Osnovi programiranja

2018/19

O kursu

- **Model 70+30**
  - Redovno prisustvo nastavi = 4 poena
  - Test = 10 poena
  - 2 kolokvijuma = 20 + 36 poena
  - Završni deo ispita = 30 poena
- **Uslov za izlazak na završni ispit – više od polovine poena na predispitnim obavezama**

Najvažnije teme

- Cilj predmeta: programerska pismenost
- Uvod u programiranje
- Osnove programskog jezika Python
  - aritmetičke operacije
  - standardni ulaz/izlaz
  - ugrađene funkcije
  - upravljačke komande, grananje, ponavljanja
  - funkcije, rekurzivne funkcije
  - stringovi
- Strukture - liste, skupovi, torke, rečnici

Kompjuterski programi

Rešavanje problema korišćenjem kompjuterskih programa se može razložiti na više etapa:

1. Definisanje problema
2. Analiza problema
3. Definisanje algoritma
4. Projektovanje programa
5. Kodiranje
6. Testiranje
7. Analiza rezultata
8. Isporuka programa
9. Održavanje

Šta je algoritam?

- Algoritmi su korišćeni davno pre nego što je iko pokušao da definiše šta je to algoritam.

Primeri:

- Euklidov postupak za konstruisanje pomoću lenjira i šestara
- Euklidov postupak za određivanje NZD
- Vavilonski iterativni postupak za određivanje kvadratnog korena (1500 g.p.n.e.).

Šta je algoritam?
Postupak za određivanje kvadratnog korena

- Da biste odredili kvadratni koren bilo kog pozitivnog broja *S:*
  1. **Napravite početnu pretpostavku.** Pretpostavite da je vrednost kvadratnog korena od S neki pozitivni broj x_0.
  2. **Poboljšajte pretpostavljeni rezultat.** Primenite formulu $x_1 = (x_0 + S / x_0) / 2$. Broj x_1 je bolja aproksimacija za sqrt(S).
  3. **Ponavljajte korak 2 dok ne konvergirate ka rešenju.** Primenjujte formulu $x_{n+1} = (x_n + S / x_n) / 2$ sve dok se cifre brojeva x_{n+1} i x_n ne budu poklapale do u željenu decimalu.

Algoritmi

Reč algoritam, onako kako je mi danas poznajemo, prvi je upotrebio persijski matematičar Musa Al Horezmi pre više od 1200 godina.

- Konačan uređen niz precizno formulisanih pravila kojima se rešava jedan ili čitava klasa problema

Može se opisati tekstualno na prirodnom jeziku ili grafički

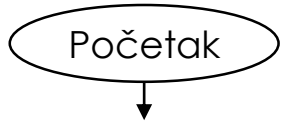
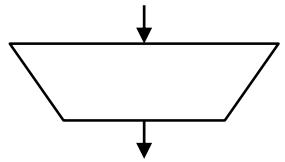
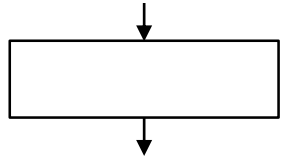
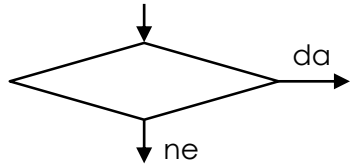
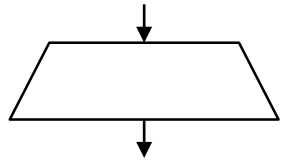
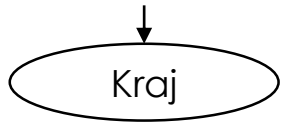
Zadatak

- Opisati algoritam za zamenu probušene gume na autu

Zadatak
Algoritam

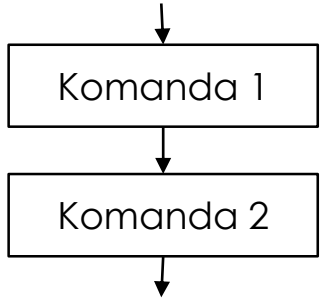
- Probušena guma na autu:
  - Potrebno: auto
  - Ako je guma probušena onda
    - Potrebno: rezervni točak, dizalica, ključ
    - Podići auto dizalicom
    - Ako ima neodšrafljenih šrafova
      - Odšrafiti šraf
    - Skinuti točak
    - Staviti rezervni točak
    - Ako ima nezašrafljenih šrafova
      - Zašrafiti šraf
    - Spustiti auto
    - Spakovati alat
  - Srećan put

Alogitmi – grafički simboli

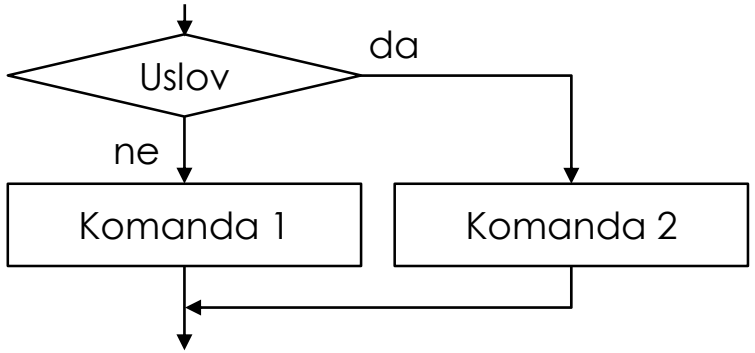
	Početak algoritma
	Ulazne veličine
	Izračunavanje
	Uslovno grananje
	Izlazne veličine
	Kraj algoritma

Alogitmi – osnovne strukture

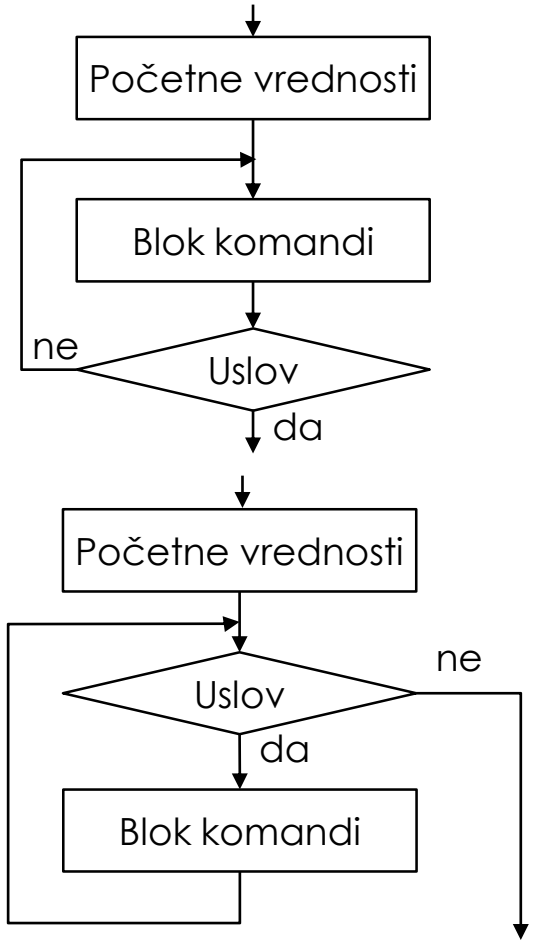
- Linijska struktura



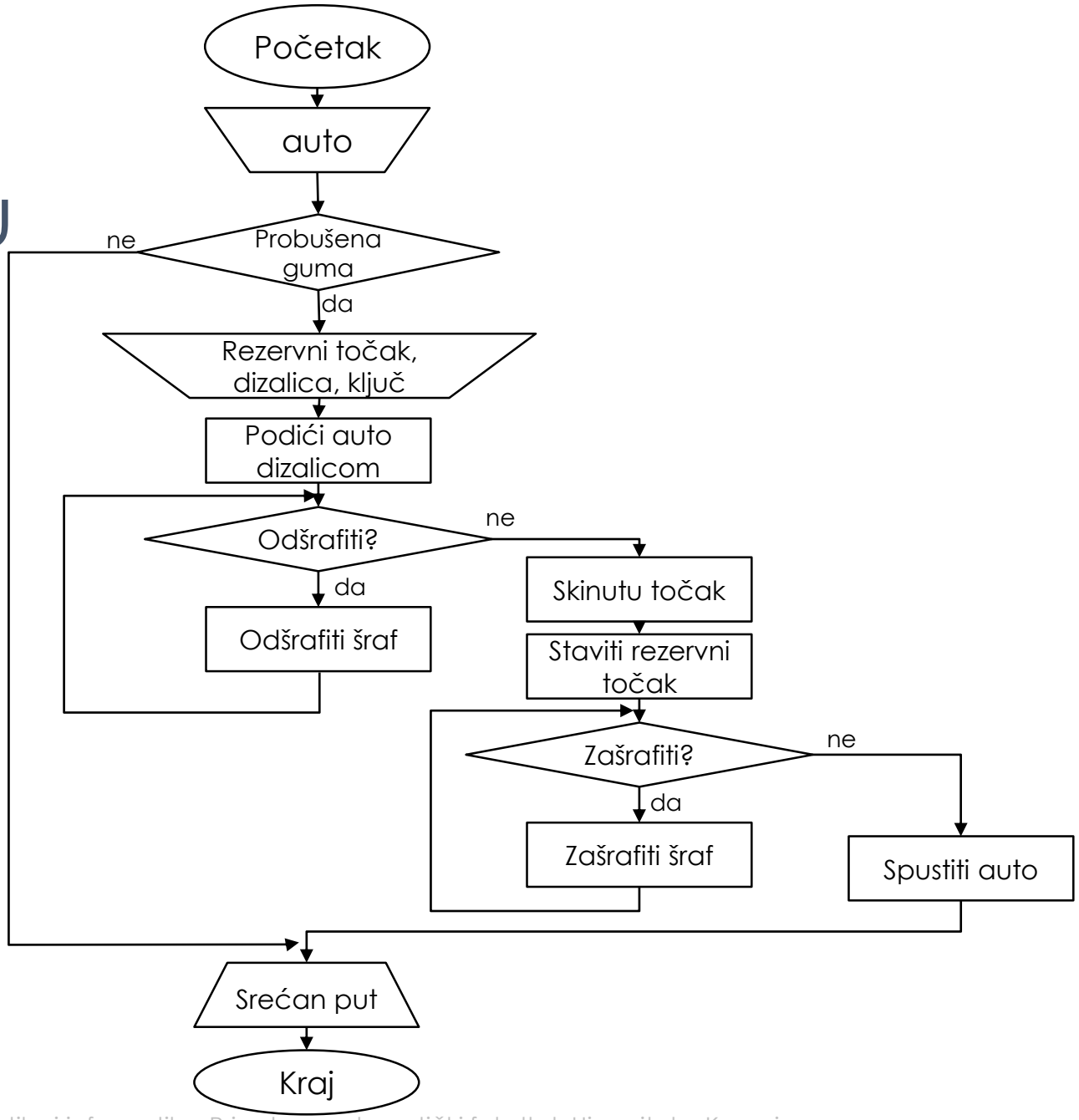
- Razgranata struktura



- Ciklička struktura



Algoritmi: probušena guma na autu



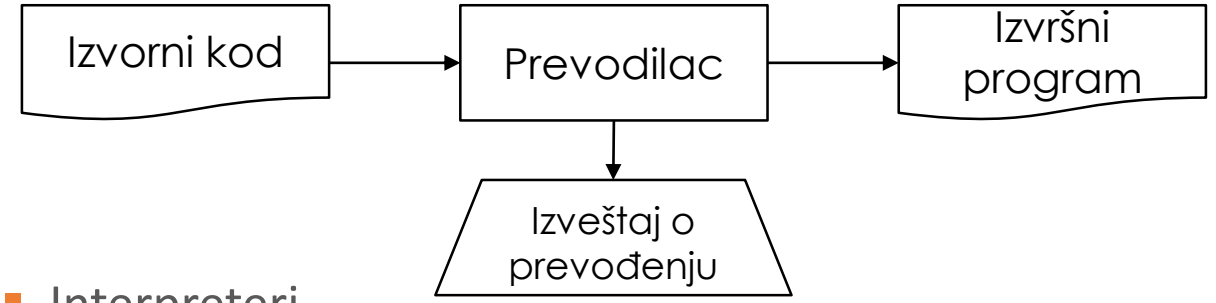
Vrste programskih jezika

- Algoritmi se u računaru realizuju odgovarajćim programom
- Program može biti realizovan korišćenjem različitih simbola (jezika), kojima se reprezentuju pojedine operacije podržane od strane računarskog sistema

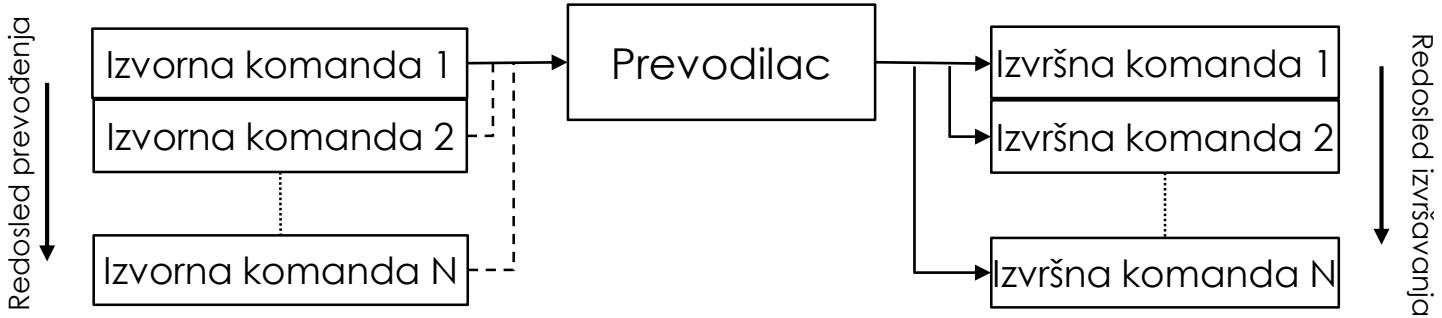
- Prema stepenu bliskoski sa arhitekturom računara
  - Jezici niskog nivoa (mašinski jezik, Assembler)
  - Jezici visokog nivoa (FORTRAN, Pascal, C, C++, C#, Java...)

Vrste programskih jezika

- Prema načinu prevođenja
  - Kompajleri



  - Interpreteri



Vrste programskih jezika

- Viši programski jezici dele se prema načinu kako se program formira:
  - Imperativni (opisuju kako se nešto izračunava)
  - Deklarativni (opisuju šta treba izračunati) – Prolog i SQL
- Program prilikom izvršavanja prolazi kroz određena stanja koja su definisana promenljivim u memoriji. Programer u imperativnom jeziku određuje način na koji pojedine promenljive menjaju svoja stanja, dok se kod deklarativnih jezika ovim pitanjem bave prevodilac ili interpreter, na osnovu opisa željenog rezultata.
- Imperativni jezici se razlikuju prema organizaciji izvornog koda:
  - Proceduralni – Basic, Fortran, Pascal, C
  - Objektno orijentisani – C++, Java, C#
- Funkcionalni pristup programiranju je bliži deklarativnom, gradi se od funkcija - Haskel

Python

Python

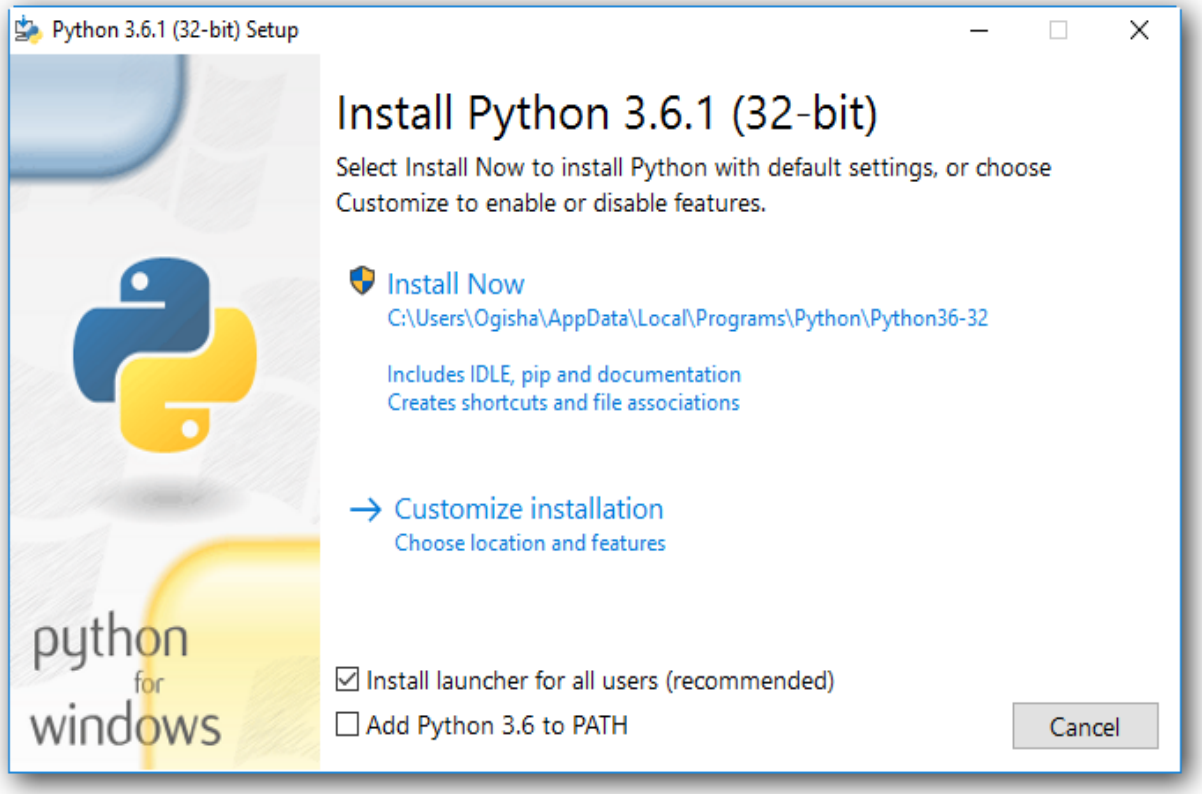
- Savremeni jezici primenjuju različite principe koji omogućavaju pisanje i organizaciju programa na različite načine.
- Python omogućava:
  - Proceduralno
  - Objektno
  - Kombinacija proceduralnog i objektnog
- Python podržava i pojedine elemente funkcionalnog programiranja

Zašto Python?

- Osmislio ga je holandiski programer Guido van Rossum krajem osamdesetih godina prošlog veka
- Python je interpretirani, objektno-orijentisani jezik visokog nivoa, namenjen za pravljenje svih vrsta aplikacija – od inženjerskih i naučnih, do poslovnih i web primena
- Pravila jezika su jednostavna, a izvorni kod je čitak i često dosta kraći negu u slučaju Java ili C/C++ ekvivalenata.
- Dostupan je za različite operativne sisteme: Windows, Linux, Mac
- Python stavlja na raspolaganje kvalitetne standardne biblioteke, koji nude gotova rešenja za mnoge probleme.

Python

- Na adresi https://www.python.org/downloads/ možete preuzeti instalacioni fajl programskog jezika Python za odgovarajući operativni sistem. Trenutno aktuelna verzija je 3.7.2 pa će postupak instalacije biti prikazan na osnovu nje. Pokrenuti instalacioni fajl, pojaviće se prozor sa slike:

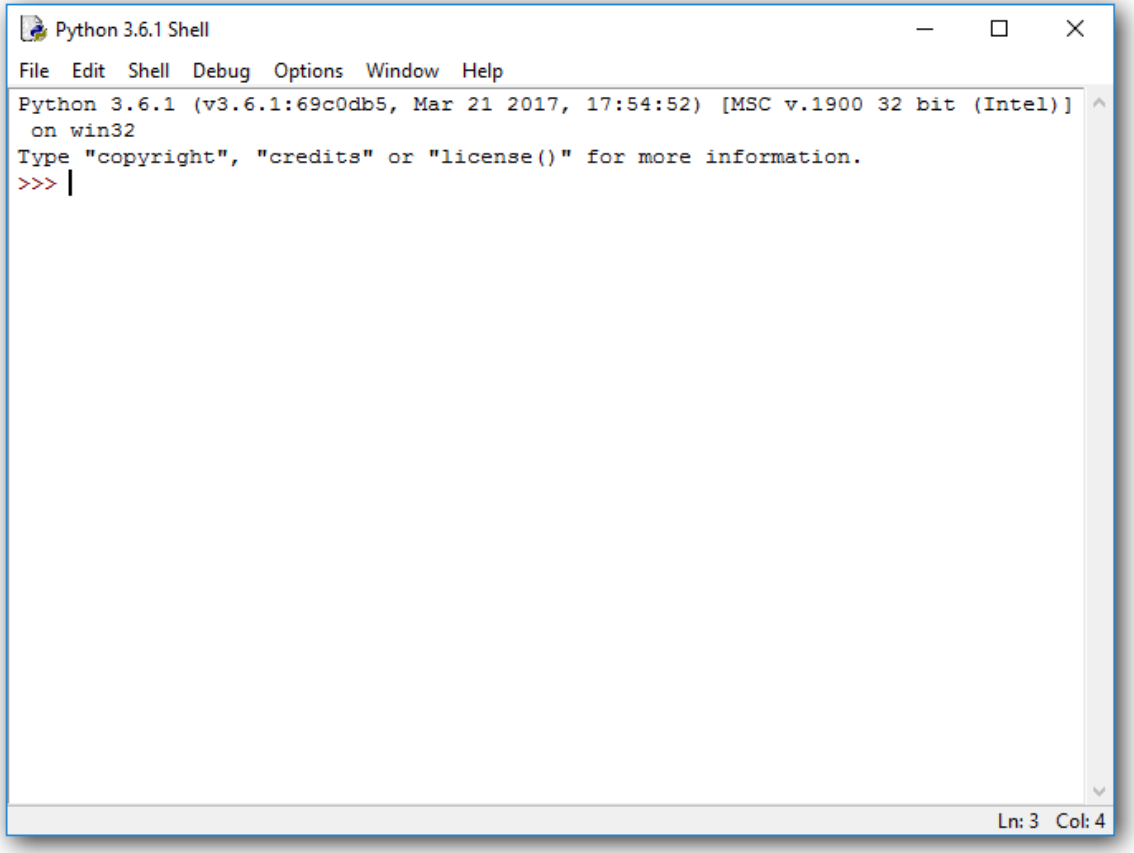


Python

- Programi se pišu uz pomoć odgovarajućeg softverskog paketa pod imenom `razvojno okruženje`
- Razvojno okruženje čini skup softverskih alata pomoću kojih se kreira program. U obavezne alate spadaju editor, debager, interpreter ili prevodilac i sistem za pomoć.
- Editor – za unos teksta izvornog koda
- Debager – testira rad programa u toku izvršavanja
- Sistem za pomoć – omogućava pregled dokumenata koja pojašnjavaju korišćenje

Python

- Uz osnovnu instalaciju dobija se besplatno razvojno okruženje IDLE, koje uključuje iteraktivni interpreter, editor teksta i alate za testiranje koda
- Nakon pokretanja, otvara se sledeći prozor:



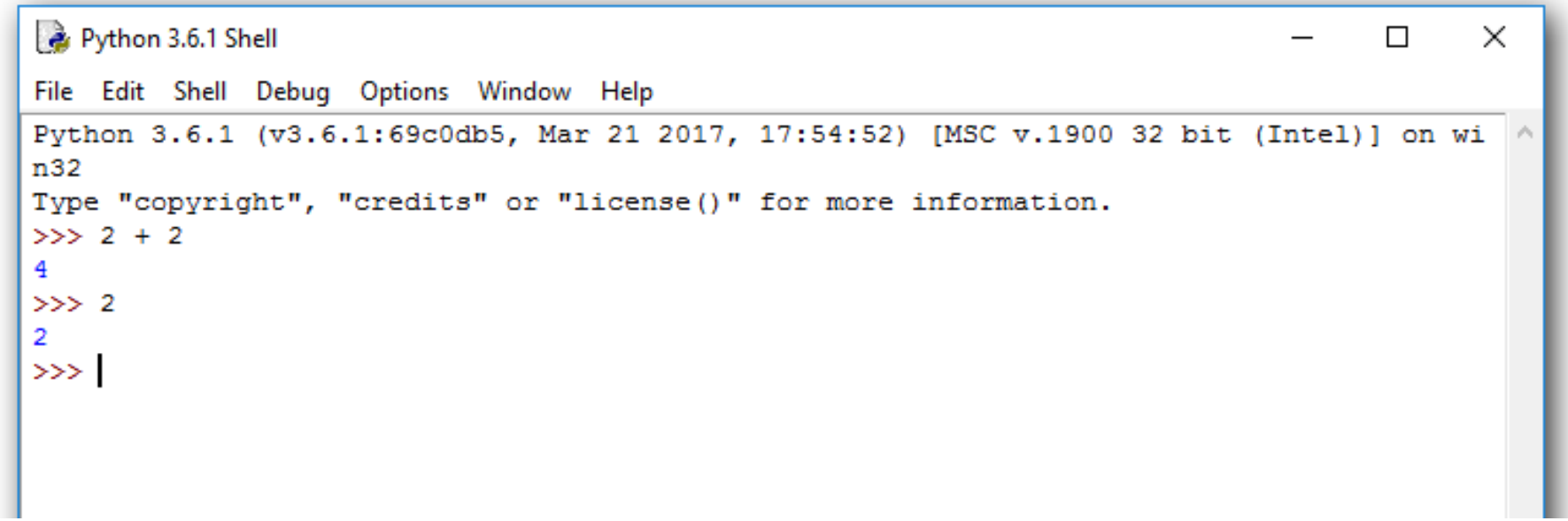
Python

- Simbol `>>>` označava tekuću liniju za unos komande koja se, po pritisku na taster `<Enter>`, izvršava od strane interpretera, a njen rezultat ispisuje u sledećem redu.

```
>>> 2 + 3
5
>>> 2 - 3
-1
>>> 2 * 3
6
>>> 9 / 4
2.25
```

kao kalkulator

Python



- U slučaju da smo napravili grešku u kucanju, ispisaće nam se poruka o grešci:

```
>>> 6 +
SyntaxError: invalid syntax
>>>
```

- Kao što se može videti, pogrešan deo koda je automatski uklonjen i možemo nastaviti sa radom.

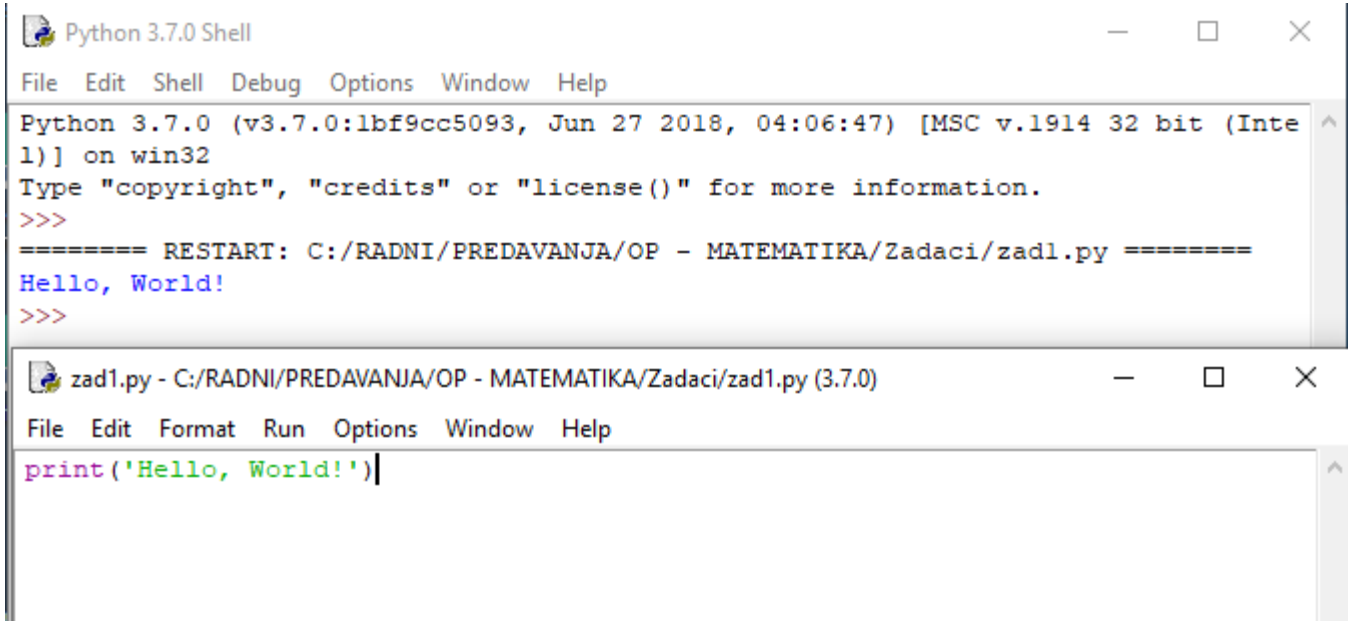
Python

- Rad u interaktivnom okruženju nije pogodan za izvršavanje složenijeg postupka jer se naredbe koje ga realizuju, moraju zapamtiti i ponoviti uvek u istom redosledu.
- Zato se naredbe grupišu u program koji je zapisan u datoteci na disku.
- Program se može pokrenuti, kako iz IDLE-a tako i van njega, na za to predviđene načine.
    - Programski kod se zapisuje u fajl – skript
    - Ekstenzija fajla je `.py`
    - Interpreter izvršava kod

Prvi program

- Testiranje rada programa podrazumeva da se, uz pomoć editora u IDLE-u, prvo formira programska datoteka.
- Editor se pokreće iz menija `File`, u okviru osnovnog prozora razvojnog okruženja, izborom opcije `New File` za novi, odnosno `Open`..., za otvaranje postojećeg programa

Po snimanju na disk, u datoteku sa ekstenzijom .py (u primeru zad1.py), program se pokrece izborom opcije `Run Module`, iz menija `Run`.



Promenljive

- Ime promenljive sme sadržati samo
  - Brojeve
  - Velika i mala slova
  - Donju crticu _
- Ne sme biti ključna reč
- Ime ne sme početi brojem
- Python razlikuje velika i mala slova

and def exec if not return
assert del finally import or try
break elif for in pass while
class else from is print yield
continue except global lambda raise

Neka od dozvoljenih imena:

```
jabuka, slova123, A4, ime_psa
```

Neka od nedozvoljenih imena:

```
ime psa, 123slova, jabuka#
```

Vrste grešaka

- Syntax Error

```
>>> 1a = 100
SyntaxError: invalid syntax
>>> a = 3 * 2 +
SyntaxError: invalid syntax
>>>
```

- Runtime Error – greška se javlja tek kada se pokrene program
- Semantic Error

```
>>> x = 5 + "Praktikum"
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    x = 5 + "Praktikum"
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Promenljive

```
>>> a = 1
>>> b = 2
>>> c = 2.5
>>> a
1
>>> c
2.5
>>> x
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    x
NameError: name 'x' is not defined
>>>
```

Promenljive

```
>>> poruka = "Programiranje je lepo"
>>> pi = 3.14
>>> n = 5
>>> poruka
'Programiranje je lepo'
>>> pi
3.14
>>> n
5
```

Tipovi i vrednosti

- Dodelom vrednosti se određuje tip neke promenljive. Ukoliko nismo sigurni kog je tipa neki podatak, interpreter nam to može reći

```
>>> type(poruka)
<class 'str'>
>>> type(pi)
<class 'float'>
>>> type(n)
<class 'int'>
```

tip	oznaka	domen vrednosti	primeri vrednosti
celobrojni	int	celi brojevi	0, 1, -1243
realni	float	realni brojevi	1.25, .02, -145.25
komleksi	comlex	kompleksi brojevi	-3j, -0.5 – 4.25j
logički	bool	{ False, True }	False, True
tekstualni	str	niz znakova	‘Ucim programiranje’
torka	tuple	nepromenljiv niz proizvoljnih objekata	(), (1.25, True, ‘abc’)
lista	list	promenljiv niz proizvoljnih objekata	[],[1, 2], [‘A’, 1, 1 + 2j]
skup	set	neuređen skup proizvoljnih nepromenljivih objekata. Duplikati nisu dozvoljeni	{},{1, 2}, {‘A’, 1, {1, 2}}
rečnik	dict	parovi ključ-vrednost	{11: ‘Beograd’, 21:’Novi Sad’, 34:’Kragujevac’}
nedefinisano	NoneType	{ None }	None

Aritmetičke operacije

- Python kao kalkulator
- Prioritet operacija je uobičajen

Znak	Operacija
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje
%	Ostatak pri deljenju
//	Celobrojno deljenje
**	Stepenovanje

```
>>> 2 + 3
5
>>> 2 - 3
-1
>>> 2 * 3
6
>>> 9 / 4
2.25
```

```
>>> 2 + 3 * 6
20
>>> 2 + 4 * 9 - 3
35
>>> 27 // 10
2
>>> 27 / 10
2.7
>>> 27 % 10
7
```

Standardni ulaz/izlaz

- Upisivanje tekstualne ili brojčane vrednosti je moguće pomoću naredbe `input()`. Sama naredba ništa ne znači ukoliko nije pridružena nekoj promenljivoj.
- Tip prihvaćenih podataka je uvek `string`, ukoliko se ne zahteva drugačije.

```
a=tip_podatka(input("Unesi vrednost")
```

naziv promenljive

tip promenljive: int, float, bool ili string

naredba za unos

tekstualni deo kao poruka, ne mora da postoji

- Funkcija za ispis je `print()`, na pr:

```
print("Ovo je tekst ","a ovo je broj ",30)
```

Funkcija `print`

```
print ('Zdravo svete')
print (34)
print (34*34)
print (12+14/(6+1))
recenica = 'Sunce je zuto'
print (recenica)
rec1 = 'Zuto'
rec2 = 'sunce'
print (rec1, rec2)
ime = 'Milos'
print (ime, 'ima', 65, 'kilograma')
print ("Kvadrat broja %d je %d" %(4, 16))
print ("%s je danas dobio ocenu %d iz %s" %("Petar", 5, "matematike"))
```

```
Zdravo svete
34
1156
14.0
Sunce je zuto
Zuto sunce
Milos ima 65 kilograma
Kvadrat broja 4 je 16
Petar je danas dobio ocenu 5 iz matematike
```

Ugrađene funkcije

- `min, max, abs`

```
print(min(2.3,5.8),max(8,12))
broj = -3
print(max (abs (broj), min (2, 4)))
```

- Matematičke funkcije

```
import math as m
print(m.factorial(4))
print(m.sqrt(2))
```

`pow, sqrt, floor, trunc, exp, log, sin, cos,...`

Celobrojno deljenje

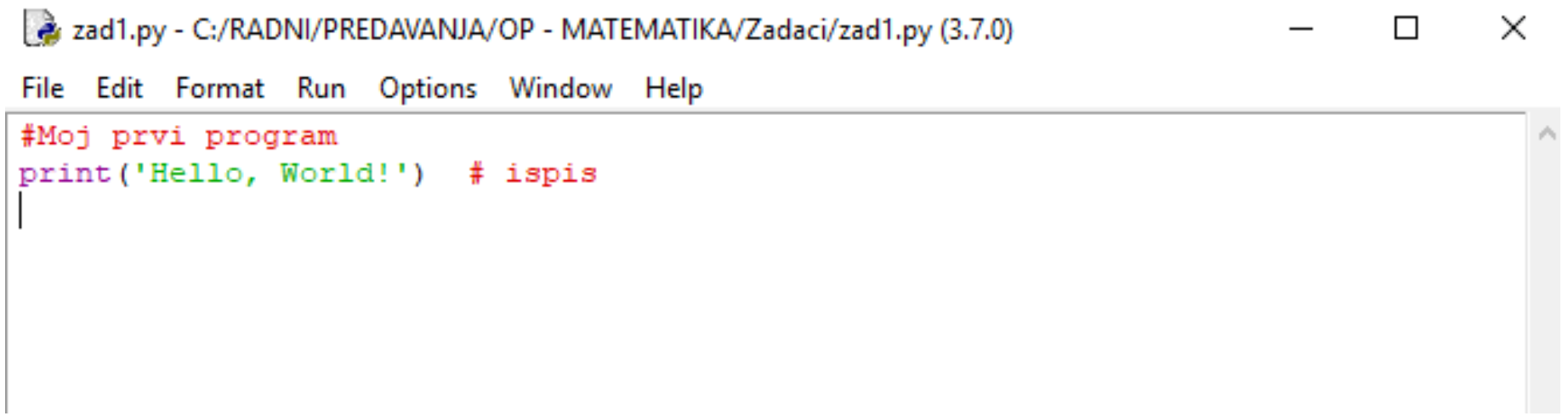
- Uneti brojilac **a** i imenilac **b** razlomka $\frac{a}{b}$ i prevedi ga u mešoviti broj.

```
a = int(input("Brojilac "))
b = int(input("Imenilac "))
mesoviti_ceo_deo = a // b
mesoviti_brojilac = a % b
mesoviti_imenilac = b
print(mesoviti_ceo_deo, "celih i", mesoviti_brojilac, "/",
mesoviti_imenilac)
```

```
Brojilac 37
Imenilac 12
3 celih i 1 / 12
>>>
```

Komentari

- U skipt fajl se mogu dodati komentari
- Komentar počinje simbolom #
- Python ne analizira sadržaj od # do kraja reda



Višestruka dodela vrednosti

```
>>> a, b, poruka = 3, 4, 'jednostavno zar ne!'
>>> print(a, b, poruka)
3 4 jednostavno zar ne!
```

- Imenovanje promenljivih pri rešavanju problema

razumljivo	nerazumljivo
pi = 3.14	a = 3.14
O = 2 * r * pi	b = 2 * x * a
P = r**2 * pi	c = x**2 * a

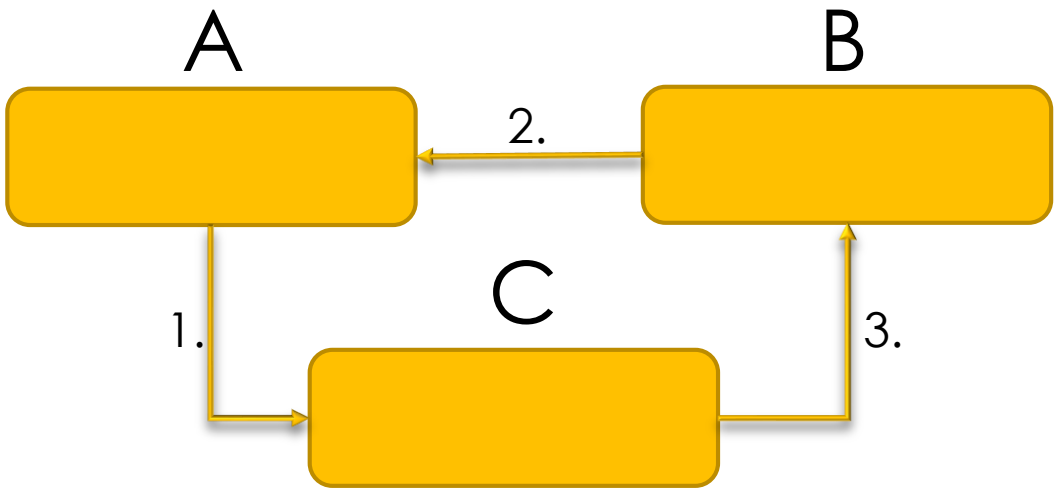
Razmena vrednosti

- Učitati dve vrednosti sa tastature i imenovati ih uz pomoć promenljivih **a** i **b**. Razmeniti vrednosti na koje promenljive ukazuju i potom ih prikazati na ekranu.

zad1.py - C:/RADNI/PREDAVANJA/OP - MATEMATIKA/Zadaci/zad1.py (3.7.0)

File Edit Format Run Options Window Help

```
# razmenjuje učitane vrednosti sa tastature
a = input('Prva vrednost = ')
b = input('Druga vrednost = ')
print('pre razmene a =', a, 'b =', b)
# razmena
c = a # 1
a = b # 2
b = c # 3
print('posle razmene a =', a, 'b =', b)
```



Razmena vrednosti

- Razmena vrednosti može se realizovati preko naredbe višestruke dodele vrednosti, što predstavlja optimalno rešenje u stilu jezika Pyhton

zad1.py - C:/RADNI/PREDAVANJA/OP - MATEMATIKA/Zadaci/zad1.py (3.7.0)

File Edit Format Run Options Window Help

```
# razmenjuje učitane vrednosti sa tastature
a = input('Prva vrednost = ')
b = input('Druga vrednost = ')
print('pre razmene a =', a, 'b =', b)
# razmena
a, b = b, a # razmena
print('posle razmene a =', a, 'b =', b)
```

Primer

- Šta će biti vrednost promenljivih A i B nakon izvršenja sledećih naredbi:

```
A=4
B=7
A=2*A-B
B=7*A-B
A=A-A
print(A,B)
```

Funkcije

Potprogram

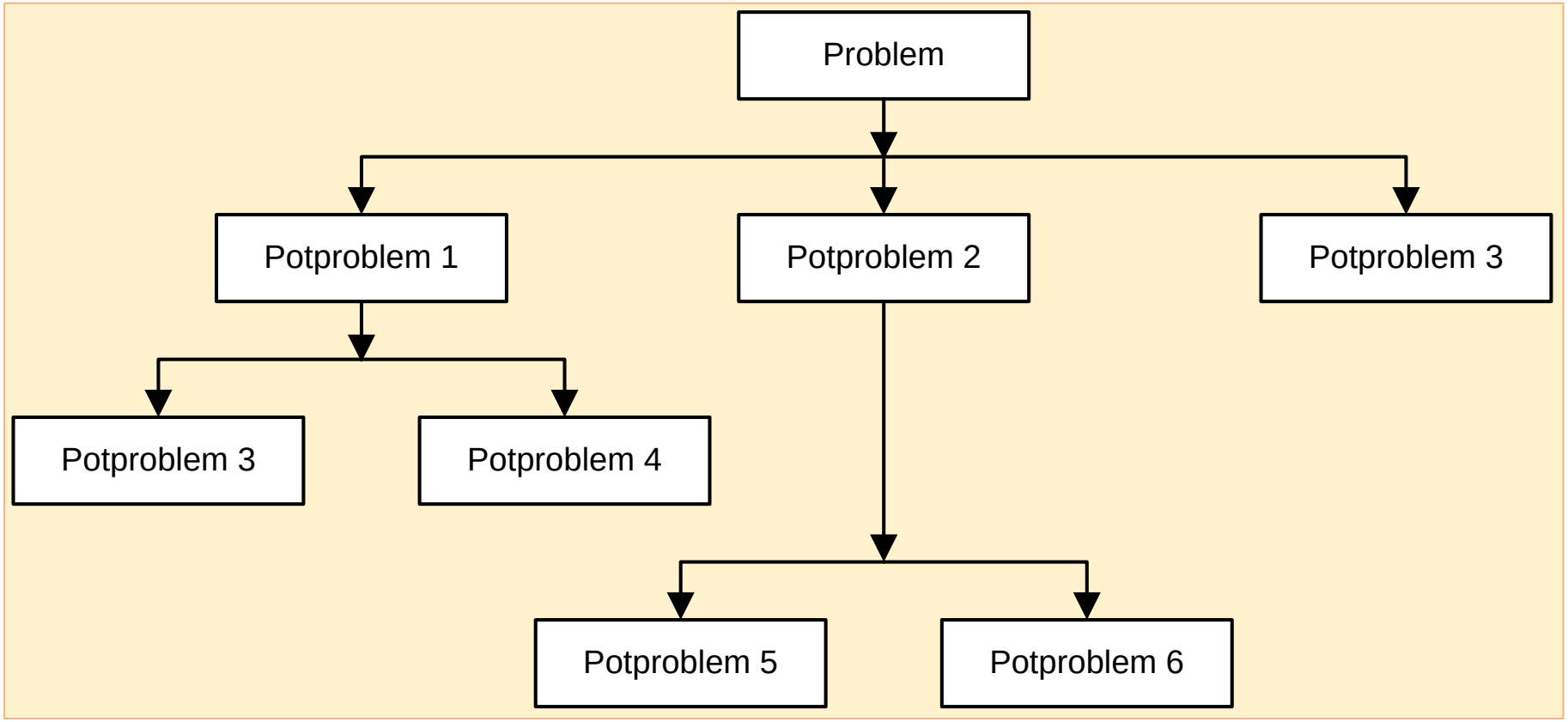
- Složeniji programi se mogu podeliti na odgovarajuće logičke i funkcionalne celine, takozvane `potprograme`.
- Potprogram omogućava organizovanje određenog broja instrukcija u celine koje obavljaju određene zadatak.
- Potprogram preuzima ulazne veličine i vrši njihovu obradu u cilju dobijanja određenih rezultata
- Potprogram možemo pozivati više puta u toku izvršavanja programa.

Prednosti korišćenja potprograma

- Omogućava koncentrisanje na izvršavanje samo određenog zadatka
- Različiti članovi razvojnog tima mogu razvijati različite potprograme koji se kasnije sklapaju u jednu celinu
- Omogućava izvršavanje istog zadatka na više mesta u programu jednostavnim pozivanjem odgovarajućeg potprograma

Programiranje “odozgo na dole”

- Funkcije i procedure omogućavaju realizaciju programiranja "odozgo na dole". Ovaj metod podrazumeva razbijanje problema na potprobleme



Funkcije

- Funkcija je samostalan deo programa koji obavlja određeni zadatak
- Preko svog naziva i liste parametara, delu programa iz koga je pozvana vraća odgovarajuće rezultate
- Svaka funkcija ima jedinstveni naziv preko koga se može pozvati proizvoljan broj puta iz bilo kog dela programa u cilju izvršenja tog zadatka.

Funkcije

- Python pruža mogućnost definisanja korisničkih funkcija

```
def ime_funkcije(parametri_funkcije) :
    telo funkcije
```

- Funkcija mora biti definisana pre prvog korišćenja(poziva)

```
def stampaj_izjavu():
    print ("Moje ime je Ana i OK sam.")
    print ("Spavam celu noc i radim ceo dan.")

stampaj_izjavu()

print (stampaj_izjavu)
print (type(stampaj_izjavu))
```

```
Moje ime je Ana i OK sam.
Spavam celu noc i radim ceo dan.
<function stampaj_izjavu at 0x0341FE40>
<class 'function'>.
```

Poziv funkcije iz funkcije

- U telu funkcije može se naći poziv ranije definisane funkcije

```
def stampaj_opet():
    stampaj_izjavu()
    stampaj_izjavu()


stampaj_opet()
```

```
Moje ime je Ana i OK sam.
Spavam celu noc i radim ceo dan.
Moje ime je Ana i OK sam.
Spavam celu noc i radim ceo dan.
```

Funkcije sa parametrima

- Funkcija može biti definisana sa jednim ili više parametara.
- Poziv funkcije mora odgovarati definiciji funkcije po broju i redosledu parametara.
- Parametri navedeni u opisu funkcije nazivaju se formalni parametri.
- Parametri navedeni pri pozivu funkcije nazivaju se stvarni parametri.
- Formalni i stvarni parametri se moraju podudarati po broju i po tipu.
- Pri izvršavanju funkcije, formalnim parametrima se dodeljuju vrednosti stvarnih parametara.

Funkcije sa parametrima

```
def stampaj_2puta(param):
    print(param)
    print(param)

stampaj_2puta('Spam')
stampaj_2puta(math.pi)
stampaj_2puta('Spam'*4)
stampaj_2puta(17)
```

```
Spam
Spam
3.141592653589793
3.141592653589793
SpamSpamSpamSpam
SpamSpamSpamSpam
17
17
```

Lokalne promenljive funkcije

- Promenljive definisane i korišćene unutar funkcije, nisu vidljive van nje

```
def saberi(a,b):
    c = a + b
    print (c)

saberi(3,5)
print(c)
```

```
8
NameError: name 'c' is not defined
```

Lokalne promenljive funkcije

- Promenljive definisane I korišćene unutar funkcije, nisu vidljive van nje

```
c = 10
def saberi(a,b):
    c = a + b
    print (c)

saberi(3,5)                8
print(c)                   10
```

Vraćanje vrednosti iz funkcije

- Funkcija kao rezultat može da vrati jednu vrednost

```
a = saberi (12,26)          38
print (a)                   None

def saberi2 (a,b):
    c = a + b
    return c

saberi2 (3,5)
a = saberi2 (12,26)
print (a)                   38
```

Funkcije sa više rezultata

- U nekim situacijama funkcija treba da vrati više vrednosti. Na primer, želimo da pretvorimo centimetre u metre i preostale centimetre.

```
def cm_u_mcm(cm):
    return (cm // 100, cm % 100)

(m, cm) = cm_u_mcm(178)
print(178, "cm", "=", m, "m", "i", cm, "cm")
```

Funkcije sa više rezultata

- Napiši funkciju koja na osnovu broja sekundi proteklih od prethodne ponoći izračuna trenutno vreme u satima, minutima i sekundama, vodeći računa da broj sati bude između 0 i 23.

```
def sek_u_satminsek(s):
    sek = (s // (60*60)) % 24
    min = (s // 1) % 60
    sat = (s // 60) % 60
    return  (sat,min,sek)

(sat, min, sek) = sek_u_satminsek(1000)
print(sat, ":", min, ":", sek)
(sat,min,sek) = sek_u_satminsek(7200)
print(sat, ":", min, ":", sek)
```

Grananje

Logičke operacije

Znak	Relacija
And	Logičko „i“
Or	Logičko „ili“
Not	Logičko „ne“

```
x     | y     | x and y | x or y
True  | True  | True    | True
True  | False | False   | True
False | True  | False   | True
False | False | False   | False
```

- Tabela istinitosti logičkih operacija and i or

```
print(' x     | y     | x and y | x or y')
print('True  | True  |', True and True, '   |', True or True)
print('True  | False |', True and False, '  |', True or False)
print('False | True  |', False and True, '  |', False or True)
print('False | False |', False and False, '  |', False or False)
```

Operatori

Znak	Relacija
>	Veće
<	Manje
>=	Veće i jednako
<=	Manje i jednako
==	Jednako
!=	Različito

- Ispisati veći od uneta dva realna broja

```
x = float(input('x= '))
y = float(input('y= '))
print('veći broj je', x * (y < x) + y * (x <= y) )
```

Grananje

- Neke naredbe se izvršavaju samo ako je neki uslov ispunjen. Da bi se opisalo uslovno izvršavanje nekih naredbi koristi se naredba `if`

```
if uslov:
    naredba_1
    ...
    naredba_k
```

```
if uslov:
    naredba_1
    ...
    naredba_m
else:
    naredba_1
    ...
    naredba_n
```

```
if uslov1:
    naredba_1
    ...
    naredba_m
elif uslov2:
    naredba_1
    ...
    naredba_n
else:
    naredba_1
    ...
    naredba_k
```

- Zagrade u uslovu nisu OBAVEZNE.
- Dvotačka je obavezna nakon navođenja uslova
- Naredba uslova će vratiti tačno/netačno
- Ako je tačan uslov koji se zadaje izvršiće naredba_1 …

Grananje

- Šta je rezultat datog koda?

```
a=0
if a:
    print(a)
```

```
a=0
if a==0:
    print(a)
```

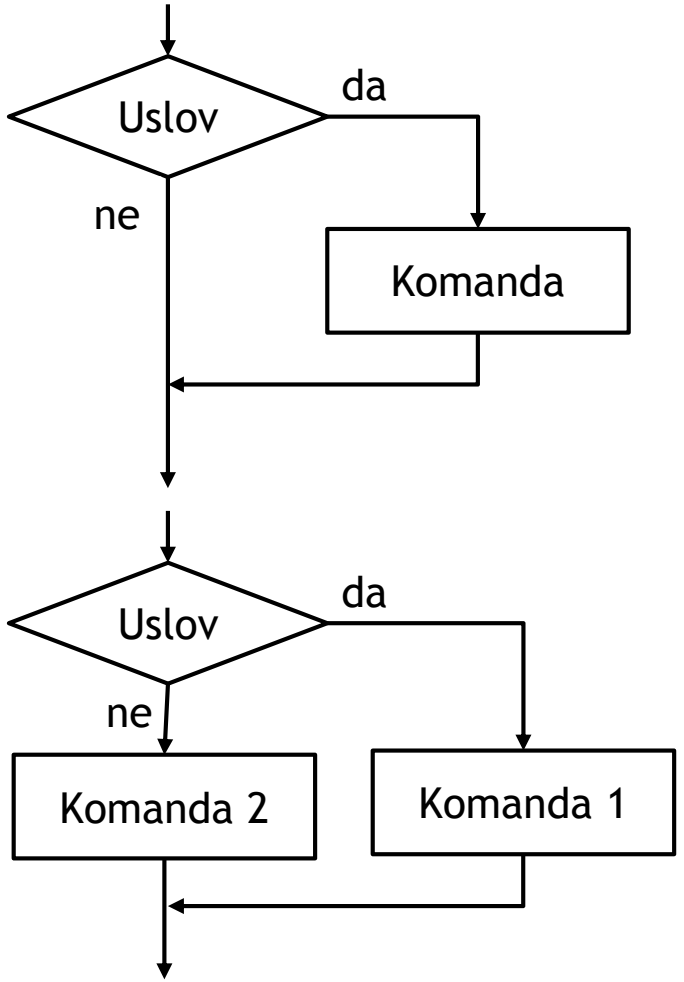
Uslovno i alternativno izvršavanje

- Za uneti broj ispisati da li je manji od 10

```
a = int(input("Unesi broj "))

if a < 10:
    print ('Broj je manji od 10')
```

- Za uneti broj ispisati da li je paran ili neparan

```
if a%2 == 0:
    print ('Broj je paran')
else:
    print ('Broj je neparan')
```



Blok komandi

- Za unete dužine stranica pravougaonika ili kvadrata (ukoliko su unete vrednosti jednake) izračunati površinu.

```
a=int(input("Unesite stranicu a: "))
b=int(input("Unesite stranicu b: "))

if a==b:
    p=a**2
    print("Povrsina kvadrata iznosi: ", p)
else:
    p=a*b
    print("Povrsina pravougaonika iznosi: ", p)
```

Više od jedne naredbe

Primer

- Napisati program koji za uneto *x* izračunava vrednosti funkcija *f(x)* i *g(x)*, koje su date formulama

$$f(x) = \begin{cases} -1, & \text{ako je } x < 0 \\ 2x + 3, & \text{ako je } x \geq 0 \end{cases}$$

i

$$g(x) = \begin{cases} \ln(-x), & \text{ako je } x < 0 \\ e^x, & \text{ako je } x \geq 0 \end{cases}$$

Primer

```
import math
x=float(input('Unesi x '))
if x<0:
    f=-1.0
    g=math.log(-x)
else:
    f=2*x+3
    g=math.exp(x)
print(f,g)
```

Primer

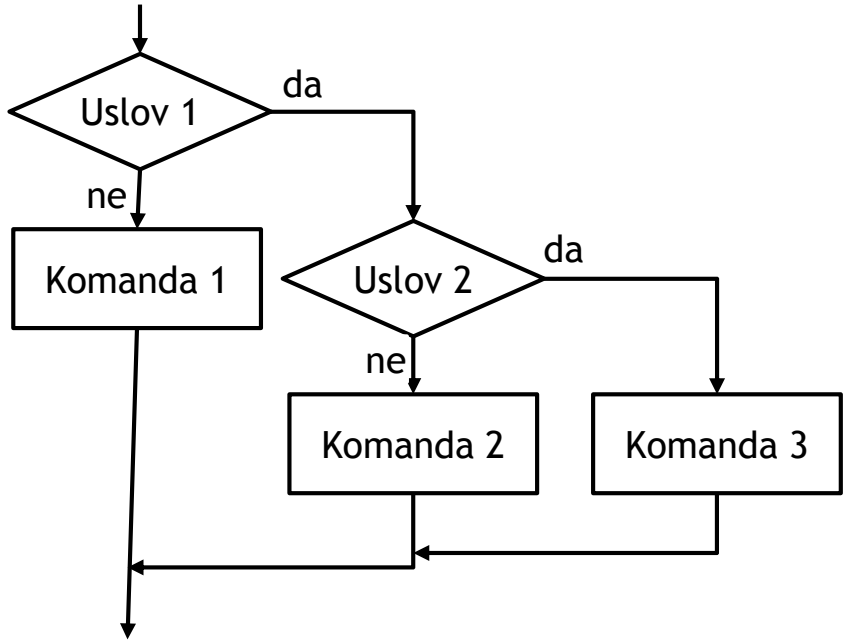
- Ako se od odsečaka sa dužinama x, y, z može konstruisati trougao, izračunati njegovu površinu po formuli:

$$P=\sqrt{s * (s - x) * (s - y) * (s - z)}$$

Lančani uslovi

- Odrediti odnos dva uneta broja

```
x = int(input("Unesi broj x "))
y = int(input("Unesi broj y "))

if x < y:
    print ('x je manje od y')
elif x > y:
    print ('x je vece od y')
else:
    print ('x i y su jednaki')
```



Primer

- Unetu operaciju primeniti na zadate vrednosti promenljivih **a** i **b**

```
a = 25
b = 5
c = input('uneti operaciju: ')
if (c == '+'):
    print (a + b)
elif (c == '*'):
    print (a * b)
elif (c == '-'):
    print (a - b)
elif (c == '/'):
    print (a / b)
```

Ugnježdeni uslovi

- Odrediti odnos dva uneta broja

```
x = int(input("Unesi broj x "))
y = int(input("Unesi broj y "))

if x == y:
    print ('x i y su jednaki')
else:
    if x < y:
        print ('x je manje od y')
    else:
        print ('x je vece od y')
```

Primer

- Napisati program koji na osnovu broja bodova osvojenih na ispitu, određuje ocenu prema sledećoj tabeli

Broj bodova	Ocena
51 - 60	6
61 - 70	7
71 - 80	8
81 - 90	9
91 - 100	10

Primer

```
bodovi = float(input('Unesi broj bodova '))
if bodovi >= 51:
    if bodovi >= 61:
        if bodovi >= 71:
            if bodovi >= 81:
                if bodovi >= 91:
                    ocena = 10
                else:
                    ocena = 9
            else:
                ocena = 8
        else:
            ocena = 7
    else:
        ocena = 6
else:
    ocena = 5
print(ocena)
```

ILI

```
bodovi = float(input(Bodovi '))
if bodovi >= 91:
    ocena = 10
else:
    if bodovi >= 81:
        ocena = 9
    else:
        if bodovi >= 71:
            ocena = 8
        else:
            if bodovi >= 61:
                ocena = 7
            else:
                if bodovi >= 51:
                    ocena = 6
                else:
                    ocena = 5
print(ocena)
```

Vezivanje više uslova logičkim operatorima

Kako bi napisali date uslove:

- `1 < x < 2`
- `x = y = z`
- `X ne pripada interval –a do a`

Vezivanje više uslova logičkim operatorima

Stanari jedne zgrade treba da renoviraju jedan sprat i radnicima su dali svoje uslove kada smeju da buše i grade:

- rekli su im da mogu da rade od 9 sati izjutra ali
- da moraju da naprave pauzu u vreme odmora od 14-17
- ne smeju posle 22 časa da buše

```
sati = int(input('Koliko je sati: '))
if ((sati>=9 and sati<13) or (sati>=17 and sati <22)):
    print("moze da se radi")
else:
    print("ne moze da se radi")
```

Rekurzija

Rekurzija

- U matematici ili informatici označava postupak ili funkciju koji u svojoj definiciji koriste sami sebe
- Rekurzivne definicije su prisutne i u matematici:

Definicija prirodnih brojeva:

- 1 je prirodan broj
- Ako je n prirodan broj, onda je to i n+1

Fibonačijev niz:

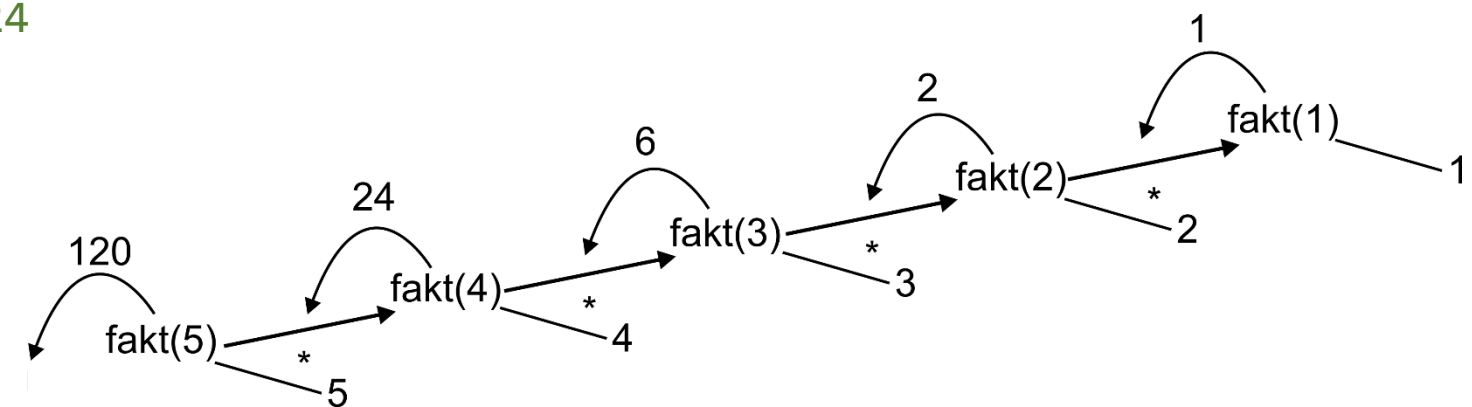
- 1. član niza je 0
- 2. član niza je 1
- Svaki n-ti član niza (n>2) je suma prethodna dva ($a_n = a_{n-1} + a_{n-2}$)

Rekurzija - Faktorijel

```
def fakt(n):
    if n<2:
        return 1
    else:
        return n*fakt(n-1)
print(fakt(5))
```

- $0! = 1 \quad n! = n \cdot (n-1)!$

```
fakt(5) = 5 · fakt(4)
        = 5 · (4 · fakt(3))             // (jer je fakt(4) = 4 · fakt(3))
        = 5 · (4 · (3 · fakt(2)))       // (jer je fakt(3) = 3 · fakt(2))
        = 5 · (4 · (3 · (2 · fakt(1)))) // (jer je fakt(2) = 2 · fakt(1))
        = 5 · (4 · (3 · (2 · 1)))       // (jer je fakt(1) = 1)

        = 5 · (4 · (3 · (2 · 1)))       // množenje se sada vrši redom
        = 5 · (4 · (3 · 2))             // kojim se f-je završavaju, tj. unazad
        = 5 · (4 · 6)
        = 5 · 24
        = 120
```



Rekurzija

- Svako rešenje rekurizvnog problema se sastoji od dva glavna dela:
  - **Bazni slučaj(evi)** - rešenja trivijalnih instanci problema
  - **Rekurzivni slučajevi** - zavisnost rešenja problema od rešenja manjih instanci

Definicija Rekurzija predstavlja metod u kome rešenje polaznog problema nalazimo koristeći rešenja podproblema iste strukture - jednostavnije instance istog problema.

Rekurzija – često postavljana pitanja

- Funckija nije završila sa radom i ja ne mogu ponovo da je pozovem?
- Ukoliko funkcija pozove samu sebe, dobiću beskonačnu petlju?
- Kako da odredim koji su mi bazni slučajevi potrebni ili mi možda neki nedostaje?

Rekurzija - Fibonači

```
def fibonacci (n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

print(fibonacci(5))
```

```
5
```

Verižni razlomci

- Verižni razlomak je izraz oblika:

$$\alpha = a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{a_3 + \ddots}}}$$

gde su a_i, b_i proizvoljni izrazi.

Jednostavni verižni razlomak je verižni razlomak kojem su svi b_i jednaki 1.

Verižni razlomci

- Jedan od verižnih razlomaka se vezuje za vrednost broja pi:

$$\frac{4}{\pi} = 1 + \cfrac{1}{3 + \cfrac{4}{5 + \cfrac{9}{7 + \cfrac{16}{9 + \cfrac{25}{11 + \cfrac{36}{13 + \cdots}}}}}}$$

- Uočavamo dva niza:
  - Sabirci: 1, 3, 5, 7, 9, 11,...
  - Deljenici: 1, 4, 9, 16, 25, 36,...
- Rekurzija se ne može širiti u beskonačnost

Verižni razlomci

```
import math
def verizni(n):
    if n>200:
        return 2*n-1
    else:
        return (2*n-1)+n*n/verizni(n+1)


print(verizni(1))
print(4/math.pi)
```

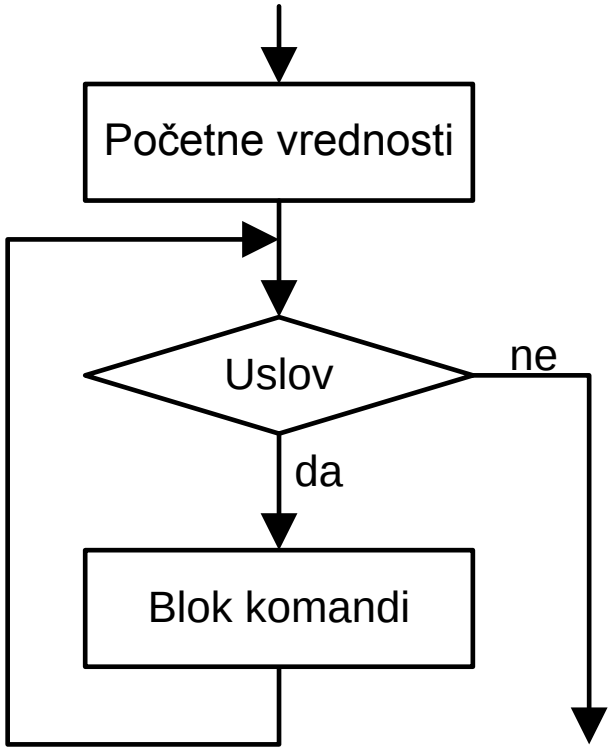
```
1.2732395447351628
1.2732395447351628
```

Ponavljanje

Ponavljanje naredbi - WHILE

- Python pruža mehanizam da se jedna ili više naredbi ponovi određen broj puta ili dok je neki uslov ispunjen

```
while (uslov):
    naredba 1
    naredba 2
```



- Naredbe se izvršavaju sve dok je uslov tačan.
- Uslov ne mora biti u zagradi.

Ponavljanje naredbi - WHILE

- Poređenje - želimo 5 puta da štampamo `'Zdravo!'`

```
br = 0
if br < 5:
    print('Zdravo!')
    br = br + 1
```

```
Zdravo!
```

```
br = 0
while br < 5:
    print('Zdravo!')
    br = br + 1
```

```
Zdravo!
Zdravo!
Zdravo!
Zdravo!
Zdravo!
```

Ponavljanje naredbi - WHILE

- Šta radi dati deo koda?

```
n=5
while n > 0:
    print(n)
    n = n-1                    # šta se dešava bez ovog reda?
print("Gotovo!")
```

Ponavljanje naredbi - WHILE

- Odrediti koren svih celih brojeva od 1 do n (n>1) i prikazati tabelarno.

```
import math

n=int(input())
br=1
while br <= n:
    print(br,'   ',math.sqrt(br))
    br=br+1
```

```
5
1     1.0
2     1.4142135623730951
3     1.7320508075688772
4     2.0
5     2.23606797749979
```

Ponavljanje naredbi - WHILE

- Izračunati zbir unetih brojeva. Brojeve unositi dok se ne unese 0.

```
x = int(input("Unesi broj: "))
s = 0
while x != 0:
    s = s + x
    x = int(input("Unesi broj: "))
print("Zbir unetih brojeva je ", s)
```

```
Unesi broj: 3
Unesi broj: 5
Unesi broj: 4
Unesi broj: 7
Unesi broj: 0
Zbir unetih brojeva je  19
```

Primer

- Koliko puta će se ponoviti ciklus i šta će biti vrednosti promenljivih **a**,**b** i **s** posle izvršenja ovog niza naredbi?

```
a = 1
b = 1
while a+b<8:
    a = a+1
    b = b+2
s = a+b
```

Primer

- Šta je rezultat sledećeg koda?

```
i = 1
while i <= 6:
    print(5*i, '\t', end='')
    i = i + 1
print()
```

Ponavljanje naredbi - FOR

- Na osnovu prethodnog, `while` predstavlja najopštiji način za realizovanje ponavljanja algoritamskih koraka.
- Može se koristiti kako za situacije kada broj ponavljanja nije poznat, tako i za zadatke u kojima se on unapred zna (brojačke petlje).
- Međutim, za brojačke petlje pogodnije je koristiti drugi mehanizam – naredbu `for`

```
for brojač in opseg :
    telo_petlje
```

Ponavljanje naredbi - FOR

```
for i in range(5):
    print("Zdravo")
```

```
Zdravo
Zdravo
Zdravo
Zdravo
Zdravo
```

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

Ponavljanje naredbi - FOR

- Funkcija `range` generiše sekvencu celih brojeva

Poziv funkcije	Generisana sekvenca
range(n)	za n > 0: 0, 1, 2, 3, … , n-1
range(start, stop)	za stop > start: start, start+1, start+2, … , stop-1
range(start, stop, korak)	za stop > start i korak > 0: start, start+korak, start+2*korak, …, n važi stop – korak <= n < stop
range(start, stop, korak)	za stop < start i korak < 0: start, start+korak, start+2*korak, …, n važi stop – korak >= n > stop

Ponavljanje naredbi - FOR

```
a = int(input("Unesi ceo broj "))
b = int(input("Unesi ceo broj "))
for i in range(a,b+1):
    print (i)
print()

for i in range(5,25,5):
    print (i)
print()

for i in range(25,4,-5):
    print (i)
```

```
Unesi ceo broj 3
Unesi ceo broj 9
3
4
5
6
7
8
9

5
10
15
20

25
20
15
10
5
```

Ugnježdene petlje

- Ispis tablice množenja

```
for x in range(1, 11):
    for y in range(1, 11):
        print(x*y)
```

```
for x in range(1, 11):
    for y in range(1, 11):
        print(x*y,' ',end='')
    print()
```

```
1  2  3  4  5  6  7  8  9  10
2  4  6  8  10  12  14  16  18  20
3  6  9  12  15  18  21  24  27  30
4  8  12  16  20  24  28  32  36  40
5  10  15  20  25  30  35  40  45  50
6  12  18  24  30  36  42  48  54  60
7  14  21  28  35  42  49  56  63  70
8  16  24  32  40  48  56  64  72  80
9  18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90  100
```

Komande BREAK i CONTINUE

- Za prekid izvršavanje petlje, bez obzira na ispunjenost uslova koristi se komanda BREAK

```
for i in range(1,10):
    if i == 3:
        break
    print (i)
```

```
1
2
```

- Komandom CONTINUE se prekida izvršavanje trenutne iteracije (preskače se deo koda) i prelazi se na sledeću iteraciju

```
for broj in range(1,10):
    if broj == 3:
        continue
    print (broj)
```

Primer

- Napisati algoritam i program kojim se za uneti prirodan broj n (n ≥ 1) i realan broj x,izračunava broj S na sledeći način:

$$\sum_{i=1}^{n} \frac{x^i}{i!}$$

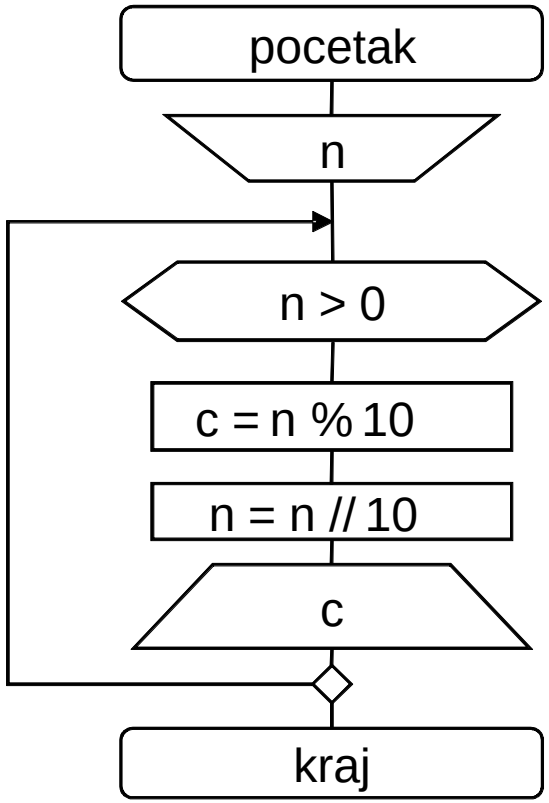
```
n=int(input("Unesi n: "))
x=float(input("Unesi x: "))
s=0
f=1
for i in range(1,n+1):
        f*=x/i
        s+=f

print(s)
```

```
import math as m

for i in range(1,n+1):
        s+=x**i/m.factorial(i)
```

Primer

- Napisati algoritam i program koji za uneti ceo broj **n** ispisuje njegove cifre.

```
n = int(input("Unesi broj: "))

while n > 0:
    c = n % 10
    n = n // 10
    print(c)
```



Primer

- Šta je rezultat rada sledećeg programa?

```
def printMultiples(n):
  i = 1
  while i <= 6:
    print (n*i, '\t', end='')
    i = i + 1
  print()

i = 1
while i <= 6:
  printMultiples(i)
  i = i + 1
```

Stringovi

Rad sa stringovima

- String je niz karaktera ograničen jednostrukim ili dvostrukim navodnicima

```
a="Tekst sa dvosturkim navodnicima"
b='Tekst sa jednostrukim navodnicima'
print(a)
print(b)
```

```
Tekst sa dvosturkim navodnicima
Tekst sa jednostrukim navodnicima
```

Rad sa stringovima

- String mora početi i završiti se istim navodnicima

```
s = 'On je rekao, "Zdravo, svete!"'
t = "Vise navodnika '''''  ali i dalje ispravno."
print(s)
print(t)
```

```
On je rekao, "Zdravo, svete!"
Vise navodnika '''''  ali i dalje ispravno.
```

Rad sa stringovima

- String se može štampati u više linija
  - Ako se započne sa tri navodnika (jednostruka ili dvostruka) i tako i završi
  - Ako se u string umetne specijalni karakter \n

```
s1 = """Ovo
je string koji ima
vise linija."""
print(s1)
s2 = 'Ovo\ntakodje'
print (s2)
```

```
Ovo
je string koji ima
vise linija.
Ovo
takodje
```

Rad sa stringovima

- Specijalni karakteri
  - `\n` – završava trenutnu liniju i nastavlja u sledećoj
  - `\t` – ubacuje tab u string
  - `\'` – ubacuje ' u strung
  - `\"` – ubacuje " u string
  - `\\` – ubacuje \ u string

Konkatenacija stringova

- Dva stringa se spajaju u jedan korišćenjem operatora +

```
s0 = "Zdravo"
s1 = "Svete"
print(s0+s1)
print(s0+' '+s1)
print("Halo, " + "gde si?")
s1 = "Prvi string"
s2 = ", drugi string"
print(s1+s2)
```

```
ZdravoSvete
Zdravo Svete
Halo, gde si?
Prvi string, drugi string
```

Konkatenacija stringova

- Više stringova se spajaju u jedan, na isti način, korišćenjem operatora +

```
s0 = "Spajanje"
s1 = "vise"
s2 = "stringova"
s3 = "zajedno"
razmak = " "
s = s0 + razmak + s1 + razmak + s2 + razmak + s3
print (s)
```

```
Spajanje vise stringova zajedno
```

Kopiranje stringova

```
s = 'Ha'
print (s * 10)
a = 'Program'
print(a * 3)
```

```
HaHaHaHaHaHaHaHaHaHa
ProgramProgramProgram
```

- Množenje nulom i negativnim brojem ne daje nikakav rezultat
- Nije dozvoljeno množenje razlomljenim brojem i sabiranje sa celim brojem

```
print('Hello' * 8.1)
print('123' + 4)
```

Deljenje stringa

- Delovima stringa može se pristupiti preko indeksa navedenog u zagradama [ ]

```
s = 'programiranje'
print ('s = ', s)
print ('s[0] = ', s[0],' s[3] = ', s[3])
print ('s[-1] = ', s[-1])
print ('s[1:5] = ', s[1:5])
print ('s[5:-2] = ', s[5:-2])
```

```
s =  programiranje
s[0] =  p  s[3] =  g
s[-1] =  e
s[1:5] =  rogr
s[5:-2] =  amiran
```

String funkcije

- Dužina stringa – len

```
s = "Hello!"
print (len(s))
print (len("Jedan obican string "))
```

```
6
20
```

```
print (s[2:-1],s[2:len(s)])
```

```
llo llo!
```

String funkcije

- String u stringu – find, index

```
string.find(pattern[,begin[,end]])
```

```
str1 = "Ovo je reprezentacija nekog stringa!"
str2 = "taci"
print (str1.find(str2))                    15
print (str1.find(str2, 10))                15
print (str1.find(str2, 16))                -1
print (str1.find(str2, 10, 20))            15
print (str1.find(str2, 10, 16))            -1
```

String funkcije

- String u stringu – find, index

```
string.index(pattern[,begin[,end]])
```

```
str1 = "Ovo je reprezentacija nekog stringa!"
str2 = "taci"
print (str1.index(str2))
print (str1.index(str2, 16))
```

```
15
ValueError: substring not found
```

String funkcije

- Broj ponavljana stringa u stringu – count
- Konvertovanje svih slova u velika, odnosno mala – upper, lower
- Promena dela stringa novim stringoma – replace

```
s = 'Ovaj predmet se zove PRAKTIKUM'
print (s.count(' '))
print (s.upper())
print (s.lower())
print (s.replace('a','A'))
```

```
6
OVAJ PREDMET SE ZOVE PRAKTIKUM
ovaj predmet se zove praktikum
OvAj predmet se zove PrAktikum
```

Poređenje stringova

```
ime = input()

if ime=="Marko":
    print("Da, moje ime je Marko")

                              ili svi slučajevi:
if ime < "Marko":
    print("Uneto ime je ispred Marka")
elif ime > "Marko":
    print("Uneto ime je iza imena Marko")
else:
    print("Uneto ime je Marko")
```

Stringovi su nepromenljivi

- Posmatrajmo sledeći deo koda:

```
rec = "Osnovi programiranja"
rec[0] = 'A'
print(rec)
```

- Umesto očekivanog izlaza:

```
Asnovi programiranja
```

- Izlaz je runtime error:

```
TypeError: 'str' object does not support item assignment
```

- Predloženo rešenje – formiranje nove reči na osnovu stare:

```
nova_rec = 'A'+rec[1:]
print(nova_rec)
```

Primer

- Šta je rezultat rada sledećeg programa?

```
prvi_sabirak  = input("Unesi prvi sabirak: ")
drugi_sabirak = input("Unesi drugi sabirak: ")
zbir = prvi_sabirak + drugi_sabirak
print(zbir)
```

Primer

- Tekst je palindrom ako se čita sdesna nalevo isto kao i sleva nadesno (radar, kajak). Napisati funkciju koja testira da li je neki tekst palindrom. Problem rešiti kako iterativno, tako i rekurzivno.

```
def palindromI(tekst):
    n = len(tekst)
    limit = n//2
    for i in range(limit):
        if tekst[i] != tekst[-i-1]:
            return False
    return True
```

Primer

```
def palindromR(tekst):
    if tekst == '':
        return True
    elif tekst[0] != tekst[-1]:
        return False
    else:
        return palindromR(tekst[1 : -1])


tekst = input()
if (palindromI(tekst)):  # moze se koristiti i palindromR(tekst)
    print("Jeste")
else:
    print("Nije")
```

Liste

Liste

- Lista je niz podataka proizvoljnog tipa
- Svaki podataka u listi se naziva element

`[1, 4, 17, 256, 3, 59, 45]` - lista celih brojeva

`['april', 'jun', 'septembar', 'novembar']` - lista stringova

`[]` - prazna lista

`['Marina', 28, 101, 'PMF']` - lista elemenata različitog tipa

`[1, 4, [7, 6, [13]], [9, 5]]` - lista čiji elementi su liste

Generisanje liste

```
zivotinje=['pas','macka','mis','majmun','slon']
print (zivotinje)
['pas', 'macka', 'mis', 'majmun', 'slon']

lista=list(range(1,10))
print (lista)
[1, 2, 3, 4, 5, 6, 7, 8, 9]

lista=list(range(1,10,2))
print (lista)
[1, 3, 5, 7, 9]
```

Elementi liste, indeksiranje

- Elementim liste se pristupa preko rednog broja pozicije – indeksa
- Indeks prvog elementa u listi je 0

```
lista = [10, 9, 8, 7,-1,-2,-3]
print (lista[2], lista[-1])
print (lista[1:5])
print (lista[1:5:2])
print (lista[12])
```

```
8 -3
[9, 8, 7, -1]
[9, 7]
IndexError: list index out of range
```

Promena vrednosti

- Elementima liste je moguće promeniti vrednost

```
lista = [2, 4, 6, 8]
lista[1] = 0
lista[-1] = 12
print (lista)
```

```
[2, 0, 6, 12]
```

Spajanje listi

- Dve liste se spajaju operatorom +

```
A = [1,2,3,4,5]
B = [6,7,8,9]
C = A + B
print (C)
print (A + B[1:-1])
B = 'Jovana'
C = A + B
print (C)
B = ['Jovana']
C = A + B
print (C)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 7, 8]

TypeError: can only concatenate list (not "str") to list


[1, 2, 3, 4, 5, 'Jovana']
```

Ponavljanje elemenata liste

- Elementi liste se mogu ponavljati određen broj puta operatorom *

```
C = [0]
A = [1,2,3]
B = 3 * A
D = 4 * C
print(B, D)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3] [0, 0, 0, 0]
```

Funkcije za rad sa listama

- Dužina liste, odnosno broj elemenata liste – len

```
lista = ['jedan', 2, 3, 4, 5, [-2, -4, -5]]
print (len(lista))
print (len(lista[5]))
print (len(lista[0]))
print (len(lista[1]))
```

```
6
3
5
TypeError: object of type 'int' has no len()
```

Funkcije za rad sa listama

- Operator in

```
lista=[1,2,3,4,5]
print(5 in lista)
print(8 in lista)
if (not 10 in lista):
    print(10,"nije u listi")
else:
    print(10,"jeste u listi")
True
False
10 nije u listi
```

Funkcije za rad sa listama

- Dodavanje elemenata na kraj liste – append, extend

```
lista = [1,2,3]
lista.append([4,5])
print (lista)          [1, 2, 3, [4, 5]]
lista.append(6)
print (lista)          [1, 2, 3, [4, 5], 6]
lista.extend([7,8])
print (lista)          [1, 2, 3, [4, 5], 6, 7, 8]
lista.extend(9)        TypeError: 'int' object is not iterable
```

Funkcije za rad sa listama

- Dodavanje elemenata na željenu poziciju – insert

```
A = [1, 2, 3, 4, 5]
A.insert(2,0)
print (A)
```

```
[1, 2, 0, 3, 4, 5]
```

```
A.insert(4,['a','b','c'])
print (A)
```

```
[1, 2, 0, 3, ['a', 'b', 'c'], 4, 5]
```

Funkcije za rad sa listama

- Minimalni i maksimalni element liste – min, max

```
A = [1,13,-2,6,101,28,44,17,-27]
print (min(A), max(A))
A = ['Danijela', 'Anabela', 'Marko', 'Katarina', 'Nikola']
print (min(A), max(A))
A = [2, 3, 4, [1, 6], 5]
print (min(A), max(A))
```

```
-27 101
Anabela Nikola
TypeError: '<' not supported between instances of 'list' and 'int'
```

Funkcije za rad sa listama

- Broj ponavljanja elementa – count

```
A = [2, 3, 4, 1, 7, 2, 3, 1, 1, 0, 9, 5]
print (A.count(1))
odeljenje = ['Ana', 'Marko', 'Jovana', 'Darko', 'Jovana']
print (odeljenje.count('Jovana'))
A = [2, [3, 4], 1, [3, 4], 3, 1, 1, 0, 9, 5]
print (A.count(3), A. count([3,4]))
```

```
3
2
1 2
```

Funkcije za rad sa listama

- Brisanje elementa iz liste – del, pop, remove

```
A=[1, 2, 3, 4, 5, 6, 7, 4, 8, 9]
del(A[3])
print (A)                    [1, 2, 3, 5, 6, 7, 4, 8, 9]
A.pop(1)
print (A)                    [1, 3, 5, 6, 7, 4, 8, 9]
A.pop()
print (A)                    [1, 3, 5, 6, 7, 4, 8]
A.remove(5)
print (A)                    [1, 3, 6, 7, 4, 8]
```

Funkcije za rad sa listama

- Sortiranje i okretanje elemenata liste – sort, reverse

```
A = [2,6,1,9,3,5,4]
A.sort()
print (A)                    [1, 2, 3, 4, 5, 6, 9]
A.reverse()
print (A)                    [9, 6, 5, 4, 3, 2, 1]
A = ['Pera', 'Laza', 'Mika', 'Aca']
A.sort()
print (A)                    ['Aca', 'Laza', 'Mika', 'Pera']
```

Liste i petlje

```
for voce in ["banana", "jabuka", "kruska"]:
  print ("Volim da jedem " + voce)
```

```
Volim da jedem banana
Volim da jedem jabuka
Volim da jedem kruska
```

Liste i stringovi

- Koliko data rečenica ima reči?

```
recenica = "Danas je divan dan"
lista_reci = list(recenica.split())
print(lista_reci)
print(len(lista_reci))
```

`recenica.split()` – od date rečenice pravi listu čiji su elementi dobijeni razvajanjem rečenice po jednom (ili više) blanko karakteru

Funkcija `split()` se može primeniti dikrento na ulaz:

```
lista_reci = list(input().split())
```

Liste i stringovi

- Metoda `join()` poziva se nad listom `stringova`, formirajući jedan string od elemenata liste

```
L = ['Tata','kaže','U','laži','su','kratke' ,'noge']
recenica=' '.join(L)
print(recenica)
```

```
Tata kaže U laži su kratke noge
```

Formiranje liste brojeva

- Kako formirati listu brojeva na prethodno opisan način?

```
brojevi= list(input().split())
print(brojevi)    # kog tipa su elementi liste?
1 2 3.3 5.6
['1', '2', '3.3', '5.6']
```

Ako želimo brojeve potrebno je svaki element liste pretvoriti u broj. Na koji način?

```
brojevi= list(map(int, input().split()))
```

ili

```
for i in range(len(brojevi)):
 brojevi[i] = float(brojevi[i])
```

```
[1, 2, 3.3, 5.6]
```

Formiranje liste brojeva

- Formirati listu koja za dat ceo broj **n** sadrži kvadrate brojeva od 0 do **n**, ne računajući **n**?

```
n = int(input())
L1 = [ i**2 for i in range(n) ]
print (L1)


5
[0, 1, 4, 9, 16]
>>>
```

Filtriranje

- U mnogim zadacima je potrebno od date liste formirati novu koja sadrži elemente koji zadaovoljavaju neki dati uslov.

```
lista = [1, 2, 3, 4, 5, 6, 7, 8]
nova_lista = [x for x in lista if x % 2==0]
print(nova_lista)

[2, 4, 6, 8]
>>>
```

Liste kao argumenti funkcija

- Lista je promenljive prirode i prilikom prenošenja u funkciju može doći do (ne)željenog sporednog efekta

```
def f(lista):
    lista.append(100) # sporedni efekat
    lista = [] # lista se redefiniše kao lok. prom.
    return lista

lista = [1, 2]
print('lista pre poziva', lista)
print('rezultat f(lista):', f(lista))
print('lista posle poziva', lista)
lista pre poziva [1, 2]
rezultat f(lista): []
lista posle poziva [1, 2, 100]
```

Liste kao argumenti funkcija

- Ukoliko ne želimo da promenimo listu:

- lista se, pri navodjenju stvarnih parametara u pozivu funkcije, može kopirati u novu listu

```
lista = [1, 2]
print('lista pre poziva', lista)
print('rezultat f(lista):', f(lista[:]))
print('lista posle poziva', lista)
```

```
lista pre poziva [1, 2]
rezultat f(lista): []
lista posle poziva [1, 2]
```

Primer

Napisati funkciju koja datu listu sortira u nerastućem poretku. U glavnom delu programa uneti listu od n celih brojeva a zatim korišćenjem napisane funkcije sortirati elemente liste.

```
def sortiranje(lista):
    lista.sort()

A=[]
n=int(input())
for i in range(n):
    A.insert(i,int(input()))

print(A)
sortiranje(A)
print(A)
```

Primer

```
def sortiranje(lista):
    for i in range(len(lista)-1):
        for j in range(i+1, len(lista)):
            if (lista[i] <= lista[j]):
                lista[i],lista[j] = lista[j],lista[i]
A=[]
n=int(input())
for i in range(n):
    A.insert(i,int(input()))

print(A)
sortiranje(A)
print(A)
```

Primer

Napisati algoritam i program kojim za unetu listu od **n** celih brojeva.

a) Štampa elemente liste u obrnutom redosledu.

b) Rotira elemente liste za:

- Jedno mesto ulevo
- Jedno mesto udesno

Primer

a)

```
A=[]
n=int(input())
for i in range(n):
    A.insert(i,int(input()))


for i in range(n-1,-1,-1):
    print(A[i])
```

Primer

a)

```
A=[]
n=int(input())
for i in range(n):
    A.insert(i,int(input()))


for i in range(n-1,-1,-1):
    print(A[i])
```

Primer

b)

```
b=A[0]

for i in range(0,n-1):
    A[i]=A[i+1]

A[n-1]=b
print(A)
```

```
b=A.pop(0)
A.insert(n-1,b)
print(A)
```

Primer

b)

```
b=A[n-1]

for i in range(n-1,0,-1):
    A[i]=A[i-1]

A[0]=b
print(A)
```

```
b=A.pop()
A.insert(0,b)
print(A)
```

Primer

Za unet prvi član aritmetičkog niza, i razliku aritmetičkog niza, napraviti prvih 20 elemenata aritmetičkog niza i odštampati ih.

Primer

```
print('Unesite prvi clan aritmetickog niza')
a1 = int(input())
print('Unesite razliku aritmetickog niza')
razlika = int(input())
niz = [a1]
for i in range(19):
    a1 = a1 + razlika
    niz = niz +[a1]
print('Aritmeticki niz je:')
print(niz)
```

Primer

Napisati funkciju koja vraća listu prvih 100 Fibonačijevih brojeva. Nakon toga filtrirati listu tako da sadrži samo parne elemente.

Izlaz:

[0, 2, 8, 34, 144, 610, ..., 5168070885485832307 2, 218922995834555169026]

Primer

```
def fibonacci():
    l = list()

    l.append(0)
    l.append(1)

    for i in range(2, 100):
        l.append(l[i - 2] + l[i - 1])
    return l

def paranBroj(x):
    if(x % 2 == 0):
        return True
    return False

lista = fibonacci()
#Filtriranje - prvi nacin
novaLista = [x for x in lista if x % 2 == 0]
#Filtriranje - drugi nacin
novaLista = list(filter(paranBroj, lista))

print(novaLista)
```

Lista listi

Lista listi

- Svaki podataka u listi je lista

```
[[1, 2], [3]]
```
- liste u okviru liste nemaju jednak broj elemenata

```
[[1, 2], [2, 5]]
```
- lista ima dve liste i svaka od njih po dva elementa

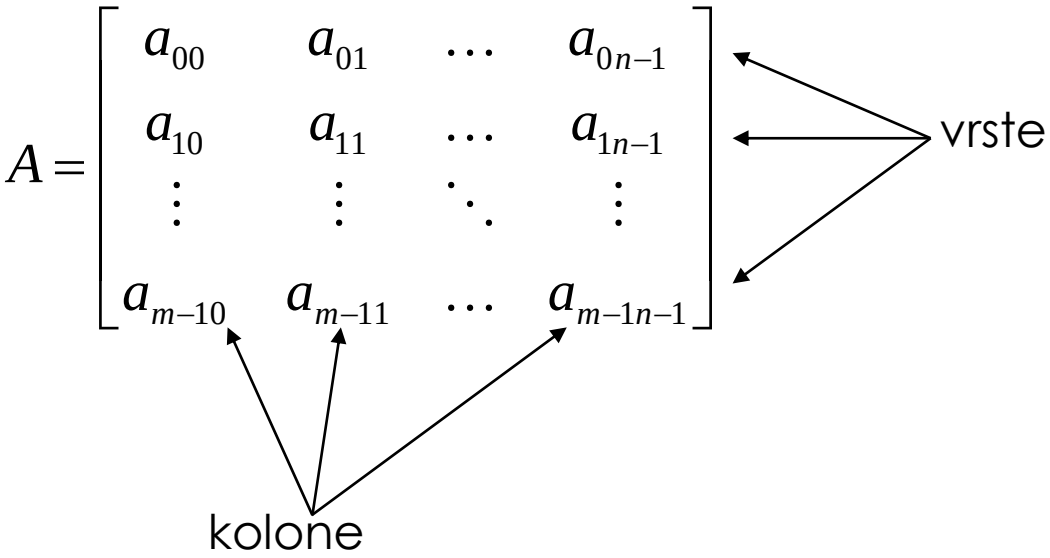
- Pristup element unutar liste:

```
A = [[1, 2], [2, 5]]
print(A)
B = A[0]
print(B)
k = A[0][1]
print(k)
```

```
[[1, 2], [2, 5]]
[1, 2]
2
```

Matrica

- Lista listi se koristi za reprezentaciju matrica u Python-u
- `Matrica` dimenzije *m×n* je šema brojeva koja se zapisuje kao:



- Matrica kod koje je ***m = n*** zove se `kvadratna matrica`.

Primer

- Napisati algoritam i program kojim se za unete prirodne brojeve ***m*** i ***n*** koji predstavljaju dimenzije matrice, unose celi brojevi koji predstavljaju elemente matrice po vrstama.
- Napisati funkciju **ZbirPoObodu(mat,m,n)** koja računa i vraća zbir elemenata te matrice koji su raspoređeni po obodu.

Primer

```
m=int(input("Unesi broj vrsta"))
n=int(input("Unesi broj kolona"))

matrix = [[0 for i in range(n)] for i in range(m)] #matrica
dimenzije nxm sa nulama

for i in range(m):
    for j in range(n):
        matrix[i][j]=int(input())

print(matrix)
print(matrix[:][0])
print(ZbirPoObodu(matrix,m,n))
```

Primer

```
def ZbirPoObodu(e,m,n):
    s=sum(e[0][:])+sum(e[m-1][:])#+sum(e[1:m-1][0])+sum(e[1:m-1][n-1])

    for i in range(1,m-1):
        s+=e[i][0]+e[i][n-1]
    return s
```

Primer

- Napisati algoritam i program kojim se za uneti prirodan broj ***n***, unose celi brojevi koji predstavljaju elemente matrice dimenzije *n x n* po vrstama.
- Napisati funkciju **ZbirParnihVrsta(mat,n)** koja vraća niz koji predstavlja zbir elemenata matrice koji pripadaju parnim vrstama

Primer

```
def ZbirParnihVrsta(e,n):
    b=[sum(e[i][:]) for i in range(0,n,2) ]
    return b

n=int(input("Unesi dimenziju"))

matrix = [[0 for i in range(n)] for i in range(n)] #matrica dimenzije nxn
sa nulama

for i in range(n):
    for j in range(n):
        matrix[i][j]=int(input())


print(ZbirParnihVrsta(matrix,n))
```

Rečnici

Rečnici

- Rečnici su generalizovana verzija liste. Posmatrajmo listu koja sadrži broj dana u mesecima godine

```
dana = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

- Ako hoćemo broj dana u mesecu januaru koristimo `dana[0]`, ili u decembru `dana[11]` ili `dana[-1]`
- Rečnik dana u mesecima godine:

```
dani ={'Januar':31, 'Februar':28, 'Mart':31, 'April':30,
       'Maj':31, 'Jun':30, 'Jul':31, 'Avgust':31,
       'Septembar':30, 'Oktobar':31, 'Novembar':30, 'Decembar':31}
```

- Sada pristupamo sa:

```
dani['Januar']
```

- Prednost: kod je čitljiv i ne treba razmišljati o indeksu

Osnovne operacije sa rečnicima

- Da označimo da je nešto rečnik koristimo zagrade `{}`
- Svaki element u rečniku je par podataka razdvojen sa dve tačke
- Prvi deo u paru se naziva `ključ (key)`, drugi je `vrednost (value)`
- Ključ igra ulogu sličnu indeksu

```
d = {'A':100, 'B':200}
```

ključ

vrednost

```
d['A'] = 100
```

- Ključevi su često stringovi, mogu biti i celi i realni brojevi
- U istom rečniku mogu se naći ključevi različitog tipa

Osnovne operacije sa rečnicima

```
d = {'A':100, 'B':200}
```

- Promena vrednosti:

```
d['A'] = 400
```

- Dodavanje novog elementa u rečnik:

```
d['C'] = 500
```

Ovo nije moguće sa listama. Ako napišemo L[2]=500 za listu L koja ima samo dva elementa L[0] i L[1], dobićemo grešku: „index out of range“

- Brisanje elementa iz rečnika:

```
del d['A']
```

Osnovne operacije sa rečnicima

- Prazan rečnik je `{}`, čemu je analogno `[]` za praznu listu, ili `''` za prazan string.
- Redosled elemenata u rečniku ne mora biti isti kao redosled dodavanja elemenata u rečnik
- Python rearanžira elemente rečnika u cilju optimizacije performansi.

Osnovne operacije sa rečnicima

- U igri Scrabble, svakom slovu je pridružena vrednost. Možemo koristiti sledeći rečnik za vrednost slova u njemu:

```
points = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
          'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
          'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
          'V':4, 'W':4, 'X':8, 'Y':4, 'Z':10}
```

- Da izračunamo skor za neku reč koristimo kod

```
skor = sum([points[c] for c in word])
```

Osnovne operacije sa rečnicima

Rečnik omogućava lepu varijantu za prikaz špila karata:

```
špil = [{'vrednost':i, 'boja':c}
        for c in ['pik', 'tref', 'herc', 'karo']
        for i in range(2,15)]
```

Špil je lista sa 52 rečnika. Metod `shuffle` se može iskoristiti za mešanje špila:

```
shuffle(špil)   //  from random import shuffle
```

Prva karta u špilu je `špil[0]`. Boja i vrednost karte:

```
špil[0]['vrednost']
špil[0]['boja']
```

Rad sa rečnicima

```
d = {‘A’:100, ‘B’:200}
```

- Kopiranje rečnika:

```
d2=d.copy()
```

- Operator `in` se koristi da nam kaže da li je neki ključ u rečniku ili nije.
- Ako pokušamo da dobijemo vrednost nekog ključa koji nije u rečniku dobićemo grešku.
- `print(d['C’])  - greška`

- Možemo da sprečimo grešku korišćenjem operatora `in`. Pre upotrebe nekog ključa, možemo da proverimo da li je u rečniku.

Rad sa rečnicima

```
slovo = input('Unesite slovo: ')
if slovo in d:
    print('Vrednost je', d[slovo])
else:
    print('Nije u rečniku')
```

Može se koristiti i `not in`

Rad sa rečnicima

```
tabela = {1 : 'Srbija', 2 : 'Italija', 3 : 'Francuska', 4 : 'Spanija’}
```

Metoda `pop()` briše pridruživanje za zadati ključ i tom prilikom vraća vrednost koja se izbacuje iz rečnika.

```
print(tabela.pop(2), tabela)
Italija {1: 'Srbija', 3: 'Francuska', 4: 'Spanija’}
tabela.pop(5)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    tabela.pop(5)
KeyError: 5
```

Rad sa rečnicima

```
tabela = {1 : 'Srbija', 2 : 'Italija', 3 : 'Francuska', 4 : 'Spanija’}
```

- Kada se u metodi navede opcioni parameter greška se ne pojavljuje:

```
tabela.pop(5, None)
None
```

- Brisanje svih elemenata iz rečnika:

```
tabela.clear()
print(tabela)
{}
```

Rad sa rečnicima

```
d = {'A':100, 'B':200}
```

- Može se napraviti petlja kroz rečnik na sličan način kao i kod liste.
- Štampanje svih ključeva rečnika:

```
for kljuc in d:
    print(kljuc)
```

- Štampanje vrednosti iz rečnika:

```
for kljuc in d:
    print(d[kljuc])
```

Rad sa rečnicima

```
d = {'A':100, 'B':200}
```

- Kako od rečnika dobiti liste ključeva i vrednosti:

```
list(d)   -  ['A','B'], ključevi iz rečnika d
list(d.valies())  -  [100,200], vrednosti iz rečnika d
list(d.items()) -  [('A',100),('B',200)], parovi (ključ,vrednost)
```

- Parovi koje vraća `d.items()` nazivaju se `torke (tuples)`. Torke su slične listama.

Rad sa rečnicima

- Svi ključevi rečnika kojima odgovara vrednost 100.

```
d = {'A':100, 'B':200, 'C':100}
L = [x[0] for x in d.items() if x[1]==100] #filtriranje liste ključeva
```

```
['A', 'C']
```

Rad sa rečnicima

- Funkcija `dict` je dodatan način kreiranja rečnika.

```
d = dict([('A',100),('B',300)])
{'A':100, 'B':300}
```

- `Sastavljanje rečnika` je slično sastavljanju liste. Sledeći jednostavan primer kreira rečnik od liste reči u kojem će vrednosti u rečniku biti dužine odgovarajućih reči:

```
reci=['jabuka','banana','jagoda']
d = {s : len(s) for s in reci}
{'jabuka': 6, 'banana': 6, 'jagoda': 6}
```

Primer

- Poznate su geografske koordinate nekoliko glavnih evropskih gradova. Za dato ime grada odredi njene geografske koordinate. Odredi posebno geografsku dužinu i posebno geografsku širinu.

```
gradovi = {"Beograd": (44.7866, 20.4489),
           "Budimpešta": (47.4979, 19.0402),
           "Beč": (48.2082, 16.3738),
           "Bratislava": (48.1486, 17.1077)}
```

Primer

```
grad = input("Unesi ime grada: ")

koordinate = gradovi[grad]
print(koordinate)

print("Geografska širina: ", koordinate[0])
print("Geografska dužina: ", koordinate[1])
```

Primer

- Neka je dat rečnik:

```
d=[{'ime':'Janko', 'telefon':'555-1414', 'email':'janko@mail.net'},
   {'ime':'Marko', 'telefon':'555-1618', 'email':'marko@mail.net'},
   {'ime':'Ana', 'telefon':'555-3141', 'email':'';},
   {'ime':'Jovan', 'telefon':'555-2718’, 'email':'jovan@mail.net'}]
```

Napišite program koji čita ovaj rečnik i štampa sledeće:

- Sve korisnike čiji telefonski broj se završava sa 8.
- Sve korisnike koji nemaju email adresu.

Torke

Torke (Tuples)

- Torka je nepromenljiva lista – ne može da se menja po formiranju

```
L = [1, 2, 3]
T = (1, 2, 3)
```

- Torke se navode unutar malih zagrada ili bez njih.
- Indeksiranje i segmentacija su isti kao i kod liste.
- Kao i kod liste funkcijom `len` možete dobiti dužinu torke.
- Torke imaju kao i liste `count` i `index` metode.
- Međutim, pošto su torke nepromenljive, torke nemaju neke metode koje imaju liste, kao što su `sort` ili `reverse`.

Torke (Tuples)

```
          nole = ('Novak', 'Đoković', 1987, 'Srbija', 12)
>>> nole[4]
12
>>> nole[4]=13     #greška
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nole[4]=13
TypeError: 'tuple' object does not support item assignment
>>> nov_nole = nole[0:3] + (13,)
>>> nov_nole
('Novak', 'Đoković', 1987, 13)
>>>
```

Torke (Tuples)

- Jedan od razloga zašto postoje i liste i torke je što su nam u nekim situacijama potrebne nepromenljive liste. Na primer:
  - lista ne može da bude ključ u rečniku (lista se može menjati, problem je kako pratiti promene ključa)
  - torke mogu da igraju ulogu ključa u rečnicima.

```
ocene = {('Petar', 'Ana'): 95, ('Milovan', 'Igor'): 87}
```

Torke (Tuples)

- Brže od listi
- Za konverziju nekog objekta u torku koristimo funkciju `tuple`.

```
t1 = tuple([1,2,3])      (1, 2, 3)
t2 = tuple('abcde’)      ('a', 'b', 'c', 'd', 'e')
```

- Prazna torka je `()`.
- Torka sa jednim elementom `(1,)`.
- `(1)` matematički izraz koji ima vrednost 1.

- Torke se koriste za vraćanje više vrednosti iz funkcije
- Torke mogu biti elementi liste

Torke (Tuples)

- Torke mogu biti elementi liste

```
meseci = [("јануар", 31), ("фебруар", 28), ("март", 31), \
          ("април", 30), ("мај", 31), ("јун", 30), \
          ("јул", 31), ("август", 31), ("септембар", 30), \
          ("октобар", 31), ("новембар", 30), ("децембар", 31)]
broj = int(input("Унеси редни број месеца:"))
mesec = meseci[broj - 1]
print("Назив:", mesec[0], "Број дана:", mesec[1])
```

Primer

- Data je lista sa podacima o prosečnoj temperaturi za svaki mesec tokom jedne godine. Dopuniti kod tako da za bilo koju listu temperatura ovog oblika određuje minimalnu temperaturu tokom godine.

```
temp = [('jan',-3), ('feb',0), ('mart',9), ('apr',18), ('maj',25),
('jun',30), ('jul',35), ('avg',37), ('sep',28), ('okt',20), ('nov',10),
('dec',2)]
mint = _________   # dopuniti ovaj red
for i in range(1,_____):   # dopuniti ovaj red
    if temp[___][___]<mint:   # dopuniti ovaj red
        mint = _________   # dopuniti ovaj red
print(mint)
```

Skupovi

Skupovi (Sets)

- Python ima i strukturu koja se zove `set (skup)`.
- Set struktura radi slično matematičkim skupovima.
- Skupovi su poput listi u kojima nije dozvoljeno ponavljanje elemenata.
- Skupovi se označavaju vitičastim zagradama, `{}`

```
S = {1,2,3,4,5}
```

- Prazan skup dobijamo pomoću funkcije `set()`

```
S = set()
```

Skupovi (Sets)

- Funkcija set se može koristiti i za konverziju objekata u skupove.

```
set([1,4,4,4,5,1,2,1,3])
set('this is a test')
{1, 2, 3, 4, 5}
{'a', ' ', 'e', 'i', 'h', 's', 't’}
```

- Skupove možemo sastavljati na različite načine:

```
s = {i**2 for i in range(12)}
{0, 1, 4, 100, 81, 64, 9, 16, 49, 121, 25, 36}
```

Skupovi (Sets)

- Nad skupovima je moguće primeniti relacijske operatore:
  - == (jednakost)
  - <= (podskup)
  - < (pravi podskup)
  - >= (nadskup)
  - > (pravi nadskup)

Skupovi (Sets)

- Python smešta podatke u skup u proizvoljnom redolsedu, i ne uvek u redosledu koji ste vi postavili.
- Kod skupova nije važan redosled, već samo podaci koji ga čine. To znači da indeksiranje kod skupova nema smisla. Zato se ne može , na primer, tražiti s[0], za set s.
- Operacije za rad sa skupovima

Operator	Opis	Primer
|	Unija	{1,2,3} | {3,4}→{1,2,3,4}
&	Presek	{1,2,3}&{3,4}→{3}
-	Razlika	{1,2,3}-{3,4}→{1,2}
^	Simetrična razlika	{1,2,3}^{3,4}→{1,2,4}
in	Jeste u skupu	3 in {1,2,3}→True

Skupovi (Sets)

- Metode koje se primenjuju na skupove

Operator	Opis
S.add(x)	Dodaje x u skup
S.remove(x)	Sklanja x iz skupa
S.issubset(A)	Vraća True ako je S ⊂ A, a False ako nije
S.issuperset(A)	Vraća True ako je A ⊂ S, a False ako nije.

Primer

```
>>> {'crvena', 'plava'} == {'plava', 'crvena'}
True
>>> a, b, c = {1, 2, 3, 4, 5}, {3, 4, 5}, {1, 2}
>>> c.add(3) # metod za ubacivanje elementa u skup
>>> c
{1, 2, 3}
>>> c.discard(3) # uklanja element ako postoji
>>> c
{1, 2}
>>> a & b # presek skupova
{3, 4, 5}
>>> b & c # presek je prazan
set()
```

Primer

```
>>> b | c # unija skupova
{1, 2, 3, 4, 5}
>>> a - b # razlika skupova
{1, 2}
>>> a >= b # a nadskup b
True
>>> b <= a # b podskup a
True
>>> a.clear() # uklanja sve elemente iz skupa
>>> a
set()
```

Primer

- Brisanje duplih elemenata iz liste
- Možemo iskoristiti činjenicu da skupovi ne mogu imati ponovljene elemente.

```
L = [1,4,4,4,5,1,2,1,3]    # lista sa ponavljanjem elemenata
L = list(set(L))           # konvertujemo listu u skup, pa nazad u listu
```

```
[1,2,3,4,5]
```

Primer

- Napisati program u kojem se unosi broj n a zatim n celih brojeva. Za sve cifre koje su se javile u zapisu unetih brojeva odrediti koliko puta su se ukupno javile.

Primer

```
n = int(input())
brojevi = []
for i in range(n):
    brojevi = brojevi + [int(input())]
print(brojevi)

lista_cifara = []
for x in brojevi:
    while x > 0:
        lista_cifara = lista_cifara + [x % 10]
        x = x // 10
print(lista_cifara)

skup_cifara = set(lista_cifara)
for x in skup_cifara:
    print(x," : ", lista_cifara.count(x))
```

```
3
121
23568
5
[121, 23568, 5]
[1, 2, 1, 8, 6, 5, 3, 2,
5]
1  :  2
2  :  2
3  :  1
5  :  2
6  :  1
8  :  1
>>>
```

Razni zadaci

Zadatak 1

Milica je kupovala razne proizvode i svaki put kada se vratila iz kupovine dopisivala je sve proizvode koje je kupila na spisak. Napiši program koji pomaže Milici da odredi proizvod koji je najčešće kupovala tokom prethodne godine.

Na ulazu se zadaje **n** (broj proizvoda), a zatim **n** proizvoda, svaki u novom redu (string).

Na standardni izlaz ispisati jedan red koji sadrži naziv najčešće kupovanog proizvoda, jedan razmak i broj puta koliko je taj proizvod kupljen. Ako je više proizvoda kupljeno isti broj puta, ispisati onaj koji je prvi po abecednom redosledu.

Zadatak 1

Ulaz:
9
jabuka
hleb
kruska
jabuka
sljiva
hleb
mleko
jabuka
hleb

Izlaz:
hleb 3

Zadatak 1

```
recnik = dict()

n = int(input())
for i in range(n):
      proizvod = input()
      if (recnik.get(proizvod) == None):
            recnik[proizvod] = 1
      else:
            recnik[proizvod] += 1

Max = list(recnik.items())[0]

print("---------------")
```

Zadatak 1

```
for r in recnik.items():
      # prvi uslov je za vrednost, drugi ukoliko je isti broj, gledamo leksikografski
      if (r[1] > Max[1]) or (r[1] == Max[1] and r[0] < Max[0]):
            Max = r

print(Max[0] + " " + str(Max[1]))
```

Zadatak 2

Katarina je odlučila da svojoj drugarici pošalje šifrovanu poruku, koja sadrži samo slova engleske abecede, cifre i interpunkcijske znake. Svako slovo će šifrovati posebno na osnovu narednih pravila. Mala slova se šifruju velikim slovima tako što se slovo a šifruje slovom Z, slovo b šifruje slovom Y, c slovom X itd., sve do slova y koje se šifruje slovom B i z koje se šifruje slovom A. Velika slova se šifruju potpuno analogno - od A koje se šifruje sa z do Z koje se šifruje sa a. Ostali karakteri se ne menjaju.

Sa standardnog ulaza unosi **se jedna linija teksta**.

Na standardni izlaz ispisati **šifrovani tekst**.

Ulaz:

Zdravo svima.

Izlaz:

aWIZEL HERNZ

Zadatak 2

```
def pretvoriKarakter(c):
    noviKar = ord(c) # dobijanje ASCII vrednosti od karaktera
    if (c >= 'a' and c <= 'z'):
        inkrement = 25 - 2 * (ord(c) - ord('a'))
        noviKar = ord(c) + inkrement - 32
        # -32 da bismo prebacili u veliko slovo
    elif (c >= "A" and c <= "Z"):
        inkrement = 25 - 2 * (ord(c) - ord('A'))
        noviKar = ord(c) + inkrement + 32
        # +32 da bismo prebacili u malo slovo

    return chr(noviKar) # dobijanje karaktera od ASCII vrednosti

s = input()
for slovo in s:
    novoSlovo = pretvoriKarakter(slovo)
    print(novoSlovo, end='')
```

Zadatak 3

- Data su dva niza iste dužine **v** i **t**. Niz **t** sadrži vremenske trenutke [s], a niz **v** brzine [m/s] tela izmerene u tim vremenskim trenucima. Napisati funkciju koda koji računa ukupni pređeni put i prosečnu brzinu tela na tom putu.

Zadatak 3

```
def PredjeniPut(v,t):
    s=0
    t1=0
    for i in range(0,len(v)):
        s+=v[i]*(t[i]-t1)
        t1=t[i]
    vp=s/t[-1]
    print(vp)
    print(s)

PredjeniPut([20,18,22,17],[1200,1680,2900,3700])
19.751351351351353
73080
```

Zadatak 4

- Napisati funkciju **Polinom(c, x)** koja računa vrednost polinoma $c[0] + c[1] \cdot x + c[2] \cdot x^2 + \cdots + c[n-1] \cdot x^{n-1}$.
- Može li se zadatak rešiti bez petlji?

```
def Polinom(c,x):
    p = 0
    for i in range(len(c)):
        p+=(c[i]*x**i)
    return p

print(Polinom([1,-2,3],2))
9
```

```
def Polinom(c,x):
    a1=[x**i for i in range(len(c))]
    d=[b[0]*b[1] for b in list(zip(c, a1))]
    return sum(d)
```

Zadatak 5

- Data je kvadratna jednacina kao string oblika ”ax^2+bx+c=0”, recimo ”-3x^2+2x-1=0”. Napisati funkciju koja kao argument uzima ovakav string, a vraća rešenja kvadratne jednačine.

```
def KvJed(s):
    i=s.find("x")
    if i==0:
        a=1
    else:
        a=int(s[0:i])
    s=s[i+3:len(s)]
    i=s.find("x")
    if i==0:
        b=1
    else:
        b=int(s[0:i])
```

Zadatak 5

```
s=s[i+1:len(s)]
    i=s.find("=")
    if i==0:
        c=0
    else:
        c=int(s[0:i])
    print(a,b,c)
    d=b**2-4*a*c
    if d<0:
        print("Resenja su kompleksni brojevi")
    elif d==0:
        print(-b/(2*a))
    else:
        x1=(-b+m.sqrt(d))/(2*a)
        x2=(-b-m.sqrt(d))/(2*a)
        print(x1,x2)

KvJed("-3x^2+2x-1=0")
KvJed("x^2-2x-2=0")
```

Zadatak 6

- Napisati algoritam i program kojim se za unete prirodne brojeve ***m*** i ***n*** (1 ≤ ***m***, ***n*** ≤ 20) koji predstavljaju dimenzije matrice, unose celi brojevi koji predstavljaju elemente matrice po vrstama i prirodni brojevi ***l*** i ***k*** (1 ≤ ***l***, **k** ≤ ***m***), a zatim razmenjuje vrednosti elementima u vrstama ***l*** i ***k***.

Zadatak 6

```
n=int(input("Unesi broj vrsta"))
m=int(input("Unesi broj kolona"))
l=int(input("Unesi vrstu l"))
k=int(input("Unesi vrstu k"))

matrix = [[0 for i in range(m)] for i in range(n)]
#matrica dimenzije nxm sa nulama

for i in range(n):
    for j in range(m):
        matrix[i][j]=int(input())


for j in range(m):
        matrix[k][j], matrix[l][j] = matrix[l][j], matrix[k][j]

print(matrix)
```

```
Unesi broj vrsta3
Unesi broj kolona4
Unesi vrstu l1
Unesi vrstu k2
3
4
5
6
7
8
9
10
11
12
1
2
[[3, 4, 5, 6], [11, 12, 1,
2], [7, 8, 9, 10]]
>>>
```