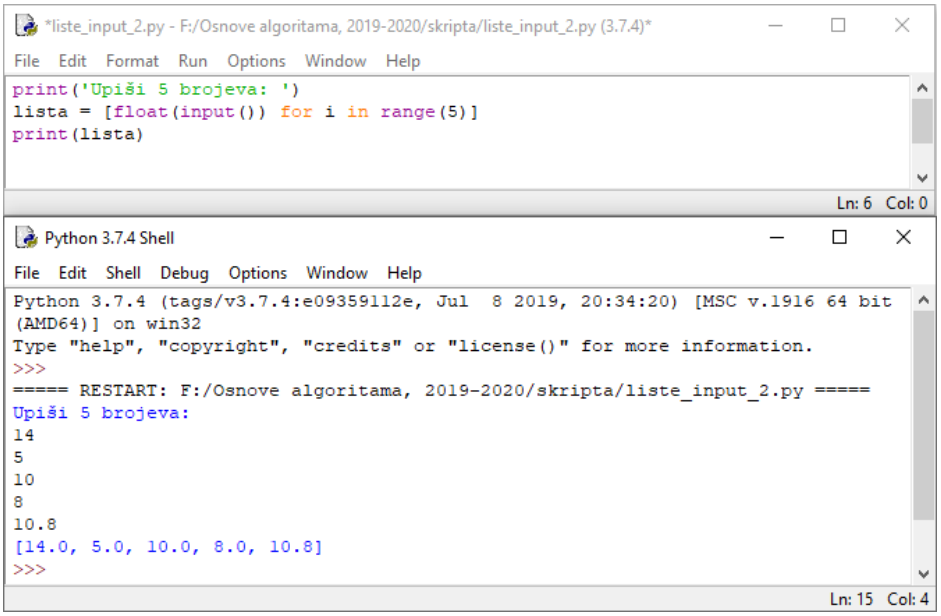


Slika 19: Liste

Na listu možemo gledati i kao ulazni podatak. Primjer programa u kojem se učitani podaci spremaju u listu prikazan je na Slici 20 i Slici 21.



Slika 20: Spremanje ulaznih podataka u listu - duži način



Slika 21: Spremanje ulaznih podataka u listu - kraći način

5.7 Naredba grananja *if*

Unutar nekog programa često je potrebno dio programa izvršiti, a dio ne, ovisno o postavljenom uvjetu. U tom slučaju koristimo naredbu grananja `if`.

Naredba grananja temelji se na logičkom tipu podataka `TRUE / FALSE`. Ako je neki uvjet ispunjen, onda će se određena naredba ili skup naredbi izvršiti, a ako uvjet nije ispunjen, onda se naredba, odnosno skup naredbi neće izvršiti. Sintaksa naredbe `if` glasi:

```
if uvjet:
    naredba ili skup naredbi
```

U programu je često potrebno izvršiti jednu naredbu ili skup naredbi ukoliko je neki uvjet ispunjen, odnosno izvršiti drugu naredbu ili skup naredbi ako uvjet nije ispunjen. U tom slučaju koristimo složeniju naredbu grananja `if-else` (dvostruko grananje). Naredbe koje će se izvršiti ukoliko je uvjet ispunjen nalaze se prije naredbe `else`, a naredbe koje će se izvršiti ukoliko uvjet nije ispunjen nalaze se nakon naredbe `else`. Sintaksa naredbe `if` glasi:

```
if uvjet:
    naredba ili skup naredbi
else:
    naredba ili skup naredbi
```

Prilikom pisanja uvjeta često se koriste logički operatori (Tablica 3) i operatori usporedbe (Tablica 4).

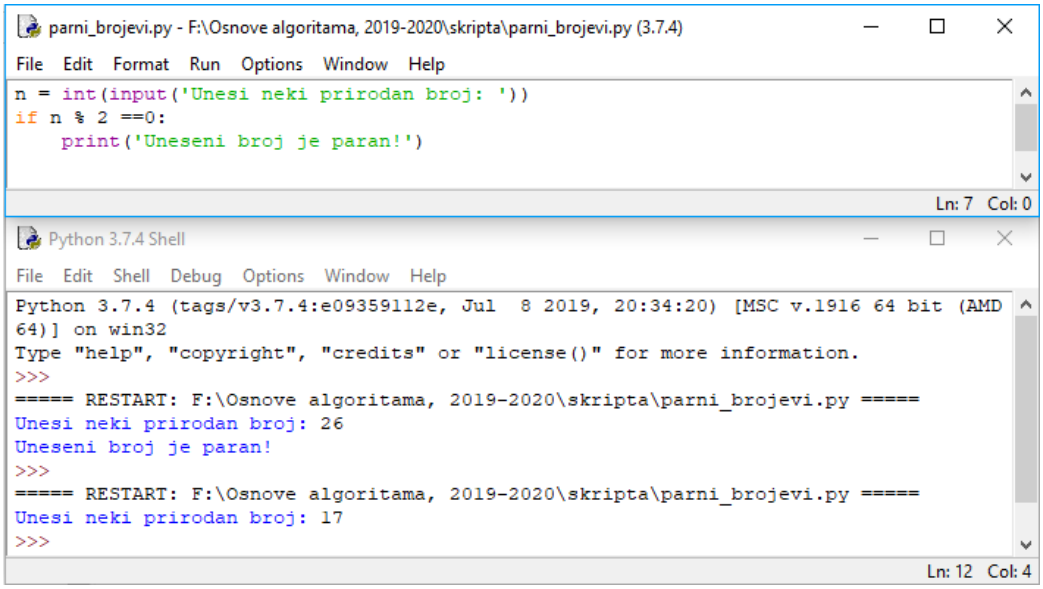
Operator	Značenje
`OR`	logički operator "ili"
`AND`	logički operator "i"
`NOT`	logički operator "ne"

Tablica 3: Logički operatori

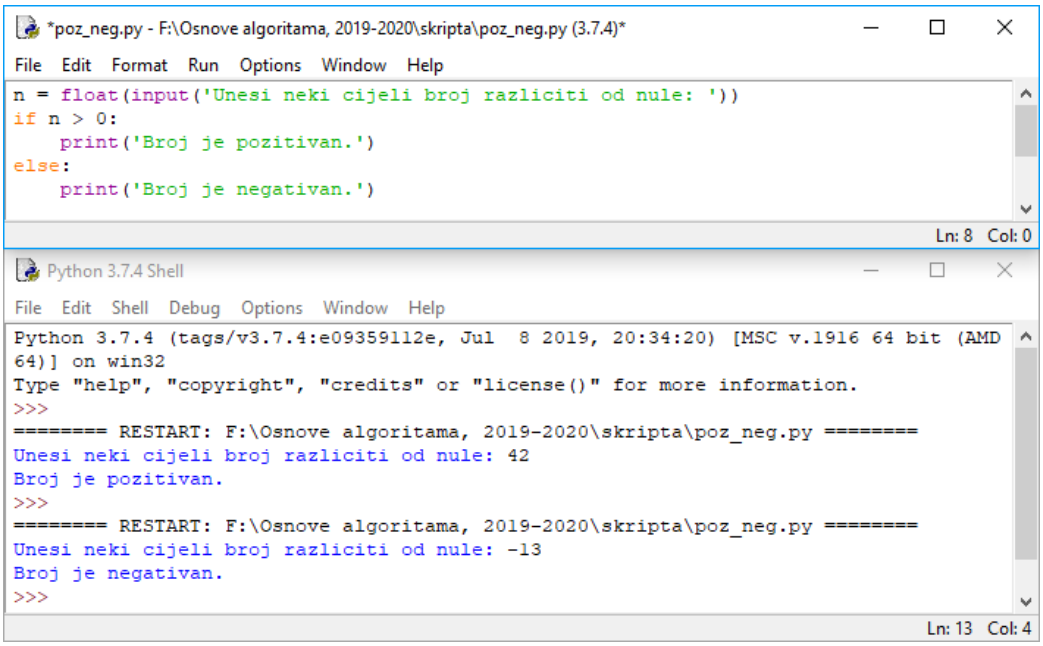
Operator	Značenje
`<`	strogo manje
`<=`	manje ili jednako
`>`	strogo veće
`>=`	veće ili jednako
`==`	jednako
`!=` ili `<>`	različito

Tablica 4: Operatori usporedbe

Primjer programa koji koristi naredbu `if` i operator `==` nalazi se na Slici 22, a primjer programa koji koristi naredbu `if-else` i operator `>` nalazi se na Slici 23:



Slika 22: Je li broj paran?



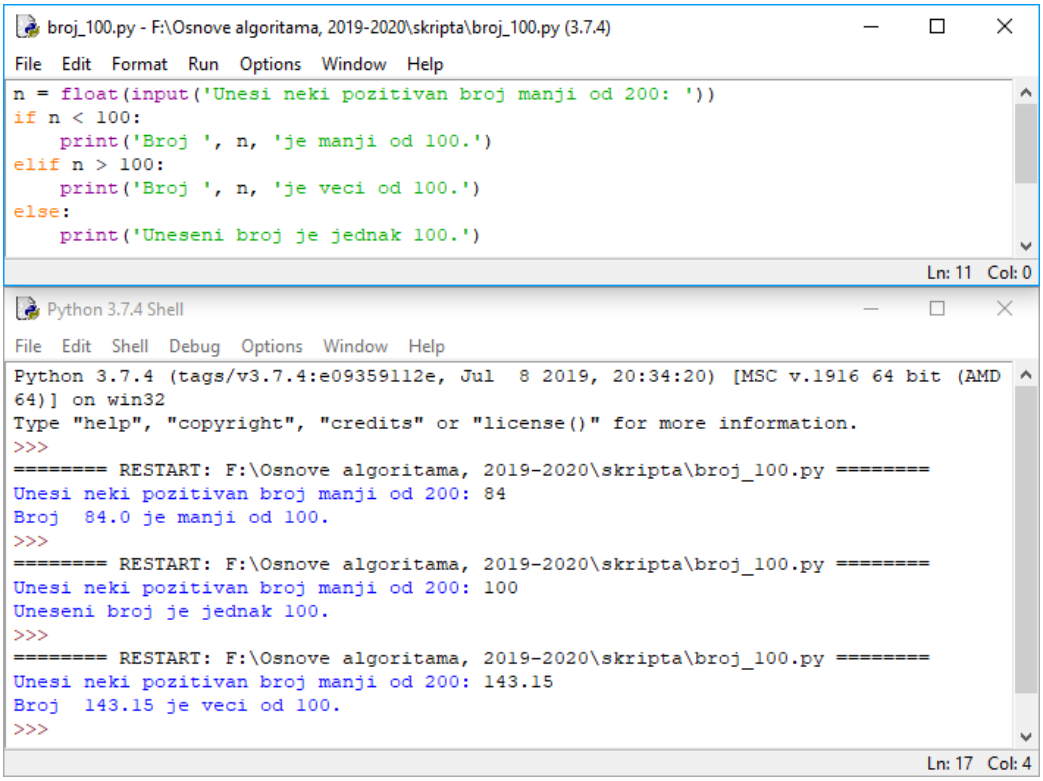
Slika 23: Pozitivan ili negativan broj?

Osim naredbi `if` i `if-else` postoji i naredba grananja `elif` koja se koristi kada je potrebno provjeriti više disjunktnih uvjeta te s obzirom na ispunjenje određenog uvjeta, izvršiti određene naredbe ili skup naredbi. Sintaksa naredbe `if-else` glasi:

`if` *uvjet 1*:
    naredba ili skup naredbi
`elif` *uvjet 2*:
    naredba ili skup naredbi
`elif` *uvjet 3*:
    naredba ili skup naredbi
...
`else`:
    naredba ili skup naredbi

Broj grana `elif` je proizvoljan te je grana `else` opcionalna.

Primjer programa koji koristi naredbu `elif` nalazi se na Slici 24:



Slika 24: Je li broj manji, jednak ili veći od 100?

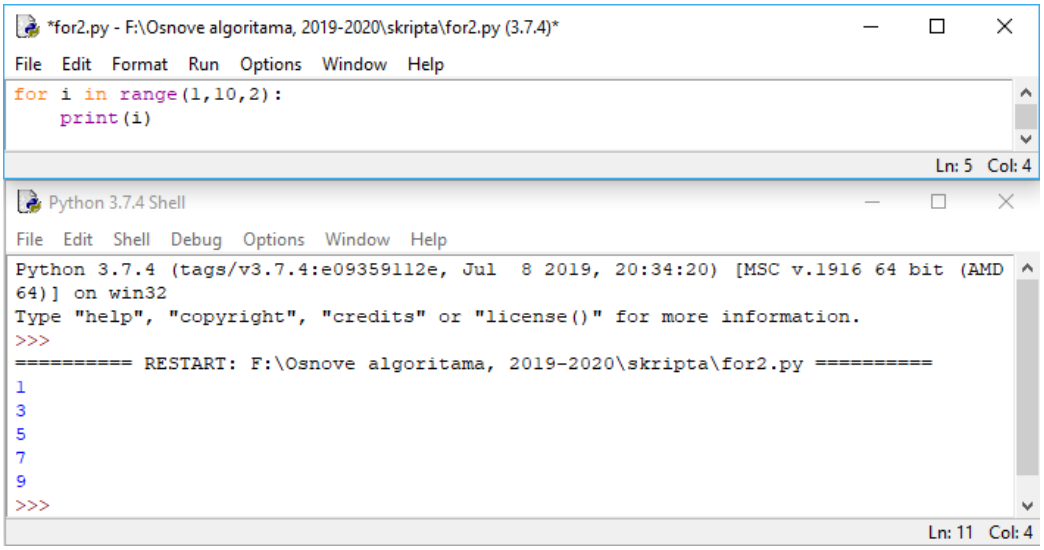
5.8 Petlje *for* i *while*

Petlje (engl. *loops*) u programiranju koristimo kada neku naredbu želimo ponoviti više puta.
Ukoliko neku naredbu želimo ponoviti točno određeni broj puta, koristimo petlju `for`. Sintaksa naredbe `for` glasi:

```
for i in range(a,b,k):
    naredba ili skup naredbi
```

pri čemu `i` označava brojač petlje, `a` je početna vrijednost brojača, `b` je završna vrijednost brojača, a `k` označava korak brojača. Naredba ili skup naredbi će se izvršavati sve dok brojač ne dostigne vrijednost `b`.

Primjer programa koji koristi opisanu petlju `for` nalazi se na Slici 25:

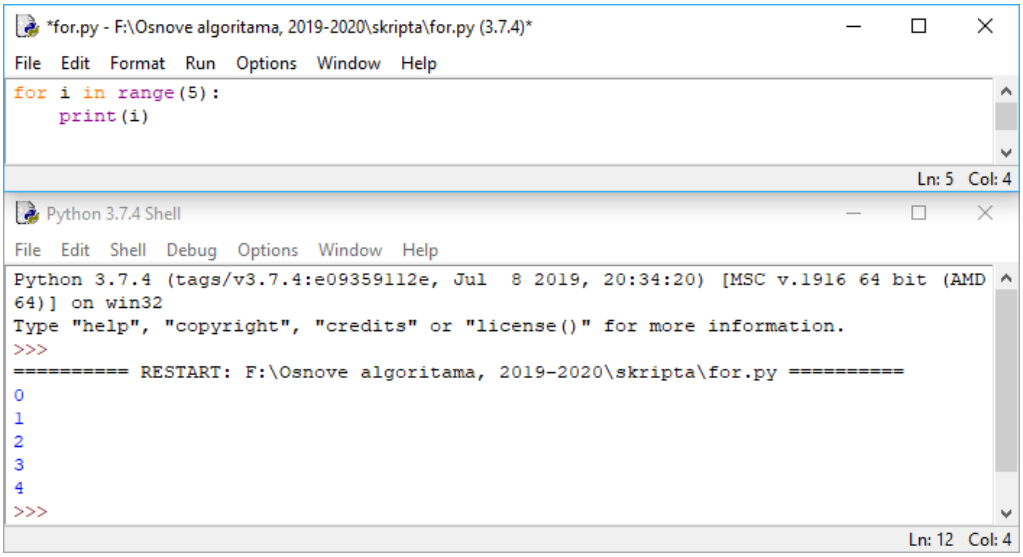


Slika 25: Ispis neparnih brojeva iz intervala $[1, 2, ...10)$

Ukoliko želimo da se neka naredba ponovi n puta, pri čemu je n neki prirodan broj, dovoljno je napisati:

```
for i in range(n):
    naredba ili skup naredbi
```

Primjer programa koji koristi opisanu petlju `for` nalazi se na Slici 26:

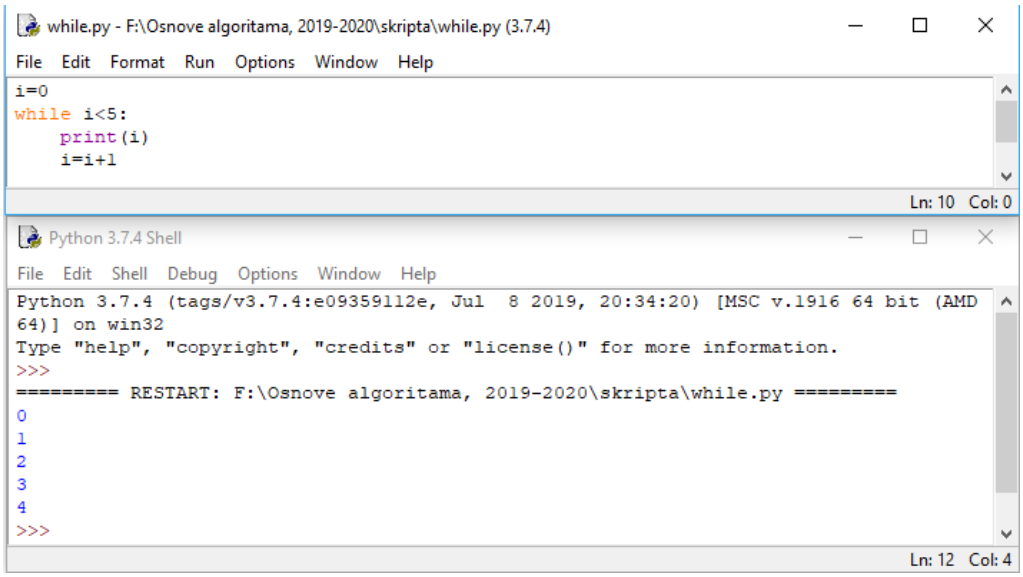


Slika 26: Ispis prvih pet brojeva iz skupa $\mathbb{N}_0$

Ukoliko neku naredbu ili skup naredbi želimo ponavljati sve dok je ispunjen neki uvjet, koristimo petlju `while`. Sintaksa naredbe `while` glasi:

`while` *uvjet*`:`
naredba ili skup naredbi

Primjer programa koji koristi petlju `while` nalazi se na Slici 27.



Slika 27: Ispis prvih pet brojeva iz skupa $\mathbb{N}_0$

Razlika između `for` i `while` petlje je što petlja `while` ne sadrži brojač, već joj izvšavanje neke naredbe više puta omogućuje dani uvjet. Uvjet u petlji `while` može biti određen aritmetičkim, logičkim i/ili relacijskim (usporednim) operatorima.

5.9 Matrice

Matrica je pravokutna tablica ispunjena brojevima. Na matrice možemo gledati kao dvodimenzionalne nizove, tj. nizove s dva indeksa. Prvi indeks označava redak, a drugi indeks stupac matrice. Ako matrica ima m redaka i n stupaca, govorimo o matrici tipa $m \times n$. Za učitavanje matrice A dimenzije $m \times n$ čiji su elementi realni brojevi koristimo sljedeću naredbu:

```
A = [[float(input()) for j in range(n)] for i in range(m)]
```

Analogno bismo koristili naredbu za učitavanje matrice čiji su elementi prirodni, odnosno cijeli brojevi. Ako matricu želimo ispuniti s nekim određenim brojem, taj broj upisat ćemo umjesto `float(input())`.

Da bismo ispisali matricu u standardnom obliku, koristimo sljedeću sintaksu:

```
for i in range(m):
    for j in range(n):
        print(A[i][j], end=' ')
    print()
```

Literatura

[1] V. Krčadinac, *Osnove algoritama*, skripta, Sveučilište u Zagrebu, PMF-Matematički odsjek 2016.
`https://web.math.pmf.unizg.hr/nastava/oa/oa-skripta.pdf`

[2] *The Python Tutorial*: `https://docs.python.org/3/tutorial/`

[3] *Python - Mathematical functions*:
`https://docs.python.org/3/library/math.html`

[4] *Python - Mathematical functions for complex numbers*:
`https://docs.python.org/3/library/cmath.html#module-cmath`

[5] Wikipedia, *History of Python*, listopad 2018.
`https://en.wikipedia.org/wiki/History_of_Python`