PROGRAMIRANJE U PYTHONU

Zbirka riješenih i pojašnjenih zadataka u programskom jeziku Python za učenike drugog razreda srednje škole

Autor: Zoran Hercigonja, mag.edu.inf

ISBN: 978-953-59549-0-3

Impressum:

Naslov: **Programiranje u Pythonu**

Autor: **Zoran Hercigonja, mag.edu.inf.**

Nakladnik: **Vlastita naklada autora**

URL: **http://issuu.com**

Mjesto i godina izdavanja: **Imbriovec Jalžabetski, 2017.**

ISBN: 978-953-59549-0-3

Sadržaj:

Predgovor

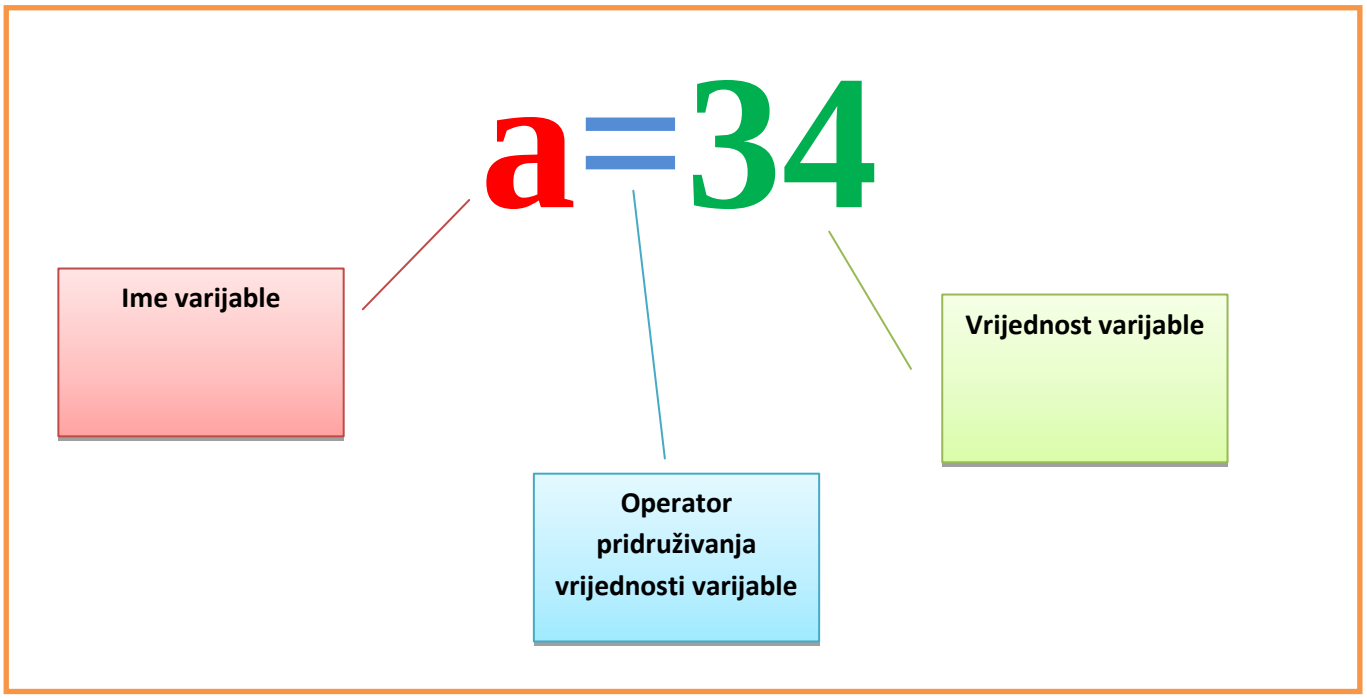
Zbirka zadataka "Programiranje u Pythonu" je namijenjena svim učenicima drugih razreda srednjih škola koji se po prvi puta na nastavi informatike susreću s programskim jezikom Python i njegovim temeljnim programskim strukturama. U zbirci su obrađena sljedeća područja: Varijable, Naredbe unosa i ispisa, If-else uvjetovanje, Elif uvjetovanje, For petlja, While petlja, Funkcije, Jednodimenzionalna polja, Dvodimenzionalna polja i Klase. Svaka programska struktura je detaljno opisana te popraćena mnoštvom programskih primjera napisanih prema težini. Za svaki programski primjer je dati njegov ispis i konačno rješenje kao i opis samog programa. Zbirka je također namijenjena svima onima koji se po prvi puta susreću s Python programskim jezikom i žele naučiti osnove programiranja u tom jeziku.

1. VARIJABLE

Varijabla je memorijska lokacija simboličnog imena u koju se sprema vrijednost nekog podatka. Varijabla se sastoji od imena, adrese i vrijednosti.

Glavni postulati varijable su sljedeći:

a) Ime variajble smije sadržavati samo brojeve, velika i mala slova engleske abecede i donju crticu _
b) Ime ne smije početi s brojem
c) Python razlikuje velika i mala slova



Vrijednost varijabli pridružujemo do desna na lijevo.Varijablama je moguće osim brojčanih vrijednosti pridružiti i tekstualen vrijednosti kao na primjer:

```
a = 'informatika'
```

Samo naravno treba voditi brigu o znaku pridruživanja da se ne poistovjeti sa dvostruko jednako == što predstavlja operator usporedbe.

Osnovni tipovi operatora s kojima varijable mogu raditi su:

Aritmetički

Znak	Naziv operacije
+	**Zbrajanje**
-	**Oduzimanje**
*	**Množenje**
/	**Dijeljenje**
%	**Dijeljenje s ostatkom (modularno dijeljenje)**
**	**Potenciranje**

Relacijski

Znak	Naziv operacije
>	**Veće**
<	**Manje**
>=	**Veće i jednako**
<=	**Manje i jednako**
==	**Jednako**
!=	**Različito**

Logički

Znak	Naziv operacije
And	**Logički i**
Or	**Logički ili**
Not	**Logički ne**

Istovjetnosti

Znak	Naziv operacije
Is	**Istovjetan-vraća vrijednost TRUE**
Is not	**Nije istovjetan-vraća vrijednost FALSE**

Uza prikazane operatore, varijable često poprimaju i određeni tip podataka.

Tipovi podataka, dijele se na :

Brojevi:

cjelobrojni tip podatka **integer**

realni tip podatka **float**

logički tip podatka **bool**

Znakovi:

string-niz znakova

Int()-pretvara u cijeli broj odbacujući decimalu

float()-pretvara u realni broj

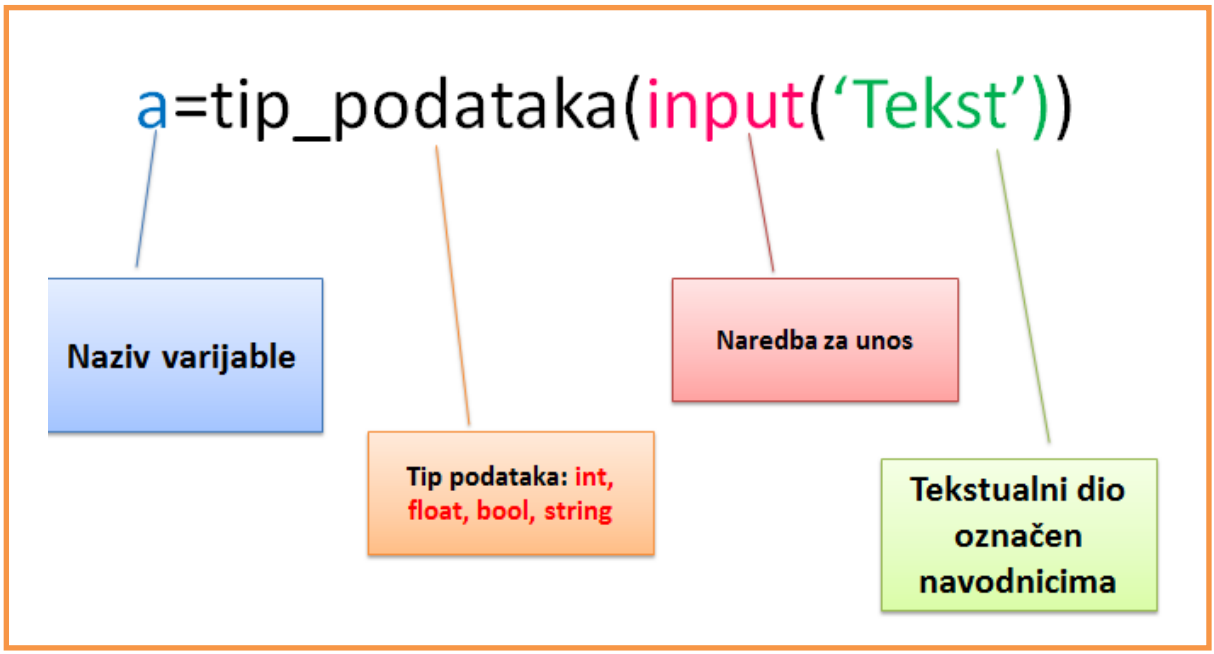
bool()-radi s logičkim tipovima podataka (true, false)

string()-radi sa znakovnim tipovima podataka

2. NAREDBE UNOSA I ISPISA

INPUT

Da bi mogli manipuirati podacima i nad njima vršiti različite operacije, potrebno je dobaviti podatke nad kojima će se moći vršiti različite operacije. Kako bi omogućili upisivanje tekstualne ili brojčane vrijednost u Python programski jezik, potrebno je upotrijebiti naredbu **input.**



Naredba **input** sama za sebe ne nači ništa. Nju moramo pridružiti nekoj varijabli kao običnu brojčanu ili tekstualnu vrijednost. Navikli smo različitim varijablama pridruživati različite vrijednosti no uobičajenim pridruživanjem vrijednosti nekoj varijabli (na primjer: **a=5**), odredili smo vrijednost unutar skripte programaskog jezika. Kako bi omogućili korisniku da preko tipkovnice unosi podatke u Python shell sučelje za izvršavanje programa, potrebno je upotrijebiti upravo naredbu **input.** Kao što smo dosad pridruživali neku vrijednost varijabli pomoću znaka pridruživanja „=“, sada ćemo naredbu **input** zajedno s njezinom sintaksom pridružiti nekoj varijabli **a**. Kako u Pythonu unaprijed nije zadan tip podataka, on svaki unos interpretira kao niz znakova, a ne brojčanu vrijednost. Stoga prije naredbe input, potrebno je odrediti tip podataka za rad s brojevima (**int, float**).

Primjeri unosa vrijednosti primjenom različitih tipova podataka:

```
>>> b=float(input('unesi realan broj'))
unesi realan broj3.14
>>> b
3.14
```

```
>>> a =int(input('unesi cijeli broj:'))
unesi cijeli broj:100
>>> a
100
```

```
>>> c =input('unesi ime:')
unesi ime:Ana
>>> c
'Ana'
```

Svaki tip podataka nužno zahtijeva pisanje zagrada.

```
>>> ime=input()
asd
>>> ime
'asd'
>>> a=input()
100
>>> a
'100'
```

u varijablu **ime** je spremljen niz znakova 'asd'

u varijablu **a** je spremljen niz znakova '100'

Naredba input bez teksta

U slučaju da želimo raditi sa znakovnim nizovima, onda nije potrebno naredbi **input** pridruživati tip podataka. Nakon tipa podataka (ukoliko za to postoji potreba), dodaje se ključna riječ **input.** Iznimno je bitno za naredbu **input** pisanje zagrada. Unutar tih zagrada, moguće je pisati tekst. Nije nužno pisati tekst, ali pomoću teksta jasnije možemo korisniku odrediti što se od njega traži: da li se traži unos brojčane vrijednosti ili znakovnog niza.

PRINT

Naredba za ispisivanje vrijednosti međutim ne predstavlja neko veliko odstupanje od sintakse naredbe za unos. Naredba za unos aktivira se ključnom riječi **print**. Nakon te riječi otvaraju se i zatvaraju zagrade. Unutar zagrada, moguće je ispisati tekst, ali i vrijednosti varijabli. **Glavna je razlika što se tekst uvijek stavlja unutar navodnika**. To je znak Pythonu da se radi o poruci koju korisnik želi ispisati. Vrijednost variajble se ispisuje na način da se ta variajbla pozove unutar naredbe **print** na način da se upiše njezino slovo na odgovarajuće mjesto. Sve tekstualne vrijednosti unutar naredbe print kao i vrijednosti varijabli, moguće je odvojiti zarezom „**,**“. Dakle naredba ispisa omogućuje:

- ispis vrijednosti na standardni izlaz (zaslon monitora)
- višestruke vrijednosti odvojene zarezom
- **print** sam dodaje razmak između višestrukih vrijednosti

Primjer naredbe print u Pythonu

```
>>> print('ovo je neki tekst','a ovdje je i broj',33)
ovo je neki tekst a ovdje je i broj 33
```

ZADACI

Zadatak 1:

Upisati jednu riječ i umnožiti je 5 puta

Primjer ispisa:

```
Unesite 1. rijec'python'
python python python python python
```

Rješenje:

```
a=input("Unesite 1. rijec")

c=(a+' ')*5

print(c)
```

U rješenju programa, primijenjen je razmak ' ' koji se označava jednostrukim navodnicima.

Zadatak 2:

Upišite dvije riječi i spojite ih u niz**.**

Primjer ispisa:

```
Unesite 1. rijec 'python'
Unesite 2. rijec' je super'
python  je super
>>>
```

Rješenje:

```
a=input("Unesite 1. rijec")
b=input("Unesite 2. rijec")

c=a+' '+b
print(c)
```

Zadatak 3:

- Unesite troznamenkasti broj
- Kod ispisa ispisati:
- Broj koji unosite
- Broj u binarnom brojevnom sustavu (baza 2)

Primjer ispisa:

```
====
Unesite troznamenkasti broj325
('Vas unos je', 325, 'Binarni broj je', '0b101000101')
>>>
```

Rješenje:

```
a=int(input("Unesite troznamenkasti broj"))

b=bin(a)

print("Vas unos je",a,"Binarni broj je", b)
```

Zadatak 4:

Unesite znak preko tipkovnice (slovo, brojku, specijani znak) i ispišite ASCII kod tog znaka

Primjer ispisa:

```
Unesite znak';'
('vase znak:', ';', 'ASCII kod', 59)
```

Rješenje:

```
a=input("Unesite znak")

b=ord(a)

print("vase znak:",a, "ASCII kod",b)
```

POMOĆ:

ord(a)

Kod ovog zadatka, primijenjena je posebna posebna naredba **ord()** pomoću koje se dohvaća vrijednost nekog znaka napisana u ASCII kodnom sustavu.

Zadatak 5:

Unesite vrijednost varijable i ispišite memorijsku adresu te varijable

Primjer ispisa:

```
Unesite vrijednost a:25
('mem. lokacija je:', 38368808)
```

POMOĆ:

id(a)

Rješenje:

```
a=int(input("Unesite vrijednost a:"))

print("mem. lokacija je:", id(a))
```

U rješenju je primijenjena posebna naredba **id()** kojom je moguće dohvatiti adresu memorijske lokacije gdje je trenutačno spremljen podatak u memoriji.

Zadatak 6:

Zbrojite dva broja i rezultat ispišite kao decimalni broj

Primjer ispisa:

```
====
Prvi broj 2.6
Drugi broj5.8
('Zbroj je:', 8.4)
```

Rješenje:

```
a=float(input("Prvi broj"))
b=float(input("Drugi broj"))
c = a+b
print("Zbroj je:", c)
```

Zadatak 7:

Unesite dva broja i izračunajte njihovu sumu, razliku, količnik, umnožak

Primjer ispisa:

```
Unesite 1. broj 2
Unesite 2. broj3
('Zbroj je:', 5)
('Razlika je :', -1)
('Umnozak je:', 6)
('Kolicinik je:', 0)
```

Rješenje:

```
a=int(input("Unesite 1. broj"))
b=int(input("Unesite 2. broj"))
c=a+b
d=a-b
e=a*b
f=a/b
print("Zbroj je:", c)
print("Razlika je :", d)
print("Umnozak je:",e)
print("Kolicinik je:",f)
```

Zadatak 8:

- Unesite dva broja i izračunajte njihovu **sumu, razliku, količnik, umnožak**
- Sumu ispisati u binarnom sustavu
- Razliku ispisati u heksadekadskom sustavu
- Umnožak ispisati u oktalnom sustavu
- Količnik ispisati u dekadskom sustavu
-

Primjer ispisa:

```
Unesite 1. broj 25
Unesite 2. broj5
('Zbroj je:', '0b11110')
('Razlika je :', '0x14')
('Umnozak je:', '0175')
('Kolicinik je:', 5)
```

Rješenje:

```
a=int(input("Unesite 1. broj"))
b=int(input("Unesite 2. broj"))

c=a+b
d=a-b
e=a*b
f=a/b

print("Zbroj je:", bin(c))
print("Razlika je :", hex(d))
print("Umnozak je:",oct(e))
print("Kolicinik je:",f)
```

U rješenju ovog programskog primjera, korištene su naredbe **bin(), hex(), oct()** kojima omogućavamo da se neki broj u dekadskom brojevnom sustavu pretvori u binarni, heksadekadski i oktalni brojevni sustav.

Zadatak 9:

Napisati program koji ispisuje rezultat na temelju izraza: $x=b^2-4ac$

Primjer ispisa:

```
1.broj2
2.broj3
3.broj4
('Rezultat', -23)
```

Potenciranje u Pythonu

a2**

Rješenje:

```
a=int(input("1.broj"))
b=int(input("2.broj"))
c=int(input("3.broj"))

x=b**2-4*a*c

print("Rezultat", x)
```

Zadatak 10:

- Izračunati aritmetičku sredinu sedam brojeva
- **NAPOMENA:** u jednoj liniji koda napraviti unos vrijednosti 7 varijabli

Primjer ispisa:

```
Unesi niz brojeva1,2,3,4,5,6,7
('aritm.sredina je:', 4)
```

Rješenje:

```
a,b,c,d,e,f,g=input("Unesi niz brojeva")

sredina=(a+b+c+d+e+f+g)/7

print("aritm.sredina je:", sredina)
```

U rješenju ovog programa, vidljivo je da nije naznačen brojčani tip podataka. Iako je naredba unosa napisana bez pripadajućeg tipa podataka, python je uspješno uspio izračunati aritmetičku sredinu. U istom retku, napravljen je upis u 7 različitih varijabli.

3. IF-ELSE ODLUKE

Odluke u Pythonu su realizirane u obliku grananja ili **if-else uvjetovanja.** One omogućavaju slijednom ili linijskom programu odabir jedne od dvije mogućnosti odnosno kretanje u jednom od odabranih smjerova izvršavanja naredbi. Tako na temelju ispitivanja uvjeta te ispunjenja njegovog logičkog testa, moguće je izvršavati jedan od dva ponuđena bloka naredbi. Grananje predstavlja binarnu operaciju **TRUE/FALSE** jer se temelje na odabiru jednog od dva ponuđena puta.

Primjer toga može biti ispitivanje odnosa brojeva pomoću operatora usporedbe nakon čega kao rezultat dobivamo jednu od dvije vrijednosti TRUE ili FALSE.

```
>>> 7>5
True
>>> 3!=3
False
>>> 3==3
True
>>> "Marko" > "Markić"
True
```

Primjer 1: Operatori usporedbe

Usporedbom dva broja dobili smo rezultat TRUE ili FALSE. Primjerice 7>5 znači da je broj sedam veći od pet i ta je tvrdnja sigurno točna. U tome slučaju odluka u programu se kreće u smjeru TRUE, a u suprotnom bi bila FALSE.

- ako je **uvjet** onda
  - Naredba 1
  - Naredba 2
- **inače**
  - Naredba 1
  - Naredba 2

Govorni jezik

```
if uvjet:
        Naredba1
        Naredba2
else:
        Naredba1
        Naredba2
```

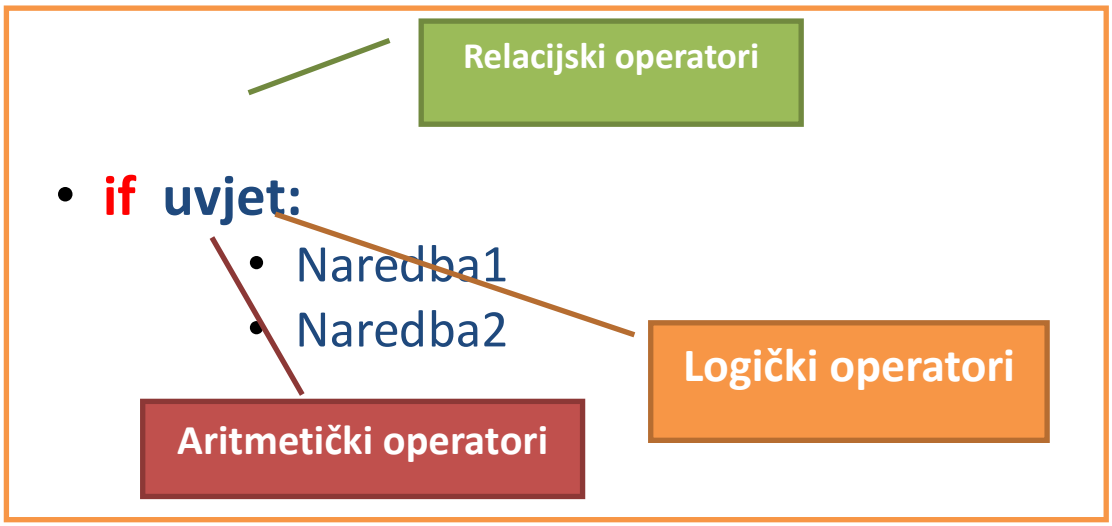
Python

Odluke u programiranju možemo prevesti u govorni jezik kao frazu: **ako je (uvjet) onda.....inače.** Ukoliko je uvjet ispunjen izvrši prvi blok naredbi u suprotnom prjeđi na blok naredbi nakon naredbe **inače.** U Pythonu naredba za odluke koristi se kao ključna riječ **IF (uvjet) ....ELSE....** Dakle ukoliko je uvjet ispunjen, izvršava se blok naredbi odmah ispod naredbe if, a ukoliko uvjet nije ispunjen, izvršava se blok naredbi ispod naredbe else. Izrazito je bitno napomenuti da odluka u pythonu koja se sastoji od ključne riječi **if** i **uvjeta** završava sa znakom dvotočke „**:**“ Taj znak označava kraj naredbe i prelazak u blok naredbi koje će se izvršiti nakon što uvjet bude ispunjen. Isto tako naredba else završava također sa znakom dvotočke „**:**“ Ona u oba slučaja označava kraj retka i prelazak u izvršavanje bloka naredbi.

U programskom jeziku Python to bi izgledalo ovako:

```
broj=25
if  broj % 7 == 0:
    print("Djeljiv sa sedam")
else:
    print("Nije djeljiv sa sedam")
```

Imamo varijablu **broj** s pridruženom vrijednosti **25**. Varijabla s vrijednosti ulazi na mjesto uvjeta u if odluci. Pitamo se da li je vrijednsot varijable 25 djeljiva sa 7. Ukoliko jest, a to je slučaj kada je rezultat = 0 onda ispiši „Djeljiv sa sedam“, u suprotnom ispiši „Nije djeljiv sa sedam.“



Na mjesto uvjeta u if-else uvjetovanja može biti uvršteno nekoliko različitih operatora: **relacijskih**, **aritmetičkih** i **logičkih**.

U nastavku su pobrojeni spomenuti operatori koji se koriste u uvjetu **if-a**

RELACIJSKI OPERATORI

Relacijski operator	
>	Veće
<	Manje
<=	Manje ili jednako
>=	Veće ili jednako
==	Jednako
!=	različito

```
n = int(input('Prvi broj:'))
m = int(input('Drugi broj:'))
if n>m:
    print('Veci je', n)
else:
    print('Veci je', m)
```

LOGIČKI OPERATORI

Logičke operacije	
and	**I**
or	**Ili**
not	**ne**

```
n = int(input('0 ili 1:'))
m = int(input('0 ili 1:'))
if n and m == True:
    print("rezultat je  ISTINA")
else:
    print("rezultat je LAZ")
```

ARITMETIČKI OPERATORI

Aritmetičke operacije	
+	Zbrajanje
-	Oduzimanje
*	Množenje
/	Dijeljenje
**	Potenciranje
%	Dijeljenje s ostatkom

```
n = int(input('Unesi broj:'))
if n / 2==0:
    print('paran')
else:
    print('neparan')
```

ZADACI

Zadatak 1:

Unesi neki prirodan broj i provjeri da li je taj broj **djeljiv sa sedam**

Primjer ispisa:

```
Unesite broj 25
('Broj', 25, 'nije djeljiv sa sedam')
```

Rješenje:

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

if a % 7==0:
    print("Broj", a, "je djeljiv sa sedam")
else:
    print("Broj", a, "nije djeljiv sa sedam")
```

U rješenju zadatka pod uvjetom, korištena je aritmetička operacija dijeljenja s ostatkom **%**. Uneseni broj 25 podijeljen je sa 7 i ostatak dijeljenja je uspoređen s nulom. Budući da je dijeljenje s brojem 25 dalo ostatak 3, taj ostatak je uspoređen s nulom. Budući da 3 nije jednako 0, program je odabrao mgućnost else i ispisao da broj nije djeljiv sa 7.

Zadatak 2:

- Omogućite unos realnog broja i ispitajte da li je broj **negativan ili pozitivan**.
- Ako je pozitivan ispisati pozitivan
- U suprotnom ispisati negativan

Primjer ispisa:

```
unesi broj: -0.32
negativan
```

Rješenje:

```
num = float(input("unesi broj: "))
if num > 0:
    print("pozitivan")
else:
    print("negativan")
```

Prije rješavanja zadatka, potrebno je predočiti si koji su brojevi pozitivni, a koji negativni pomoću brojevnog pravca. S lijeve strane brojevnog pravca se nalaze negativni brojevi, a s desne pozitivni brojevi. Prema tome određujemo **uvjet if.** U ovom programu je upotrijebljen operator **veće (>).** Dakle za svaki broj provjerimo da li je veći od 0. Ukoliko je veći, ispisuje se **pozitivan.** Mogli smo napraviti i obrnuto upotrebom operatora **manje (<).** Provjeravamo da li je svaki uneseni broj manji od nule. Ako jest, onda ispisujemo prvo **negativan.** Potrebno je znati da za rješavanje zadatka postoji više načina.

Zadatak 3:

Unesite dva broja. Zbrojite dva broja. Ako je zbroj veći od 20 ispišite : **Suma je veća od 20** u suprotnom ispišite: **Suma je manja od 20**

Primjer ispisa:

```
Unesite 1.broj12
Unesite 2.broj3
Suma je manja od 20
```

Rješenje:

```
a=int(input("Unesite 1.broj"))
b=int(input("Unesite 2.broj"))

c=a+b
if c >  20:
    print("Suma je veca od 20")
else:
    print("Suma je manja od 20")
```

Rješenje je vrlo jednostavno. Unosimo dva broja i zbrajamo ih. Ukoliko je zbroj veći do 20 ispisujemo odgovarajuću poruku. Naravno zbroj (sumu) smo mogli usporediti s bilo kojim drugim brojem. Primjerice umjesto 20 je mogao biti broj 100.

Zadatak 4:

- Unesite dva broja. Pomnožite ih i zbrojite. Ispišite umnožak i zbroj.
- Provjerite da li su zbroj i umnožak jednaki
- Ako su zbroj i umnožak jednaki ispisati: **umnozak i zbroj 2 broja su jednaki**
- U suprotnom ispisati:
- **Umnozak i zbroj 2 broja su razliciti**

Primjer ispisa:

```
Unesi 1. broj2
Unesi 2. broj6
('Umnozak_', 12)
('Zbroj_', 8)
Umnozak i zbroj dva broja  su razliciti
```

Rješenje:

```
a=int(input("Unesi 1. broj"))
b=int(input("Unesi 2. broj"))

c=a*b
d=a+b
print("Umnozak_",c)
print("Zbroj_",d)
if c==d:
    print("Umnozak i zbroj dva broja su jednaki")
else:
    print("Umnozak i zbroj dva broja  su razliciti")
```

Budući da je svrha odluka (uvjetovanja) upravo provjeravanje uvjeta te omogućavanje izvršavanja određenog bloka naredbi ukoliko su uvjeti ispunjeni, prije **if-else uvjetovanja** prethodi nekoliko naredbi i operacija. Na primejr u ovom zadatku smo unijeli dva broja i izračunali zbroj i umnožak. Pomoću **if-else uvjetovanja** provjerili smo jesu li zbroj i umnožak jednaki. Zbroj i umnožak su jednaki u slučaju kada zbrojimo ili pomnožimo brojeve 2 i 2 (2+2=4 i 2*2=4). Pomoću if-else smo provjerili jednu takvu situaciju.

Zadatak 5:

- Unesite 2 broja
- Ako su oba broja jednaka izračunati površinu kvadrata **(p=a**2)**
- U suprotnom izračunati površinu pravokutnika **(a*b)**

Primjer ispisa:

```
Unesite stranicu a: 16
Unesite stranicu b: 16
('Povrsina kvadrata iznosi: ', 256)
```

Rješenje:

```
a=input("Unesite stranicu a: ")
b=input("Unesite stranicu b: ")
p=0

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

Zadatak je vrlo sličan prethodnom zadatku. Po unesenim brojevima if-else uvjet treba usporediti da li su brojevi jednaki. Samo u tom slučaju moguće je izračunati površinu kvadrata, jer kvadrat kao geometrijski lik ima sve četiri stranice jednake duljine. Ovim binarnim uvjetom smo omogućili da se izvrši jedna od zadanih operacija što anravno ovisi o ispunjenju uvjeta.

Zadatak 6:

- Upišite neku riječ.
- Zatim provjerite ako se u toj riječi nalazi samoglasnik **a**
- Ako postoji, ispisati **samoglasnik se nalazi u napisanoj rijeci**, u suprotnom ispisati **nema samoglasnika**

NAPOMENA:
Znakove u programskom kodu označavamo s jednostrukim navodnicima 'a'.
Koristiti operator istovjetnosti **in**

Primjer ispisa:

```
Upisati rijec 'informatika'
('Samoglasnik a se nalazi u rijeci', 'informatika')
```

Rješenje:

```
a=input("Upisati rijec")

if 'a' in a:
    print("Samoglasnik a se nalazi u rijeci",a)
else:
    print("nema samoglasnika")
```

U zadatku je upotrijebljen operator istovjetnosti **in**. Pomoću njega provjerili smo postojanje znaka a u unesenoj riječi.

Zadatak 7:

- Unesite broj. Ako je unesen **broj 1** omogući operaciju **zbrajanja dva broja** i ispiši zbroj;
- Unosom bilo kojeg drugog broja omogući operaciju množenja dva broja
- i **ispiši umnožak**

Primjer ispisa:

```
Vas odabir 1
Unesite 1. broj2
Unesite 2. broj5
('Zbroj:', 7)
>>>
================
===
Vas odabir 3
Unesite 1. broj2
Unesite 2. broj5
('Umnozak:', 10)
```

Rješenje:

```
a=int(input("Vas odabir "))

if a==1:
    b=int(input("Unesite 1. broj"))
    c=int(input("Unesite 2. broj"))
    d=b+c
    print("Zbroj:",d)
else:
    b=int(input("Unesite 1. broj"))
    c=int(input("Unesite 2. broj"))
    d=b*c
    print("Umnozak:", d)
```

Primjetimo da se u dva bloka naredbi u tijelu **if** i **else** nalaze jednake varijable unosa i variajble spremanja rezultata operacija zbrajanja i množenja. Svaki blok naredbi je zavisan o sebi i ima svoju autonomiju. Pamti se samo varijabla napisana u trenutno odabranom bloku naredbi.

Zadatak 8:

- Omogućite dva unosa broja **0 ili 1**.
- Upotrijebite operaciju **logičkog I**
- Ako je rezultat **TRUE** ispišite **true**
- U suprotnom ispišite **FALSE**

Primjer ispisa:

```
Unesite 0 ili 1:1
Unesite 0 ili 1:0
False
```

Rješenje:

```
a=int(input("Unesite 0 ili 1"))
b=int(input("Unesite 0 ili 1"))


if a and b==True:

    print("True")
else:
    print("False")
```

U primjeru ovog jednostavnog zadatka koristili smo se logičkom operacijom **and**. Postavljanjem ovakvog uvjetovanja, možemo provjeriti cijelu tablicu istinitosti za **logički I.**

a	&	b	=	Izlaz
0		0		0
0		1		0
1		0		0
1		1		1

Zadatak 9:

- Unesite dva broja
- Ispitati sljedeći logički izraz
- **a>0 and b<a**
- Ako je prethodni izraz ispravan
- Ispisati **TRUE** u suprotnom **FALSE**

Primjer ispisa:

```
Unesite 1. broj12
Unesite 2. broj3
True
>>>
=============== RE
===
Unesite 1. broj3
Unesite 2. broj12
False
```

Rješenje:

```
a=int(input("Unesite 1. broj"))
b=int(input("Unesite 2. broj"))


if a>0 and b<a==True:

    print("True")
else:
    print("False")
```

Zadatak je bio vrlo jednostavan. Potrebno je ispitati određen logički uvjet. Koristili smo logički operator **and** i operatore usporedbe (**< i >**).

Zadatak 10:

- Unesite dva broja
- Ispitati sljedeći logički izraz
- **a>0 and b<a**
- Ako je prethodni izraz ispravan
- Ispisati **TRUE** te provjeriti sljedeći
- logički izraz **a>b or b>0**
- u suprotnom ispisati **FALSE**
- Ako je prethodni izraz ispravan
- Ispisati **TRUE**
- u suprotnom **FALSE**

Primjer ispisa:

```
Unesite 1. broj12
Unesite 2. broj3
a>0 and b<a=True
a>b or b>0=True
```

Rješenje:

```
a=int(input("Unesite 1. broj"))
b=int(input("Unesite 2. broj"))


if a>0 and b<a:
    print("a>0 and b<a=True")
    if a>b or b>0:
        print("a>b or b>0=True")
    else:
        print("False")

else:
    print("False")
```

Zadatak je vrlo sličan prethodnom zadatku, samo što se ovdje koristi **ugniježđeni if-else.** Ugnježđivanje znači da se nakon ispitanog jednog dijela uvjeta, prelazi na ispitivanje drugog dijela uvjeta. Ukoliko su oba uvjeta ispunjena, izvršava se određeni blok naredbi. Primjerice je ukoliko prvi dio uvjeta **a>0 and b<a** ispunjen, prelazi se na provjeru drugog dijela. Ukoliko prvi dio zadanog uvjeta nije ispunjen, odmah se ispisuje poruka o neuspjehu **(False).** Ovakvo uvjetovanej se koristi u slučaju da je potrebno ispitati više uvjeta prije izvršenja naredbi ili više dijelova jednog uvjeta kao u našem slučaju.

Zadatak 11:

Napisati program za množenje dva broja u rasponu zaključno s brojem 10. Na početku programa upitati korisnika da li želi množiti brojeve. Ukoliko odgovori pozitivno s **da** omogućiti unos i množenje. Ukoliko odgovori negativno, omogućiti izlazak iz programa.

Primjer ispisa:

```
Želite li unijeti brojeve da/neda
Unesite prvi broj2
Unesite drugi broj6
Umnozak brojeva je: 12
```

Rješenje:

```
a=input("Želite li unijeti brojeve da/ne")

if a=='da':
    b=int(input("Unesite prvi broj"))
    if b<=10:
        c=int(input("Unesite drugi broj"))
        if c<=10:
            d=b*c
            print("Umnozak brojeva je:",d)
        else:
          print("Broj nije u rasponu <=10")
    else:
        print("Broj nije u rasponu <=10")
else:
    print("Izlazak iz programa")
```

Za primijetiti je da u rješenju programa imamo nekoliko ugniježđenih **if-else uvjetovanja.** Prvi **if** ispituje uneseni odgovor korisnika. Ukoliko je odgovor pozitivan, omogućava unos brojeva. S dva dodatna **if-a** provjerava se da li je uneseni broj unutar zadanog raspona.

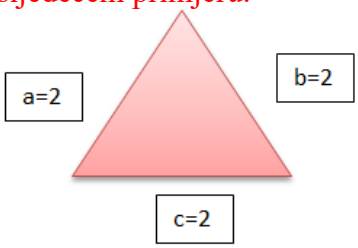
Ukoliko su brojevi unutar raspona, izvršava se operacija množenja. U suprotnom se prekida izvršavanje programa.

Zadatak 12:

- Napisati program koji će omogućiti korisniku unos stranica trokuta.
- Program redom ispisuje stranice trokuta redoslijedom kojim ih je korisnik unio.
- Program provjerava da li takav trokut postoji. Ako postoji onda
- se provjerava je li trokut **jednakostraničan, raznostraničan ili jednakokračan.**
- Nakon provjere, program ispisuje obavijest o postojanju takvog trokuta i vrsti trokuta (**jednakostraničan, raznostraničan ili jednakokračan.**).
- U suprotnom ispisuje da trokut ne postoji.

NAPOMENA:

Provjera postojanja geometrijskog lika trokuta. Ako je zbroj dviju pripadnih stranica veći od treće stranice, onda trokut postoji. Ukoliko je zbroj dvije stranice trokuta manji od treće znači trokut ne postoji. Ako je zbroj dviju stranica veći od treće onda trokut ima svoj vrh kao u sljedećem primjeru.



a+b>c => 2+2>2

U primjeru ispod zbroj dviju stranica je manji od treće što znači da trokut nema pripadajući vrh i može se pretvoriti u pravac.

c=6

a+b>c => 1+1<6

Primjer ispisa:

```
Unesi a3
Unesi b3
Unesi c2
Prva stranica: 3
Druga stranica: 3
Treca stranica: 2
Postoji ovakav trokut.
Jednakokracni.
```

Rješenje:

```
a=int(input("Unesi a"))
b=int(input("Unesi b"))
c=int(input("Unesi c"))

print("Prva stranica:",a)
print("Druga stranica:",b)
print("Treca stranica:",c)
if a+b>c and a+c>b and b+c>a:
    print("Postoji ovakav trokut.")
    if a==b  and b!=c:
        print("Jednakokracni.")
    if c==b and  b!=a:
        print("Jednakokracni.")
    if c==a and  a!=b:
        print("Jednakokracni.")

    if a==b and b==c:
        print("Jednakostranican")

    if a!=b and b!=c and a!=c :
        print("Raznostranican")
else:
    print("Ne postoji trokut s ovim stranicama")
```

Prva provjera se odnosi na provjeru da li unesene vrijednosti sačinajvaju trokut ili neki drugi geometrijski lik. Budući da trokut ima tri strane, potrebno je napraviti sve tri kombinacije provjere unutar istog uvjetovanja. Sljedeći uvjeti su naravno vezani za pojedini trokut. Kod jednakokračnog na primjer, potrebno je ispitati sve tri mogućnosti jer dvije stranice mogu biti jednake duljine u različitim položajima trokuta. Da bi se riješio zadatak, potrebno si je predočiti sve tri vrste trokuta i međusobno usporediti odnose stranica. Tek nakon toga, moguće je odrediti uvjete unutr **if-else uvjetovanja.** Za primijetiti je da se unutar glavnog uvjetovanja nalazi pet samostalnih **if-ova** bez upotrebe **else** dijela. Naravno i takav oblik postavljanja naredbi uvjetovanja je moguć. U programu je to izvedeno radi lakšeg pisanja i snalaženaj u kodu.

4. ELIF ODLUKE

Ukoliko postoji potreba za ispitivanjem višestrukih uvjeta, standardna struktura odluke if-else ne zadovoljava. To znači da ako želimo ispitati više uvjeta na standardni način if-else uvjetovanjem, morali bismo za više uvjeta ugnijezditi više if-else uvjetovanja. Tako bi programski kod bio vrlo nečitljiv i težak za pisanje i razumijevanje. Stoga standardno **if-else** uvjetovanje nadopunjujemo dodatnom strukturom **elif**.

```
if uvjet:
        Blok naredbi
        Blok naredbi
elif uvjet_1:
        Blok naredbi
        Blok naredbi
elif uvjet_2:
        Blok naredbi
        Blok naredbi
elif uvjet_N:
        Blok naredbi
        Blok naredbi
else:
        Blok naredbi
        Blok naredbi
```

U prethodnom primjeru **elif** konstrukcije, vidljivo je da uza svaku ključnu riječ **elif** postoji uvjet. Uvjeta može biti vrlo mnogo **(N)**, onoliko koliko sam programer odredi. Svaki **elif** ispituje svoj uvjet i ako je uvjet ispunjen, izvršava se blok naredbi smješten u tijelu **elif-a. Elif** uvjetovanje možemo zamisliti kao uzastopno postavljanje **if** uvjeta

. U govornom jeziku ovu **elf** konstrukciju bi čitali ovako:

```
ako uvjet:
        Blok naredbi
        Blok naredbi
inače-ako uvjet_1:
        Blok naredbi
        Blok naredbi
inače-ako uvjet_2:
        Blok naredbi
        Blok naredbi
inače-ako uvjet_N:
        Blok naredbi
        Blok naredbi
inače:
        Blok naredbi
        Blok naredbi
```

Dakle ukoliko je prvi uvjet ispunjen izvrši blok naredbi **inače** prijeđi na ispitivanje sljedećeg uvjeta; ako je uvjet ispunjen, izvrši blok naredbi inače prijeđi na ispitivanje sljedećeg uvjeta. Zato smo konstrukciju **elif** preveli kao **inače-ako** jer ona i jest kombinacija odluke **if (ako)** i **else (inače).**

ZADACI

Zadatak 1:

Napišite program koji će za uneseni broj ispisati da li je broj **pozitivan, nula** ili **negativan** koristeći **elif** uvjetovanje**.**

Primjer ispisa:

```
Unesite broj: 3
pozitivan broj
>>>
============== RES
===
Unesite broj: 0
nula
>>>
============== RES
===
Unesite broj: -34
negativan broj
```

Rješenje:

```
num = float(input("Unesite broj: "))
if num > 0:
    print("pozitivan broj")
elif num == 0:
    print("nula")
else:
    print("negativan broj")
```

Ideja rješenja zadatka je bila sljedeća. Rješenje možemo predočiti kao brojevni pravac. Brojevi s lijeve strane brojevnog pravca su negativni, a s desne strane pozitivni. Svi negativni brojevi na brojevnom pravcu su manji od nule, a pozitivni veći od nule. Na taj način bi postavili zadatak.Za rješenje zadatka, bilo je potrebno koristiti operatore usporedbe **(>, >=, <, <=, ==)**. Pomoću njih smo ideju brojevnog pravca preveli u programski kod. Pomoću uvjeta **elif** smo ispitali da li je uneseni broj veći do nule ili jednak nuli. Sa zaključnim **else** smo zaključili da su svi ostali brojevi koji ne zadovoljavaju prethodne uvjete pripadaju u kategoriju negativnih brojeva.

Zadatak 2:

Napravite program koji će za unos dva broja vršiti sljedeće operacije: **zbrajanje, oduzimanje, množenje i dijeljenje.**

Na početku je potrebno korisniku omogućiti unos dva broja, zatim odabrati operaciju odabirom znaka te operacije (**+,-,*,/**)

NAPOMENA:

Kod testiranja operaciju u programskom kodu unijeti s jednostrukim navodnicima '+'

Primjer ispisa:

```
Unesi prvi broj:12
Unesi drugi broj3
Unesi operaciju'+'
15
```

Rješenje:

```
n = int(input('Unesi prvi broj:'))
m = int(input('Unesi drugi broj'))
o = input ('Unesi operaciju')
if o == '+':
    print(n+m)
elif o == '-':
    print(n-m)
elif o == '*':
    print(n*m)
elif o == '/':
    print(n/m)
```

Kod postavljanja uvjeta u **if i elif**, bilo je potrebno znakove: **+, -, *, /** staviti pod navodnike jer jedino na taj način python zapis tumači kao znak, a ne varijablu.

Zadatak 3:

Program treba korisniku omogućiti unos broja bodova od **1-100**. Zatim program treba prema bodovnoj skali za svaki broj bodova ispisati o kojoj se ocjeni radi. Bodovne pragove je potrebno odrediti kao uvjete u **elif** uvjetovanju prema tablici.

Bodovi	Ocjene
<=**50**	Nedovoljan
<=**63**	Dovoljan
<=**76**	Dobar
<=**89**	Vrlo dobar
<=**100**	Odlican

Primjer ispisa:

```
Unesi broj bodova:46
nedovoljan
```

Rješenje:

```
p= int(input('Unesi broj bodova:'))
if p<=50:
    print('nedovoljan')
elif p<=63:
    print('dovoljan')
elif p<=76:
    print('dobar')
elif p<=89:
    print('vrlo dobar')
else:
    print('odlican')
```

Kod rješavanja zadatka treba obratiti posebnu pažnju na pažljiv odabir operatora usporedbe. Da je odabran samo operator **strogo manje (<),** uneseni broj ne bi bio uspoređen s graničnim brojem u uvjetu **elif.** Na primjer da je u uvjetu određeno **p<50** ispisivalo bi se „**nedovoljan“** samo ako su zadani brojevi manji od 50. Kod uvjeta **p<=50** „**nedovoljan“** će se ispisati i onda ako je uneseni broj jednak graničnoj vriejdnosti uvjeta **elfi.**

Zadatak 4:

Napišite program koji će omogućiti unos željenog broja. Uneseni broj treba podijeliti sa 3. Zatim ako je ostatak dijeljenja s brojem 3 :

- jednak nuli ispisati **„plava“,**
- jednak 1, ispisati **„crvena“,**
- jednak 2, ispisati **„zelena“,**
- jednak 3, ispisati **„ljubicasta“,**
- u suprotnom ispisati **„zuta“**

NAPOMENA:

Za dijeljenje s ostatkom upotrijebiti operator dijeljenja s ostatkom. **%**

Primjer ispisa:

```
Unesite broj:34
crvena
```

Rješenje:

```
n = int(input('Unesite broj:'))
if n % 3 == 0:
    print ('plava')
elif n % 3 == 1:
    print ('crvena')
elif n % 3 == 2:
    print ('zelena')
elif n % 3 == 3:
    print ('ljubicasta')
else:
    print ('zuta')
```

Zadatak je bio prilično jednostavan. Bilo je potrebno samo zadani broj podijeliti s 3 pomoću dijeljenja s ostatkom. Kod dijeljenja s ostatkom, ostatak dobiven pri dijeljenju se uspoređuje sa zadanim brojem. Na primjer ostatak dijeljenja broja 34 sa 3 je 1. Ta vrijednost 1 je uspoređena sa zadanim brojem 1. Uvjet je ispunjen, pa je moguće izvršiti naredbu **print ('crvena').**

Zadatak 5:

Napisati program koji će od korisnika tražiti unos broja u intervalu **od 1-12**. Kada korisnik odabere broj, program treba riječima ispisati naziv mjeseca koji odgovara broju unutar **intervala 1-12**. Na primjer unosom **broja 10** ispisati naziv mjeseca **listopad**.

Primjer ispisa:

```
Mjesec:5
svibnanj
```

Rješenje:

```
m = int (input('Mjesec:'))
if m == 1:
    mm = 'sijecanj'
elif m == 2:
    mm = 'veljaca'
elif m == 3:
    mm = 'ozujak'
elif m == 4:
    mm = 'travanj'
elif m == 5:
    mm = 'svibnanj'
elif m == 6:
    mm = 'lipanj'
elif m == 7:
    mm = 'srpanj'
elif m == 8:
    mm = 'kolovoz'
elif m == 9:
    mm = 'rujan'
elif m == 10:
    mm = 'listopad'
elif m == 11:
    mm = 'studeni'
elif m == 12:
    mm = 'prosinac'

print (mm)
```

Primjer zadatka je vrlo sličan prethodnom zadatku. Pomoću višestrukih **elif** uvjeta ispitali smo da li je uneseni broj jednak nekom broju u intervalu od 1-12. Imamo točno 12 uvjeta koliko i mjeseca. Ukoliko uneseni broj odgovara rednom broju mjeseca u godini, ispisuje se naziv tog mjeseca riječima.

Zadatak 6:

Napisati program koji će omogućiti unos tri stranice trokuta.

- Ako su sve tri stranice jednake ispisati: **jednakostranican**,
- ako su dvije stranice jednake duljine, ispisati: **jednakokracan**,
- a u suprotnom **raznostranican.**

NAPOMENA:

Koristiti se logičkim operacijama **and** i **or**.

Trokut je jednakostranican kada:

a == b and a==c

Trokut je jednakokracan kada:

a == b or a==c or b==c

Primjer ispisa:

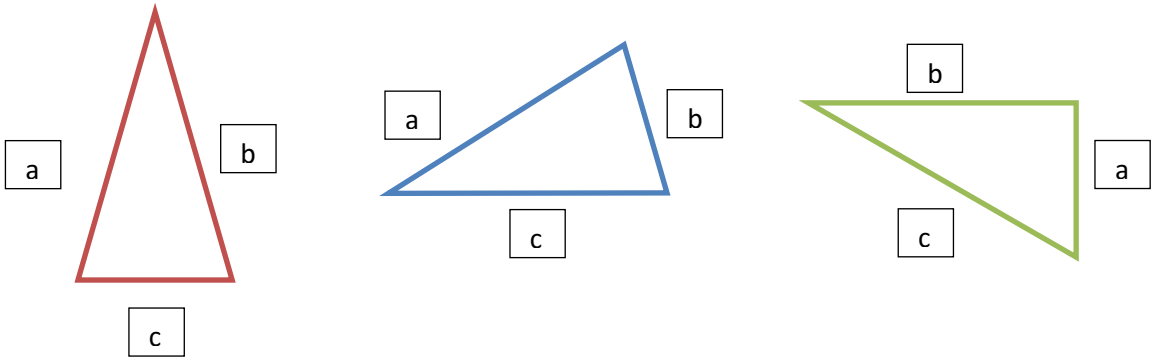
```
a: 2
b: 2
c: 2
Jednakostranican
```

Rješenje:

```
a = int(input('a: '))
b = int(input('b: '))
c = int(input('c: '))
if a == b and a==c:
    print ('Jednakostranican')
elif a == b or a==c or b==c:
    print ('Jednakokracan')
else:
    print('Raznostranican')
```

Prije rješavanja zadatka ove vrste, potrebno je poznavati temelje matematičke teorije. Treba si predočiti stranice trokuta. Iz slike jednakostraničnog trokuta, vidljivo je da su sve tri stranice međusobno jednake. Kod jednakokračnog trokuta treba voditi računa da su prema pravilu jednakokračnog trokuta dvije stranice uvijek jednake duljine.

U programu smo taj uvjet postavili **a == b or a==c or b==c.** Na taj način smo osigurali da neovisno o položaju trokuta pronađemo dvije stranice koje su jednake duljine.



Položaji Jednakokračnog trokuta

Zadatak 7:

Omogućiti korisniku da odabere jednu od **4 aritmetičke operacije (+,-,*,/).**

Nakon što korisnik odabere jednu od operacija, omogućiti **unos dva broja** nad kojima će se operacija izvršiti. Nakon toga **ispisati rezultat operacije**.

Primjer ispisa:

```
1.mnozenje||2.dijeljenje||3.zbrajanje||4.oduzimanje:3
Unesite prvi broj12
Unesite drugi broj4
('Zbroj=', 16)
```

Rješenje:

```
odabir = int(input('1.mnozenje||2.dijeljenje||3.zbrajanje||4.oduzimanje:'))

if odabir == 1:
    a=int(input("Unesite prvi broj"))
    b=int(input("Unesite drugi broj"))
    print("Umnozak=",a*b)
elif odabir == 2:
    a=int(input("Unesite prvi broj"))
    b=int(input("Unesite drugi broj"))
    print("Kolicnik=",a/b)
elif odabir == 3:
    a=int(input("Unesite prvi broj"))
    b=int(input("Unesite drugi broj"))
    print("Zbroj=",a+b)
elif odabir == 4:
    a=int(input("Unesite prvi broj"))
    b=int(input("Unesite drugi broj"))
    print("Razlika=",a-b)
```

U ovom zadatku smo napravili manji izbornik s brojevima operacija. Unutar **if** i svakog **elif** uvjetovanja, omogućili smo unos dva broja i izvršavanje odgovarajuće operacije te ispis rezultata. Primjetimo da se unutar različitih elif uvjetovanja nalaze iste varijable za unos brojeva **(a i b)**. To znači da svaki blok naredbi unutar **elif** uvjetovanja ima svoju autonomiju. Dakle vrijednost unesenih varijabli se pamti samo unutar bloka naredbi pojedinog **elif-a.**

Zadatak 8:

Omogućiti korisniku unos ocjena od **1-5**. Ukoliko korisnik odabere prolaznu ocjenu (2,3,4,5), ispisati **„ocjena ispita je (ocjena riječima), prosli ste**“; ako je ocjena negativna ispisati: **„ocjena ispita je nedovoljan, niste prosli“** . Ukoliko korisnik odabere broj koji nije u rasponu ocjena, ispisati: **„Nepostojeca ocjena“**

Primjer ispisa:

```
Unesite ocjenu ispita (1-5): 3
ocjena ispita je dobar, prosli ste
```

Rješenje:

```
ocjena = int(input("Unesite ocjenu ispita (1-5): "))


if ocjena == 1:
    print("ocjena ispita je nedovoljan, niste prosli")
elif ocjena == 2:
    print ("ocjena ispita je dovoljan, prosli ste")
elif ocjena == 3:
    print ("ocjena ispita je dobar, prosli ste")
elif ocjena == 4:
    print ("ocjena ispita je vrlo dobar, prosli ste")
elif ocjena == 5:
    print ("ocjena ispita je izvrstan, prosli ste")
else:
    print ("Nepostojeca ocjena")
```

Izrada programa je vrlo jednostavna. Program je sličan programu s ispisom naziva mjeseca. Ovdje umjesto mjeseca se ispisuju nazivi ocjena riječima, svaki puta kada korisnik unese ocjenu u rasponu od **1-5.**

Zadatak 9:

Korisniku omoguciti da na temelju upisanih sati sazna koje je doba dana.

Intervali su sljedeći:

<10	**Jutro**
<12	**Podne**
<18	**Predvecerje**
<22	**Vecer**
	Noc

Primjer ispisa:

```
Unesite broj sati i saznajte doba dana?22
noc
```

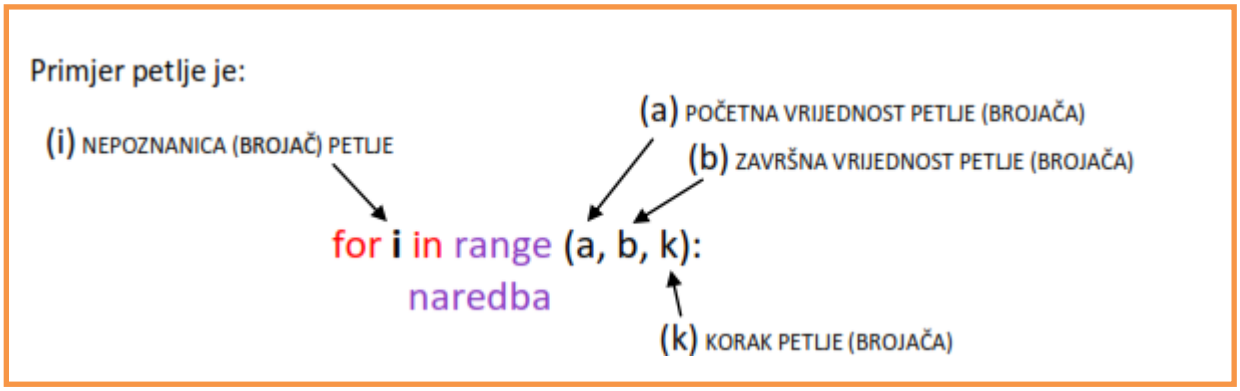
Rješenje:

```
x = int(input("Unesite broj sati i saznajte doba dana?"))

if x < 10:
 print ("Jutro")

elif x<12:
 print "Podne"

elif x<18:
  print ("Predvecerje")

elif x<22:
 print ("vecer")

else:
  print ("noc")
```

Zadatak s intervalima sličan je zadatku s određivanjem ocjene na temelju intervala s bodovima. Potrebno je koristiti operator usporedbe i svaki uneseni broj usporediti s graničnom vrijednosti u uvjetu **elif.**

5.PROGRAMSKA PETLJA FOR

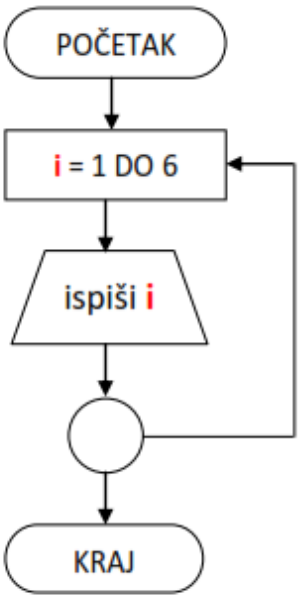
Programske petlje su programske strukture koji omogućavaju višestruko ponavljanje određenog dijela programskog koda. Petlja kao i programski konstrukt grananja ili odluke (if-else) sadržavaju uvjet. Ispunjenjem uvjeta određeni dio koda, izvršava se zadani broj puta. Osim toga petlja sadržava i brojač. Brojač je nešto poput uvjeta, gdje zadajemo broj ponavljanja nekog programskog koda. Ključna riječ za aktivaciju petlje je naredba **for.**



Uz nju dolazi oznaka **i** koja označava brojač. Brojač petlje zadaje broj ponavljanja određenog koda. Druga ključna riječ u petlji je **range** pomoću koje određujemo uvjet ponavljanja određenog koda. Unutar **range** nalazi se **početna vrijednost brojača** od koje petlja kreće ponavljati određeni kod, **zatim završna vrijednost petlje** te **korak petlje** kojim kažemo brojaču u kojme smjeru treba brojati i ponavljati petlju. Primjerice može se zadati ispis svakog drugog broja kojeg je korisnik unio.Ili se primjerice može zadati ispis negativnog broja. Završetak deklaracije označavamo sa znakom „**:**“ kao kod programskog konstrukta odluke.

Primjer programa:

Napišite program kojim ćete koristeći FOR petlju ispisati vrijednost brojača od 1 do 5.

DIJAGRAM TIJEKA	OBJAŠNJENJE
	Određujemo početnu i završnu vrijednost brojača. Ispiši svaku trenutačnu vrijednost brojača. Ponavljaj radnju dok dođeš do kraja brojača.

Rješenje programa bi izgledalo ovako:

PROGRAM	REZULTAT
for i in range (1,6): print (i)	1 2 3 4 5

- Brojač smo u ovom primjeru označili varijablom **i**
- Odredili smo početnu i završnu vrijednost brojača.
- U zadnjem prolazu vrijednost brojača je uvijek za 1 manja.
- Radnja će se ponavljati sve dok brojač ne bude imao vrijednost 6
- U svakom prolazu ispisat će se vrijednost brojača, osim u zadnjem jer se petlja zaustavlja.

ZADACI

Zadatak1:

Ispišite ponavljanje rečenice „Unosimo policu broj“ deset puta.

Primjer testiranja:

```
====
('Unosimo policu broj', 1)
('Unosimo policu broj', 2)
('Unosimo policu broj', 3)
('Unosimo policu broj', 4)
('Unosimo policu broj', 5)
('Unosimo policu broj', 6)
('Unosimo policu broj', 7)
('Unosimo policu broj', 8)
('Unosimo policu broj', 9)
('Unosimo policu broj', 10)
```

Rješenje:

```
for   i   in range (1,11):
          print ('Unosimo policu broj', i)
```

Na mjesto uvjeta u for petlji, odredili smo početnu (1) i završnu vrijednost (11).Petlja je ispisala 10 puta rečenicu „Unosimo policu broj“ te broj od 1 do 11 točnije prvih 10 brojeva. Vrijednost brojača unutar petlje ne pamti završnnu vrijednost. Da bi dobili završnu vrijednost, parametar završne vrijednosti trebamo postaviti na n+1. Da je primjerice stavljeno u petlji od 0 do 10, program bi ispisao brojeve: 0,1,2,3,4,5,6,7,8,9. Odnosno prvih 10 znamenki brojeći od nula. Dakle for petlja u Pythonu počinje brojati od zadane početne vrijednosti, ali ne ubraja završnu vrijednost zadanu u petlji.

Primjer:

```
('Unosimo policu broj', 0)
('Unosimo policu broj', 1)
('Unosimo policu broj', 2)
('Unosimo policu broj', 3)
('Unosimo policu broj', 4)
('Unosimo policu broj', 5)
('Unosimo policu broj', 6)
('Unosimo policu broj', 7)
('Unosimo policu broj', 8)
('Unosimo policu broj', 9)
```

Zadatak 2:

Ispisati samo parne brojeve brojevnog niza od prvih 10 brojeva.

Primjer testiranja:

```
0
2
4
6
8
>>>
```

Rješenje:

```
for   i   in range (0,10,2):
            print (i)
```

Kod ovog rješenja, dodano je uz raspon početne i završne vrijednosti i broj koraka. Broj koraka ovdje je 2. To znači da će program ispisivati svaki drugi broj odnosno u ovom slučaju parne brojeve.

Zadatak 3:

Ispiši prvih šest brojeva u obrnutom nizu. Npr. 1,2,3,4,5,6 ispisati kao 6,5,4,3,2,1.

Primjer testiranja:

```
6
5
4
3
2
1
>>>
```

Rješenje:

```
for   i   in range (6,0,-1):
            print (i)
```

Da bi dobili ispis obrnutog niza brojeva, moralo se najprije postaviti početna vrijednost na broj 6 to jest na broj kojim završava niz. Zatim kao završnu vrijednost postaviti broj 1 ili broj s kojim niz započinje. A broj koraka se zatim postavi na -1 čime se pythonu zapovijeda da treba krenuti u suprotnom smjeru.

Zadatak4:

Unesite prvih 10 brojeva, ali izuzmite broj 7.

Primjer testiranja:

```
1
2
3
4
5
6
8
9
10
>>>
```

Rješenje:

```
for   i   in range (1,11):
            if i != 7:
                    print (i)
```

Da bi omogućili ispisivanje brojeva u nizu, koristili smo for petlju sa zadanom početnom i završnom vrijednosti. Budući da smo ispisali niz brojeva bez broja 7 u njemu, bilo je potrebno kombinirati više programskih struktura poput odluka i petlje. Upravo je to napravljeno u ovom zadatku.

Zadatak 5:

Napišite program koji ispisuje zbroj parnih brojeva od 1 do 20

Primjer testiranja:

```
('Nakon broja', 0, 'zbroj je', 0)
('Nakon broja', 2, 'zbroj je', 2)
('Nakon broja', 4, 'zbroj je', 6)
('Nakon broja', 6, 'zbroj je', 12)
('Nakon broja', 8, 'zbroj je', 20)
('Nakon broja', 10, 'zbroj je', 30)
('Nakon broja', 12, 'zbroj je', 42)
('Nakon broja', 14, 'zbroj je', 56)
('Nakon broja', 16, 'zbroj je', 72)
('Nakon broja', 18, 'zbroj je', 90)
('Nakon broja', 20, 'zbroj je', 110)
```

Rješenje:

```
zbroj = 0
for   i   in range (0, 21, 2):
            zbroj = zbroj + i
            print ('Nakon broja', i ,'zbroj je', zbroj)
```

U for petlji je određen raspon brojeva od 0-21. Za ispis parnih brojeva, određen je korak na broj 2 što znači da se ispisuje svaki parni broj to jest svaki drugi u nizu brojeva. Poslije svake nove vrijednosti brojača, zbroj se povećava za vrijednost brojača (**zbroj = zbroj+i**).

Zadatak 6:

Pomoću petlji istovremeno ispisati dvije riječi „Python“ i „je super“ tri puta.

Primjer testiranja:

```
python
python
python
je super
je super
je super
```

Rješenje:

```
for   i   in range (0,3):
            print ('python')
for   j   in range (0,3):
            print ('je super')
```

Paralelno su pisane dvije petlje s istim uvjetom. Budući da se program ispisuje slijedno od vrha prema dnu, prvo se ispisala riječ „python“,a zatim riječ „je super“.

Zadatak 7:

Unesi neki broj i ispiši sve brojeve do tog broja počevši od 1.

Primjer testiranja:

```
unesi broj5
1
2
3
4
5
```

Rješenje:

```
n = int(input('unesi broj'))
for i in range(1, n + 1):
    print(i)
```

U programu je omogućen unos vrijednosti preko tipkovnice. Brojevi se ispisuju od poečtne vrijednosti 1 do broja koji smo zadali unosom.

Zadatak 8:

Unesi neki broj. Napravi ispis svih parnih brojeva do broja koji ste unijeli.

Primjer testiranja:

```
unesi broj9
2
4
6
8
```

Rješenje:

```
n = int(input('unesi broj'))
for i in range(2, n+1, 2):
    print (i)
```

Zadatak 9:

Unesite neki broj i napravite ispis svih brojeva do tog broja u obrnutom redoslijedu.

Primjer testiranja:

```
unesi broj:8
8
7
6
5
4
3
2
1
```

Rješenje:

```
n = int(input('unesi broj:'))
for i in range(n, 0, -1):
    print (i)
```

Kao početna vrijednost u for petlji je stavljeno unos korisnika (n), zatim kao krajnja vrijednost je stavljena nula, a kao korak -1 čime je naznačeno da se petlja kreće u obrnutom redoslijedu.

Zadatak 10:

Omogućite proizvoljan unos brojeva. Zatim unesite te proizvoljne brojeve i ispitajte koliko ih ima parnih, a koliko neparnih.

Primjer testiranja:

```
Unesi broj brojeva:2
Unesi broj:12
Unesi broj:3
('Zbroj parnih:', 12, 'Zbroj neparnih:', 3)
```

Rješenje:

```
n = int (input('Unesi broj brojeva:'))
z_p = 0
z_n= 0
for i in range(n):
    t = int(input('Unesi broj:'))
    if t % 2:
        z_n = z_n+t
    else:
        z_p =z_p+t
print ('Zbroj parnih:',z_p,'Zbroj neparnih:', z_n)
```

Najprije je napravljen unos broja broejva. To jest korisnika se pita koliko brojeva želi unijeti. Nakon toga omogućava mu se unos brojeva. Na primjer ako je napisao da će unijeti dva broja, onda mu je omogućeno da unosi samo dva broja. Nakon toga pomoću if-else strukture, ispituje se koji je broj paran, a koji neparan. Koristi se mkodularno dijeljenje s ostatkom **%.** Poslije svake nove vrijednosti brojača, zbroj se povećava za vrijednost brojača (z_n=z_n+t).

Zadatak 11:

- Napisati program koji će omogućiti unos **prirodnog broja n** i ispisati djelitelje tog broja **n**

Primjer testiranja:

```
Unesi prirodan broj32
1 2 4 8 16 32
...
```

Rješenje:

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

for i in range(1,n+1):

    if n%i==0:

        print(i, end=' ')
```

Zadatak 12:

- Napisati program koji simulira bacanje novčića **n puta** te ispisati koliko puta se pojavljuje **pismo** i **glava.**

Primjer testiranja:

```
Unesite željeni  broj 25
Vjerojatnost pismo 0.64
Vjerojatnost glava 0.36
```

Rješenje:

```
from random  import randint

n=int(input("Unesite željeni  broj "))
br_pismo=0
br_glava=0

for i in range(n):

    bacanje=randint(0,1)

    if bacanje == 0:
        br_pismo = br_pismo + 1
    else:
        br_glava = br_glava + 1

vjer_pismo=br_pismo/n
vjer_glava=br_glava/n

print ("Vjerojatnost pismo", vjer_pismo)

print ("Vjerojatnost glava", vjer_glava)
```

U ovom rješenju je uključena biblioteka **randint** za generiranje nasumičnih brojeva. Ovdje je generator iskorišten za nasumično bacanje novčića (pismo/glava). Pismo i glava su zamijenjeni binarnim stanjima 0 i 1. Imamo i dvije varijable **br_pismo** i **br_glava** u koje se smješta vrijednost kada padne pismo odnosno glava kod bacanja novčića. Varijabla **br_pismo=br_pismo+1** je izvedena na ovakav način zbog sljedećeg razloga. Postojeći sadržaj variajble **br_pismo** uvećavamo svaki puta za jedan. U prvom krugu petlje vrijednost je postavljena na 0, u drugom krugu se uvećava za 1 i tako do završetka petlje. Na taj način svaki puta kada padne pismo sadržaj postojeće vrijable povećavamo za 1.

Zadatak 13:

- Nastavnik ispituje učenike na satu
- Ispitivati će svakog **n-tog** učenika u imeniku
- Napisati program koji će omogućiti unos **broja učenika** te prirodan broj **n** kao **kriterij** odabira učenika iz imenika
- Program treba ispisivati redne brojeve učenika koje će nastavnik ispisivati.

Primjer testiranja:

```
Unesite broj učenika20
Unesite prirodan broj3
Ispitivati će se učenici s rednim brojevima: 1
Ispitivati će se učenici s rednim brojevima: 4
Ispitivati će se učenici s rednim brojevima: 7
Ispitivati će se učenici s rednim brojevima: 10
Ispitivati će se učenici s rednim brojevima: 13
Ispitivati će se učenici s rednim brojevima: 16
Ispitivati će se učenici s rednim brojevima: 19
```

Rješenje:

```
uc=int(input("Unesite broj učenika"))
n=int(input("Unesite prirodan broj"))

for i in range(1,uc+1, n):

    print("Ispitivati će se učenici s rednim brojevima:", i)
```

Ovdje po prvi puta rabimo manipulaciju parametrima petlje. Primjerice koristili smo varijablu za određivanje broja učenika odnosno završne vrijednosti petlje te variajblu za određivanje koraka petlje kada smo unosili neki kriterij po kojem ćemo tražiti učenike. U našem primjeru kriterij je 3 što znači da se ispisivao svaki treći učenik po rednom broju imenika.

Zadatak 14:

- Napišite program koji će **unositi broj učenika** te omogućiti **unos visine učenika** prema broju unesenih učenika.
- Program treba ispisati visinu **najnižeg** i **najvišeg** učenika

Primjer testiranja:

```
Koliko učenika želite unijeti: 2
Unesite visinu ucenika: 35
Unesite visinu ucenika: 65
Najniži: 35 cm
Najviši: 65 cm
```

Rješenje:

```
najnizi=0
najvisi=0

n=int(input("Koliko učenika želite unijeti: "))

for i in range(0,n):

    u=int(input("Unesite visinu ucenika: "))

    if i==0:
        najnizi=najvisi=u
    if najnizi>u:
        najnizi=u
    if najvisi<u:
        najvisi=u

print ("Najniži:", najnizi, "cm")
print ("Najviši:", najvisi, "cm")
```

U for petlji smo koristili variajblu za određivanje gornje granice petlje. Ostalo je riješeno pomoću tri uzastopna uvjetovanja. Primijetimo da smo varijable **najnizi** i **najvisi** napunili sa unesenom vrijednošću od strane korisnika. Dakle u prvom krugu petlje obje varijable (**najnizi i najvisi**) se pune vrijednošću koju korisnik prvu unosi. Svaku iduću unesenu vrijednost uspoređujemo s trenutnom vrijednosti u varijablama **najvisi i najnizi**. Ako je nova vrijednost veća/manja od trenutnih vrijednosti varijabli **najvisi/ najnizi** onda se njihov sadržaj zamjenjuje. U programu se koriste tri uvjeta. Prvi je uvjet potreban za postavljanje obje varijable na početnu vrijednost koju korisnik unosi. Ostala dva uvjetovanja ispituju odnose novounesenih vrijednosti.

Zadatak 15:

- Napisati program za učitavanje temperature zraka u prethodnih 10 dana te ispisati **srednju temperaturu iznad nule** i **srednju temperaturu ispod nule**.

Primjer testiranja:

```
Unesi broj 20
Unesi broj 3
Unesi broj 1
Unesi broj -2
Unesi broj 0
Unesi broj 1
Unesi broj 3
Unesi broj 2
Unesi broj 5
Unesi broj 6
 Srednja temperatura iznad nule  5.125
 Srednja temperatura ispod nule  -2.0
```

Rješenje:

```
brp=0
sp=0
brn=0
sn=0

for i in range (0,10):
    unos=int(input("Unesi broj "))
    if unos > 0:
        sp=sp+unos
        brp=brp+1
    if unos < 0:
        sn=sn+unos
        brn=brn+1

arp=sp/brp
arn=sn/brn

print(" Srednja temperatura iznad nule ", arp)
print(" Srednja temperatura ispod nule ", arn)
```

For petlja u ovome programu služi za ponavljanje određenog broja puta bloka naredbi unosa i provjere zadanih uvjeta. Vrijednosti su podijeljene na pozitivne i negativne pomoću dva uvjeta **if** uvjetovanja. Budući da formula za računanje srednje vrijednosti glasi **ukupan broj / broj unesenih vrijednosti,** u tijelu oba if uvjeta treba pobrojati sve svrijednosti i izračunati zbrojeve pozitivnih i negativnih. Varijabla **sp=sp+unos** je se može protumačiti kao sadržaj trenutne vrijednosti uvećan za novounešenu vrijednost varijable **unos.** Kdo pobrojavanja je dovoljno postojeći sadržaj varijable uvećati samo za jedan kao na primjer **brp=brp+1.**

Zadatak 16:

- Napisati program koji će generirati tablicu množenja brojeva do 10

Primjer testiranja:

```
 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
```

Rješenje:

```
for red in range(1,11):
    for stupac in range(1,11):
        print("{0:5}".format(red*stupac), end="")
    print()
```

Da bismo dobili tablicu množenja do 10, potrebno je koristiti dvije ugniježđene for petlje. Jedna petlja će generirati retke, a druga stupce. Petlja koaj ispisuje redak, postavlja se na prvi redak odnosno prvi broj u početnoj vrijednosti će pomnožiti sa svakim brojem petlje stupca.

1. redak * 1 stupac | 1.redak *2. stupac | 1.redak * 3.stupac .....

.

.

.

10.redak * 1. Stupac | 10.redak * 2. stupac | 10.redak * 3. stupac...

Množenje retka i stupca postižemo naredbom u ispisu **print (red*stupac). Naredbom „{0:5}“** postiže se adekvatan razmak između redaka i stupaca od pet praznih mjesta.

6.WHILE PETLJA

While petlja je programska struktura koja ima istu namjenu kao i for petlja.To znači da while petlja isto omogućava ponavljanje određenog bloka naredbi. Naravno razika ove petlje i for petlje je u tome što while petlja ne sadržava **brojač** kojeg sadržava for petlja. Brojač for petlji omogućava određivanje određenog broja ponavljanja nekog bloka naredbi. While petlja nema brojač, ali kao i for petlja ima uvjet. Taj uvjet jo omogućava da se izvršavaju naredbe. **Tako dugo dok je uvjet ispunjen ponavljaj blok naredbi**. Ako uvjet nije ispunjen, nemoj ponavljati taj blok naredbi. Tako bismo mogli prevesti funkcionalnost while petlje.

While uvjet**:**

- Blok naredbi1
- Blok naredbi 2
- .....

Python deklaracija

Dok je uvjet ispunjen:

- Izvrši naredbu jedan
- Izvrši naredbu dva.....

Govorni jezik

While petlja je vrlo jednostavna petlja koja samo omogućava ponavljanje kada je određeni uvjet ispunjen. Uvjet u while petlji može biti određen operatorima: **usporedbe, logičkim operatorima** i **aritmetičkim operatorima**.

ZADACI

Zadatak 1:

Napisati program koji će korisniku dati mogućnost pogađanja lozinke. Mogućnost pogađanja lozinke izvršavati će se toliko puta dok korisnik ne pogodi lozinku. Kada korisnik pogodi lozinku, ispisati „**upisali ste tocnu lozinku“** u suprotnom javiti vijest o pogrešno upisanoj lozinci i ponoviti unos.

Primjer ispisa:

```
Unesite lozinku: 'tajno'
Upisali ste  tocnu lozinku
```

Rješenje:

```
password = ""
while password != "tajno":
    password = input("Unesite lozinku: ")
    if password == "tajno":
        print("Upisali ste  tocnu lozinku")
    else:
        print("upisali ste pogresnu lozinku-pokusajte ponovno")
```

U rješenju je na mjestu **while uvjeta** korišten operator usporedbe **!=**. Program je ponavljao „**unesite lozinku“** tako dugo dok uvjet u while petlji nije bio ispunjen točnije dok **password==“tajno“** odnosno dok **password** više nije bio **!=“tajno“.**

Zadatak 2:

Napraviti program koji će korisniku omogućiti pogađanje brojeva. Ako korisnik upiše bilo koji broj, izvršavati će se blok naredbi ispod while petlje. Ako odabere 0, dogodi se prekid programa. Prekid programa omogućiti sa naredbom **break.** Ako korisnik upiše točan broj, ispiše se poruka o pogođenom broju i program se dalje izvršava. Ako korisnik napiše **prevelik ili premali broj od traženog**, ispisati prigodnu poruku korisniku i dalje izvršavati program, sve dok korisnik sma ne odabere opciju 0.

Primjer ispisa:

```
Igra pogadjanja brojeva
Za prekid igre unesite 0
Unesite broj: 17
Pogodili ste zadani broj cestitamo!
Unesite broj:
```

Rješenje:

```
print ("Igra pogadjanja brojeva")
print ("Za prekid igre unesite 0")

pogodi=17

while 1:
    broj=int(input("Unesite broj: "))
    if broj==pogodi:
        print ("Pogodili ste zadani broj cestitamo!")
        broj=0
    elif broj==0:
        break
    elif broj > pogodi:
        print ("Broj je veci od zadanog, pokusajte ponovo")
    elif broj < pogodi:
        print ("Broj je manji od zadanog, pokusajte ponovo")
|
```

Varijabla fiksnog broja je postavljena na 17. To je broj kojeg je korisnik trebao pogoditi. U **while petlji** uvjet se postavio na 1. U svijetu logičkih operatora to znači **true** odnosno stanje u kojem postoji vrijednost. Taj 1 u uvjetu znači da ako je unesemo bilo koji broj koji nije nula, izvršavaju se naredbe u tijelu petlje while. Kada korisnik odabere 0 to je kao da je odabrao **false** odnosno stanje bez vrijednosti; program se prekida. Prekid programa, omogućen je pomoću naredbe prekida **break**. Naredba **break** ima samo jednu funkciju: prekinuti izvođenje programa na odgovarajućem mjestu na kojem ga je programer postavio.

Zadatak 3:

Program treba tražiti od korisnika unos broja u rasponu od **10-20**. Ukoliko korisnik pogodi raspon; ispisati: **cestitamo-unijeli ste broj u rasponu** i ispisati broj kojeg je korisnik unio. Ako korisnik ne pogodi broj, ispisati: **broj nije u rasponu od 10-20; Pokušajte ponovno.** Petlja se izvršava tako dugo dok je uvjet na TRUE odnosno dok korisnik unosi **brojeve veće od nula.** Ako korisnik unese 0 odnosno FALSE, prekida se izvođenje programa.

Primjer ispisa:

```
unesite broj u rasponu od 10 do 20: 12
('Cestitamo_unijeli ste broj u rasponu ', 12)

unesite broj u rasponu od 10 do 20: 0
```

Rješenje:

```
while True:
    broj = int(input("unesite broj u rasponu od 10 do 20: "))
    if broj >= 10 and broj <= 20:
        print("Cestitamo_unijeli ste broj u rasponu ",broj)

    elif broj==False:
        break
    else:
        print("Broj nije u rasponu od 10 do 20")
        print("Pokusajte ponovno")
```

U uvjeru petlje mogli smo napisati umjesto **True** jedan (1). To u oba slučaja znači da se program prekida jedino u situaciji kada smo unijeli broj nula (0).

Zadatak 4:

Program treba omogućiti korisniku odabir **jedne od četiri aritmetičke operacije (+,-,*,/)** Svakoj operaciji je potrebno dodijeliti neki broj **(npr. 1. Zbrajanje, 2. Oduzimanje, 3. Mnozenje, 4. Dijeljenje, 5. Izlaz iz programa)** Kada korisnik odabere jednu od navedenih operacija, od njega se traži **da unese dva broja**. Ispisati rezultat odabrane operacije.

Rješenje:

```
izbor = 0
while 1:
    print "Dobrodosli u kalkulator.py!"
    print "Izaberi racunsku operaciju:"
    print " "
    print "1) Zbrajanje"
    print "2) Oduzimanje"
    print "3) Mnozenje"
    print "4) Dijeljenje"
    print "5) Izlaz iz kalkulatora"
    print " "
    izbor = input("Unesi broj ispred zeljene operacije:")

    if izbor == 1:
        prib1 = input("Zbroji ovaj broj: ")
        prib2 = input("s ovim brojem: ")
        print prib1, "+", prib2, "=", prib1 + prib2

    elif izbor == 2:
        oduz1 = input("Oduzmi od ovog broja: ")
        oduz2 = input("ovaj broj: ")
        print oduz1, "-", oduz2, "=", oduz1 - oduz2

    elif izbor == 3:
        mnoz1 = input("Pomnozi ovaj broj: ")
        mnoz2 = input("sa ovim brojem: ")
        print mnoz1, "*", mnoz2, "=", mnoz1 * mnoz2

    elif izbor == 4:
        dij1 = input("Podijeli ovaj broj: ")
        dij2 = input("sa ovim brojem: ")
        print dij1, "/", dij2, "=", dij1 / dij2

    elif izbor == 5:
        break

print "Hvala sto ste koristili kalkulator.py!"
```

Rješenje programa se sastoji od postavljanja i provjera uvjeta. While petlja jedino omogućava višestruko ponavljanje programa s blok naredbama sve do tenutka kada je odabrana opcija **break** odnosno prekid programa. U suštini while petlja omogućava izvršavanje programa nebrojeno mnogo puta. Za razliku od for petlje koja ima brojač gdje je moguće manipulirati brojem izvršavanja programa, while petlja ima samo mogućnost ponavljanja do trenutka kada je uvjet ispunjen odnosno korisnik sam odredi prekidanje programa.

7. FUNKCIJE

Funkcije su programske strukture koje služe kako bismo jednom napisani odsječak koda mogli primijeniti nad više različitih operanada. Primjerice, napisali smo porgram za zbrajanje dva broja. **Kada bismo negdje na nekom drugom mjestu u programu željeli zbrojiti dva broja, morali bismo ponovno pisati isti program.** No kada imamo funkciju, to znači da u njoj imamo zapamćen program za **zbrajanje dva broja.** Takva funkcija koja predstavlja sažeti odraz napisanog koda primijeniti će se u nekom drugom programu jednostavnim pozivom u taj programski odsječak. Dakle više nećemo morati pisati kod za zbrajanje dva broja nego ćemo samo pozvati funkciju to jest ključnu riječ pod koju smo spremili kod te operacije i nekoliko linija koda za zbrajanje dva broja će se izvršiti i prikazati rezultat operacije zbrajanja na mjestu gdje smo pozvali tu funkciju.

Definiranje funkcije:

```
def ime_funkcije (popis parametara):

        blok_naredbi

        return vrijednost
```

Nakon ključne riječi **def** piše se odabrano ime funkcije, a u zagradi popis parametara. Ako se u funkciju ne unosi ni jedan parametar, zagrade ostaju prazne, a kada ima više parametara, razdvajaju se zarezom. Na kraju tog retka, stavlja se **dvotočka** koja označava početak bloka naredbi koje definiraju funkciju. Bloku naredbi na kraju je pridodana naredba **return**. Iza nje navodi se vrijednost koju funkcija vraća ili više vriejdnosti koje funkcija vraća. Vrijednost može biti predstavljena **imenom varijable** ili **izrazom** koji određuje vrijednost koja se vraća.

Primjer 1: Napisati funkciju koja provjerava da li je upisan broj paran ili neparan.

Dekalracija:

```
def paran(n):
    if n%2==0:
        print("broj je paran", n)
    else:
         print("broj nije paran", n)
    return
'
```

Poziv:

```
>>> paran(3)
('broj nije paran', 3)
```

U primjeru 1 funkciju smo nazvali **paran (n).** Izraz **n** u zagradi naziva funkcije predstavlja parametar funkcije. Kod poziva funkcije **paran(3)** na mjesto n smo upisali konkretan broj za koji smo željeli provjeriti da li je paran ili neparan.

Primjer 2: Definirati funkciju koja će unositi prirodan broj te će ga vraćati.

Deklaracija:

```
def unos():
    a=int((input("unesi prirodan broj:"))
    return a
```

Poziv:

```
>>> unos()
unesi prirodan broj:25
```

U drugom primjeru kod deklaracije funkcije **unos()** zagrade su **ostale prazne** što znači da se u **funkciju ne unosi ni jedan parametar.**

ZADACI

Zadatak 1:

Definirajte funkciju koja omogućava unos cijelog broja četiri puta. Unutar glavnog programa potrebno je pozvati funkciju za unso cijelog broja.

Ispis:

```
Program omogucava unos pomocu poziva funkcija
unesi prirodan broj:1234
-------------
unesi prirodan broj:2
-------------
unesi prirodan broj:3
-------------
unesi prirodan broj:4
-------------
```

Rješenje:

```
#deklaracija funkcije
def unos():
    a=int(input("unesi prirodan broj:"))


#-------------------------------------------------------------#
print("Program omogucava unos pomocu poziva funkcija    ")

for i in range(1,5):
    unos()
    print("-------------")
```

Funkcija je deklarirana bez parametara. U funkciji **unos()** kreirana je naredba za unos broja. U glavnom dijelu programa u programskoj petlji omogućeno je da se funkcija unos ponavlja 4 puta to jest unutar ranga (1,5). Tako smo izbjegli uzastopno pisanje naredbe za unos na mjesto gdje nam je to najpotrebnije. Sada samo kratkim pozivom funkcije možemo u nekom dijelu programskog koda pozvati naredbu za unos brojeva. U našem primjeru, pozvali smo funkciju unosa unutar tijela for petlje. Naranvo u nekoj drugo situaciji funkciju smo mogli pozvati u drugom dijelu programa.

Zadatak 2:

Napišite funkcije za zbrajanje dva broja i pozovite je u glavnom dijelu programa. Omogućite izvršavanje te funkcije 3 puta.

Ispis:

```
Unesite 1. broj:1
Unesite 2. broj:2
('Zbroj brojeva je:', 1, 'i', 2, 'je', 3)
----------------------
Unesite 1. broj:3
Unesite 2. broj:4
('Zbroj brojeva je:', 3, 'i', 4, 'je', 7)
----------------------
Unesite 1. broj:5
Unesite 2. broj:6
('Zbroj brojeva je:', 5, 'i', 6, 'je', 11)
----------------------
```

Rješenje:

```
#deklaracija funkcije
def zbroj():
    a= int(input("Unesite 1. broj:"))
    b= int(input("Unesite 2. broj:"))
    c=a+b
    print("Zbroj brojeva je:",a,"i",b,"je", c)


#glavni dio programa i POZIV FUNKCIJE, ponavljanej poziva

for i in range(1,4):
    zbroj()
    print("----------------------")
```

Deklarirali smo funkciju **zbroj()** koja omogućava korisniku da istu pozove u nekom dijelu glavnog programa. Funkcija omogućava unos 2 broja, operaciju zbrajanja te ispis zbroja brojeva. Kod poziva funkcije omogućeno je da se ista pozove 3 puta; to je riješeno for petljom **range(1,4).**

Zadatak 3:

Deklarirati funkciju koja će omogućiti množenje dvaju brojeva. Funkcija izvršava operaciju množenja. U glavnom dijelu programa omogućiti unos brojeva za množenje.

Ispis:

```
Unesi 1. broj1
Unesi 2. broj2
('Umnozak brojeva', 1, 'i', 2, 'je', 2)
```

Rješenje:

```
#deklaracija funkcije s argumentima
def mnozenje(a,b):
    print("Umnozak brojeva",a,"i",b,"je",a*b)


#glavni dio programa i POZIV FUNKCIJE
broj1=int(input("Unesi 1. broj"))
broj2=int(input("Unesi 2. broj"))

#poziv funkcije s odabirom argumenata

mnozenje(broj1, broj2)
```

Ovo je primjer funkcije kod koje su unaprijed deklarirani argumenti **def mnozenje(a,b)** . S tim argumentima je uspješno izvedena operacija množenja dva broja. No korisnik sam u glavnom dijelu programa kreira svoje varijable za unos brojeva **broj1, broj2**. I nakon njih poziva funkciju u koju ugrađuje varijable broj1 i broj2: **mnozenje(broj1, broj2).** Originalni parametri funkcije **mnozenje(a,b)** su samo simbolički znakovi kojima označavamo broj parametara u funkciji. Zato kada u glavnom programu koristimo druge varijable to ne znači da su a i b izbrisani. Varijable **broj1 i broj2** su samo „sjele“ na odgovarajuća mjesta parametara deklariranih u funkciji **mnozenje(a,b)**

Zadatak 4:

Izraditi program koji će korisniku nakon što unese broj provjeriti **parnost i negativnost unesenog broja.** Potrebno je izraditi dvije funkcije: jednu za provjeru parnosti broja, a drugu za provjeru negativnosti broja.

Ispis:

```
unesite neki broj 12
('broj je paran', 12)
('pozitivan broj', 12)
```

Rješenje:

```
def paran(n):
    if n%2==0:
        print("broj je paran", n)
    else:
         print("broj nije paran", n)


def negativan(n):
    if n<0:
        print("negativan broj",n)
    else:
        print("pozitivan broj",n)


unos=int(input("unesite neki broj "))
paran(unos)
negativan(unos)
```

Funkcija je opet deklarirana s pripadajućim argumentom **n** nakon toga je korisnik napravio unos preko varijable **unos** i istu je varijablu ugradio u poziv funkcija **paran(unos), negativan(unos).**

Zadatak 5:

Deklarirati funkcije unosa i ispisa elemenata jednodimenzionalnog niza.

Ispis:

```
Unesite broj: 12
Unesite broj: 3
Unesite broj: 4
Unesite broj: 1
Unesite broj: 2
12
3
4
1
2
```

Rješenje:

```
#deklaracija funkcija unosa i ispisa niza

def unos (niz):
    for i in range (0,5):
        niz[i]=int(input("Unesite broj: "))

def ispis(niz):
    for i in range (0,5):
        print (niz[i])

#glavni dio programa i POZIV FUNKCIJE
niz=[0]*5
unos(niz)
ispis(niz)
```

Funkcije su deklarirane s priapadajućim argumentima unos(**niz)** i ispis(**niz)** . Omogućen je unos podataka niza od unaprijed određenih 5 elemenata **niz=[0]*5**. U glavnom dijelu programa, pozivaju se obje funkcije koje vrše unos i ispis niza.

Zadatak 6:

Kreirajte funkciju za **unos niza**, **kvadriranje** i **ispisa kvadrata vrijednosti niza**.

Ispis:

```
Unesite broj: 1
Unesite broj: 2
Unesite broj: 3
Unesite broj: 4
Unesite broj: 5
1
4
9
16
25
```

Rješenje:

```
#deklaracija funkcija
niz=[0]*5
def unos (niz):
    for i in range (0,5):
        niz[i]=int(input("Unesite broj: "))


def kvadrat(broj):
    return broj**2


def ispis(niz):
    for i in range (0,5):
        niz[i]=kvadrat(niz[i])
        print (niz[i])
#glavni dio programa i poziv funkcija


#pozivanje funkcije za unos
unos(niz)
#pozivanje funkcije za ispis kvadrata niza
ispis(niz)
```

Deklarirane su 3 funkcije. Za primijetiti je da je funkcija **kvadrat** ugrađena u funkciju za ispis **niz[i]=kvadrat(niz[i]).** U glavnom dijelu programa, pozvane su samo divje funkcije jer je treća **kvadrat** funkcija pozvana preko funkcije za ispis.

Zadatak 7:

Kreirati funkcije koje će omogućiti **unos elemenata niza, ispis elemenata niza, obrnuti ispis elemenata te ispis samo parnih vrijednosti**. Napravite **izbornik za korisnika** upotrebom **while petlje.**

Ispis:

```
Odaberite
1.Unos podataka  niza
2.ispis podataka niza
3.Obrnuti ispis
4.Ispis parnih elemenata
Vas odabir1
Unesite broj: 1
Unesite broj: 2
Unesite broj: 3
Unesite broj: 4
Unesite broj: 5
Odaberite
1.Unos podataka  niza
2.ispis podataka niza
3.Obrnuti ispis
4.Ispis parnih elemenata
Vas odabir2
1
2
3
4
5
Odaberite
1.Unos podataka  niza
2.ispis podataka niza
3.Obrnuti ispis
4.Ispis parnih elemenata
Vas odabir3
5
4
3
2
1
Odaberite
1.Unos podataka  niza
2.ispis podataka niza
3.Obrnuti ispis
4.Ispis parnih elemenata
Vas odabir4
2
4
Odaberite
1.Unos podataka  niza
2.ispis podataka niza
3.Obrnuti ispis
4.Ispis parnih elemenata
```

Rješenje:

```
#deklaracija funkcija unosa i ispisa niza
niz=[0]*5
suma = 0
def unos (niz):
    for i in range (0,5):
        niz[i]=int(input("Unesite broj: "))

def ispis(niz):
    for i in range (0,5):
        print (niz[i])

def obrnuto(niz):
     for i in range (4,-1,-1):
         print(niz[i])

def parni(niz):
    for i in range (1,5,2):
        print (niz[i])


#glavni dio programa  i poziv funkcije

izbor = 0

while 1:
    print("Odaberite")
    print("1.Unos podataka  niza")
    print("2.ispis podataka niza")
    print("3.Obrnuti ispis")
    print("4.Ispis parnih elemenata")
    izbor=int(input("Vas odabir"))
    if izbor==1:
        unos(niz)
    elif izbor==2:
        ispis(niz)
    elif izbor == 3:
        obrnuto(niz)
    elif izbor==4:
        parni(niz)
    elif izbor==5:
        break
```

Kreirane su četiri funkcije s pripadajućim argumentom **niz** . U glavnom dijelu programa, pozvane su iste funkcije unutar **while** petlje te unutar **if i elif uvjetovanja**. Time smo dobili sažetiji i uredniji kod u glavnom dijelu programa.

Zadatak 8:

Kreirajte dvije funkcije: **funkciju** za unos osobnih podataka {**oib, ime i prezime, grad**} te **funkciju** za unos podataka o školovanju (**razred, naziv, smjer, broj predmeta po razredu**) Rješite problem **OIB-a sa if-else** uvjetovanjem unutar funkcije. **Ako je OIB izvan raspona od 10 znakova ispišite poruku o grešci** i **ponovite unos**. Glavni izbornik izradite pomoću while petlje.

Ispis:

```
Unos podataka
1) Unos osobnih podataka
2) Unos podataka o skolovanju
3)  Izlaz

Unesi odabir:1
Unos osobnih podataka:
----------------------
Unesite OIB:123456
OIB nije u rapsonu 10 znakova:
Pokusajte ponovno
Unesite OIB:1234567890
Ime:'zoran'
Prezime:'hercigonja'
Grad:'varazdin'
Unos podataka
1) Unos osobnih podataka
2) Unos podataka o skolovanju
3)  Izlaz
```

Rješenje:

```
#deklaracija funkcije za unos osobnih podataka

def osobni():
    print("Unos osobnih podataka:")
    print("----------------------")
    oib=int(input("Unesite OIB:"))
    if oib >1 and oib <=10:
         oib=int(input("Unesite OIB:"))

    else:
        print("OIB nije u rapsonu 10 znakova:")
        print("Pokusajte ponovno")
        oib=int(input("Unesite OIB:"))
    ime=input("Ime:")
    prezime=input("Prezime:")
    grad=input("Grad:")

#deklaracija funkcije unosa podataka o skolovanju

def skolski():
    print("Unos podataka o skolovanju:")
    print("---------------------------")
    razred=int(input("Unesite razred"))
    naziv=input("Unesite naziv skole")
    smjer=input("Smjer skolovanja:")
    predmet=input("Broj predmeta trenutnog razreda:")


#glavni dio programa i POZIV FUNKCIJA
izbor = 0
while 1:
    print "Unos podataka"
    print "1) Unos osobnih podataka"
    print "2) Unos podataka o skolovanju"
    print "3)  Izlaz"
    print " "
    izbor = input("Unesi odabir:")

    if izbor == 1:
        osobni()

    elif izbor == 2:
        skolski()

    elif izbor == 3:
        break
```

Zadatak 9:

Potrebno je kreirati 4 funkcije koje će omogućiti **zbrajanje, oduzimanje, množenje i dijeljenje**. Unose brojeva omogućite unutar funkcija. Unutar funkcije se mora izvršiti **određena operacija i ispisati rezultat iste**. Izbornik izraditi pomoću while petlje.

Ispis:

```
Dobrodosli u kalkulator.py!
Izaberi racunsku operaciju:

1) Zbrajanje
2) Oduzimanje
3) Mnozenje
4) Dijeljenje
5) Izlaz iz kalkulatora

Unesi broj ispred zeljene operacije:1
Zbroji ovaj broj: 1
s ovim brojem: 2
```

Rješenje:

```
def zbroj():
    prib1 = input("Zbroji ovaj broj: ")
    prib2 = input("s ovim brojem: ")
    print (prib1, "+", prib2, "=", prib1 + prib2 )



#deklaracija programa s pozivom funkcija
def razlika():
    oduz1 = input("Oduzmi od ovog broja: ")
    oduz2 = input("ovaj broj: ")
    print (oduz1, "-", oduz2, "=", oduz1 - oduz2)

def umnozak():
    mnoz1 = input("Pomnozi ovaj broj: ")
    mnoz2 = input("sa ovim brojem: ")
    print (mnoz1, "*", mnoz2, "=", mnoz1 * mnoz2)

def kolicnik():
    dij1 = input("Podijeli ovaj broj: ")
    dij2 = input("sa ovim brojem: ")
    print (dij1, "/", dij2, "=", dij1 / dij2)

#glavni dio programa s pozivima funkcija
izbor = 0
while 1:
    print "Dobrodosli u kalkulator.py!"
    print "Izaberi racunsku operaciju:"
    print " "
    print "1) Zbrajanje"
    print "2) Oduzimanje"
    print "3) Mnozenje"
    print "4) Dijeljenje"
    print "5) Izlaz iz kalkulatora"
    print " "
    izbor = input("Unesi broj ispred zeljene operacije:")

    if izbor == 1:
        zbroj()

    elif izbor == 2:
        razlika()

    elif izbor == 3:
        umnozak()

    elif izbor == 4:
        kolicnik()

    elif izbor == 5:
        break
```

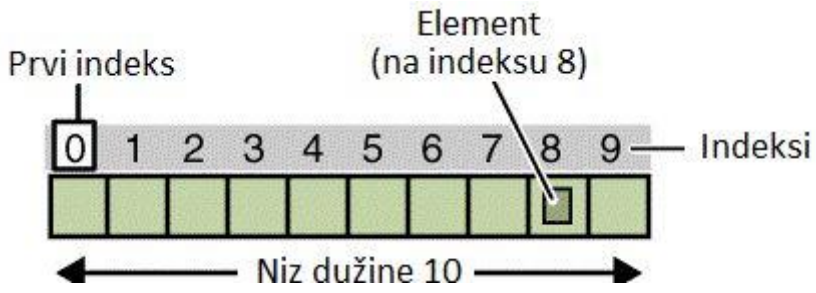
8. JEDNODIMENZIONALNA POLJA

Jednodimenzionalni niz je programska struktura koja omogućava unos više podataka spremljenih na nekoj memorijskoj lokaciji. Možemo ih zamisliti kao varijablu koja omogućuje istovremen unos više brojeva na jendu memorijsku lokaciju u memoriji računala. Primjerice deklaracijom varijable alociramo neki memorijski prostor u koji smještamo podatke u brojčanom ili tekstualnom obliku. Naravno unutar varijable smije se nalaziti samo jedan podatak. To znači da svakoj varijab li pripada samo jedna podatak. Unutar jednodimenzionalnog niza, može se smjestiti više podataka na jednu memorijsku lokaciju.

Primjerice deklaracija bvarijable u Pythonu: **a=15.** No unutar niza, moguće je smjestiti više podataka. Deklaracija jednodimenzionalnog niza izgleda ovako:

```
>>> a=[1,2,3,4,5,6,7,8]
>>> a
[1, 2, 3, 4, 5, 6, 7, 8]
```

Unutar uglatih zagrada, možemo nizati podatke; u ovom slučaju brojeve od 1 do 8. Jednodimenzionalni niz možemo zamisliti kao izduženi pravokutnik s kvadratićima koji označavaju pozicije za zapisivanje podataka niza.



Svaki kvadratić se puni odgovarajućom vrijednosti onim redoslijedom kojim korisnik puni taj niz. Pozicije za zapisivanje podataka niza nazivamo **elementima**. Svaki element ima pridružen **indeks** koji služi za **dohvaćanje elemenata niza.** Indekse uvijek započinjemo brojati od nule. Dužinu niza određujemo sami pisanjem programskog koda.

Niz možemo deklarirati na više načina:

1. način: deklaracija određivanjem fiksnih vrijednosti niza.

```
>>> a=[1,2,3,4,5,6,7,8]
>>> a
[1, 2, 3, 4, 5, 6, 7, 8]
```

2.način: deklaracija niza određivanjem duljine niza i unosom elemenata preko tipkovnice

```
niz=[0]*10
for i in range (0,10):
    niz[i]=int(input("unesite  broj: " ))

for i in range (0,10):
    print(niz[i])
```

Dakle kada niz deklariramo na način da unosimo vrijednosti preko tipkovnice, potrebno je početnu vrijednost niza postaviti na **0** zatim odrediti njegovu duljinu na određeni broj elemenata niza. U prethodnom primjeru to je 10 elemenata. Zatim su nam potrebne dvie for petlje; jedna za punjenje niza i jedna za ispis elemenata niza. Svaku do petlji ograničimo na broj elemenata na koji smo postavili niz. U našem slučaju to je 10 elemenata po nizu. Unutar petlje za unos trebamo naredbu za unos elemenata niza **input**. Istu for petlju upotrijebimo za ispis elemenata niza samo usmjesto naredbe input stavimo naredbu **print** koja će omogućiti ispis elemenata niza.

Grafički primjer niza s elementima:

a	1	2	3	4	5	6
indeks	**0**	**1**	**2**	**3**	**4**	**5**

NAPOMENA:

Ukoliko želimo ispisati specifični element niza kao na primjer **peti element**, onda moramo prije svega dobro odbrojati pozicije **indeksa niza**. **Indeks započinje brojati s nulom**. Dakle ako tražimo **5. element niza**, moramo obavezno navesti da želimo ispisati element pod **indeksom 4 jer je to brojavši od nule peta pozicija indeksa**. Dakle s tim treba biti vrlo oprezan.

ZADACI

Zadatak 1:

Deklarirajte niz naziva **niz** i ograničite ga na **10 elemenata**. Omogućite unos elemenata preko tipkovnice. Ispišite elemente niza.

Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
1
2
3
4
5
6
7
8
9
10
```

Rješenje:

```
niz=[0]*10 #deklaracija niza

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: " ))

for i in range (0,10):
    print(niz[i])
```

Odabrani način deklaracije je deklaracija niza određivanjem duljine niza i unosom elemenata preko tipkovnice. To znači da nakon deklaracije niza od 10 elemenata pomoću dviej for petlje smo izveli unos pojedinog elementa niza i ispis istih.

Zadatak 2:

Deklarirajte niz naziva **niz** i ograničite ga na **10 elemenata**. Omogućite unos elemenata preko tipkovnice. Ispišite samo **parne elemente niza**.

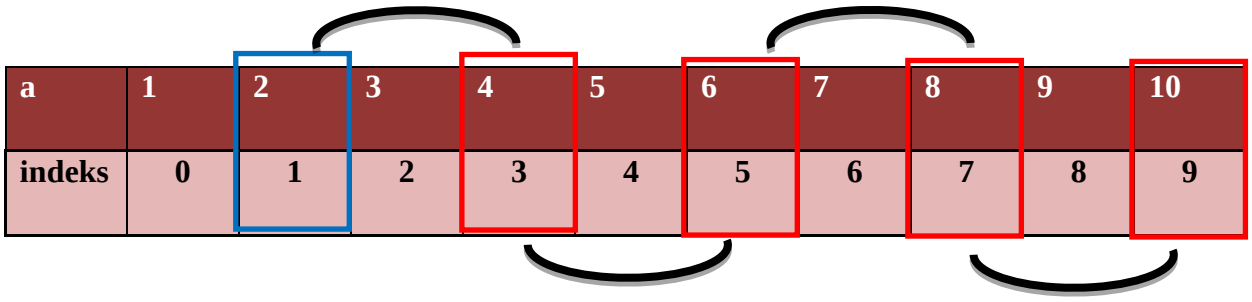
Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
2
4
6
8
10
```

Rješenje:

```
niz=[0]*10 #deklaracija niza

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: " ))

for i in range (1,10,2):
    print(niz[i])
```

Za ispis parnih članova niza, bilo je potrebno u for petlji započeti s vrijednošću 1 i ograničiti se na završnu vrijednost 10. Isto tako potrebno je postaviti treći element for uvjeta (korak for petlje) na 2. Time smo dobili ispis parnih brojeva.



Dakle prvi element petlje smo postavili na jeda **for(1,10,2)** .To znači da smo kao početnu vrijednost ispisa stavili **indeks 1.** Na indeksu 1 nalazi se prvi parni element niza ondosno **broj 2**. Treći element for petlje **for(1,10,2)** postavljen je na dva čime se nizu zadalo da ispisuje svaki drugi element niza.

Zadatak 3:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Omogućite unos elemenata preko tipkovnice. Ispišite svaki četvrti element niza.

Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
1
4
7
10
```

Rješenje:

```
niz=[0]*10 #deklaracija niza

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: " ))

for i in range (0,10,3):
    print(niz[i])
.
```

Dakle sada smo samo kod petlje za ispis početnu vrijednost postavili na nulu šot znači da se ispisuje element s indeksom 0 kao početni . Broj koraka je stavljen na 3 da se omogući ispisvanje elemenata na svakom drugom indeksu.

a	1	2	3	4	5	6	7	8	9	10
indeks	0	1	2	3	4	5	6	7	8	9

Zadatak 4:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Omogućite unos elemenata preko tipkovnice. Ispišite neparne elemente niza.

Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
1
3
5
7
9
```

Rješenje:

```
niz=[0]*10 #deklaracija niza

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: " ))

for i in range (0,10,2):
    print(niz[i])
```

a	1	2	3	4	5	6	7	8	9	10
indeks	0	1	2	3	4	5	6	7	8	9

Zadatak 5:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite niz bez unosa preko tipkovnice. Ispišite niz u obrnutom redoslijedu od 10 prema 1..

Primjer ispisa:

```
ispis niza
10
9
8
7
6
5
4
3
2
1
```

Rješenje:

```
niz=[0]*10

#Automatsko popunjavanje niza
for i in range (0,10):
    niz[i]=i+1


print("ispis niza")

for i in range (9,-1,-1):
    print(niz[i])
```

Za početka da bi kreirali niz bez da unosimo njegove elemente preko tikovnice, napisali smo **niz[i]=i+1**. Taj i+1 znači da se svaki idući element niza puni preko **for petlje i povećava se za 1.** Nadalje kod ispisa u petlji, morali smo postaviti početnu vrijednost na **9** to jest petlja će ispisivati od zadnjeg elementa na **indeksu 9** prema prvom s indeskom 0. Završna vrijednsot je -1 što znači da niz mora ići u obrnutom smjeru. A broj koraka je isto tako -1 iz razloga što se svaki indeks smanjuje kako se kreće od desno prema lijevo.

a	1	2	3	4	5	6	7	8	9	10
indeks	0	1	2	3	4	5	6	7	8	9

Od desna uliievo

Početni indeks se umanjuje za jedan i spušta se prema liejvo strani točnoje prema indeksu 0

Za svaki -1 korak.

Zadatak 6:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite unosom preko tipkovnice. Zbrojite elemente nizova i ispišite njihov zbroj.

Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
ispis niza
1
2
3
4
5
6
7
8
9
10
()
('suma niza:', 55)
```

Rješenje:

```
niz=[0]*10 #deklaracija niza
suma=0

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: "))

#suma niza
for i in range (0,10):
    suma=suma+niz[i]

#ispis niza
print("ispis niza")
for i in range (0,10):
    print(niz[i])
print()

print("suma niza:",suma)
```

Ključan dio ovog programa je linija koda **suma=suma+niz[i]** što znači da se u varijablu suma sprema zbroj svakog člana niza. Na primjer suma je na početku deklarirana na vrijednost 0 ->**suma=0**. Zatim u Sumu ulazi prvi broj niza a to je 1. Zatim se unutar te iste varijable **suma** koja sad trenutno sadržava samo vrijednost 1 vrši opet zbrajanej s idućim elementom niza. Dakle sad je **suma= 1**+idući element niza. I tkao dalje do zadnejg elementa niza.

Zadatak 7:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite niz bez unosa preko tipkovnice. Ispišite sumu **parnih** elemenata niza.

Primjer ispisa:

```
ispis niza
1
2
3
4
5
6
7
8
9
10
('Zbroj parnih brojeva:', 30)
```

Rješenje:

```
niz=[0]*10
zbroj=0

#Automatsko popunjavanje podacima
for i in range (0,10):
    niz[i]=i+1

#zbroj parnih brojeva u nizu
for i in range (0,10):
    if niz[i]%2==0:
        zbroj=zbroj+niz[i]

#ispis niza
print("ispis niza")
for i in range (0,10):
    print(niz[i])
```

Da bi u petlji provjerili koji su elementi parni koji nisu, priej engo uđu u zbroj, ispitali smo svaki element pomoću if-else odluke i ispitali svaki element niza modularnim dijeljenjem s brojem 2 (**%-modularno dijeljenje)** Ukoliko zadani član niza nije nakon modularnog dijeljenja nije imao ostatak to jest ostatak je bio == **0**, pribrojen je vrijablu zbroj i tako sve do zadnejg elementa niza.

Zadatak 8:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite niz bez unosa preko tipkovnice. Pomnožite svaki element niza **brojem 3**.

Primjer ispisa:

```
ispis niza
3
6
9
12
15
18
21
24
27
30
```

Rješenje:

```
niz=[0]*10

#Automatsko popunjavanje niza
for i in range (0,10):
    niz[i]=i+1

#Kvadriranje brojeva
for i in range (0,10):
    niz[i]=niz[i]*3

#ispis niza
print("ispis niza")
for i in range (0,10):
    print(niz[i])
print()
```

Da bi sve elemente nizova pomnožili s 3 ponovno smo koristili već prije korištenu metodu: **niz[i]=niz[i]*3.** Time smo dobili da se prvi element niza 1 smjesti u niz[i] i u njemu se izvrši operacija množenja. Nakon toga u isti niz za idući element se izvrši ista operacija i sve se spremi pononvo u niz:**niz[i]=niz[i]*3** Ovo izdvajanje niz[i]=niz[i]*3 znači da se operaciaj množenaj unutar niza prolazi svakim elemenomt niza.

Zadatak 9:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite niz bez unosa preko tipkovnice. Izvršite kvadriranje – operacija u pythonu je **a**2**.

Primjer ispisa:

```
ispis niza
1
4
9
16
25
36
49
64
81
100
```

Rješenje:

```
niz=[0]*10

#Automatsko popunjavanje niza
for i in range (0,10):
    niz[i]=i+1

#Kvadriranje brojeva
for i in range (0,10):
    niz[i]=niz[i]**2

#ispis niza
print("ispis niza")
for i in range (0,10):
    print(niz[i])
print()
```

Zadatak 10:

Kreirajte dva niza od po **5 elemenata**. Zatim napunite prvi niz preko tipkovnice. Nakon toga premjestite elemente prvog niza u drugi kreirani niz i ispišite taj niz.

Primjer ispisa:

```
unesite broj: 1
unesite broj: 2
unesite broj: 3
unesite broj: 4
unesite broj: 5
ispis originalnog niza
1
2
3
4
5
ispis drugog niza
1
2
3
4
5
```

Rješenje:

```
niz1=[0]*5
niz2=[0]*5

#unos podataka u niz
for i in range (0,5):
    niz1[i]=int(input("unesite broj: "))
print("ispis originalnog niza")
for i in range (0,5):
    print(niz1[i])
#Prepisivanje parnih brojeva u drugi niz
j=0
for i in range (0,5):
        niz2[j]=niz1[i]
        j=j+1

#ispis niza
print("ispis drugog niza ")
for i in range (0,5):
    print(niz2[i])
print()
```

Zadatak 11:

Deklarirajte niz naziva **niz** i ograničite ga na 10 elemenata. Napunite unosom preko tipkovnice. Kopirajte u drugi niz samo parne elemente prvog niza.

Primjer ispisa:

```
unesite broj: 1
unesite broj: 2
unesite broj: 3
unesite broj: 4
unesite broj: 5
unesite broj: 6
unesite broj: 7
unesite broj: 8
unesite broj: 9
unesite broj: 10
ispis originalnog niza
1
2
3
4
5
6
7
8
9
10
ispis niza
2
4
6
8
10
0
0
0
0
0
```

Rješenje:

```
niz1=[0]*10
niz2=[0]*10

#unos podataka u niz
for i in range (0,10):
    niz1[i]=int(input("unesite broj: "))
print("ispis originalnog niza")
for i in range (0,10):
    print(niz1[i])
#Prepisivanje parnih brojeva u drugi niz
j=0
for i in range (0,10):
    if niz1[i]%2==0:
        niz2[j]=niz1[i]
        j=j+1

#ispis niza
print("ispis niza")
for i in range (0,10):
    print(niz2[i])
print()
```

Kod ispisa drugog niza, pojavile su se vrijednsoti 0,0,0,0. To nije pogreška nego ukoliko unaprijed određeni niz kao u našem primjeru niz od 10 elemenata nije **cjelovito popunjen sa svim elementima, nedostajući elementi poprimaju vrijendost nula.** Dakle niz sam dodaje vrijednost 0 na prazna mjesta elemenata.

Zadatak 12:

Napunite niz s 10 elemenata i ispišite najveći i najmanji element niza.(MIN/MAX)

Primjer ispisa:

```
unesite  broj: 1
unesite  broj: 2
unesite  broj: 3
unesite  broj: 4
unesite  broj: 5
unesite  broj: 6
unesite  broj: 7
unesite  broj: 8
unesite  broj: 9
unesite  broj: 10
Niz:
1
2
3
4
5
6
7
8
9
10
('Najveci broj u nizu:', 10)
('Najmanji broj u nizu:', 1)
```

Rješenje:

```
niz=[0]*10
min=0
max=0

#unos podataka u niz
for i in range (0,10):
    niz[i]=int(input("unesite  broj: "))

#pocetna vrijednost minimuma i maksimuma
min=max=niz[0]

print("Niz:")
for i in range (1,10):
    if min>niz[i]:
        min=niz[i]
    elif max<niz[i]:
        max=niz[i]

for i in range (0,10):
    print(niz[i])


print("Najveci broj u nizu:",max)
print("Najmanji broj u nizu:",min)
```

Kod ovog zadatka najbitnija instanca su nam provjera uvjeta pomoću if-else odluke.

```
if min>niz[i]:
    min=niz[i]
elif max<niz[i]:
    max=niz[i]
```

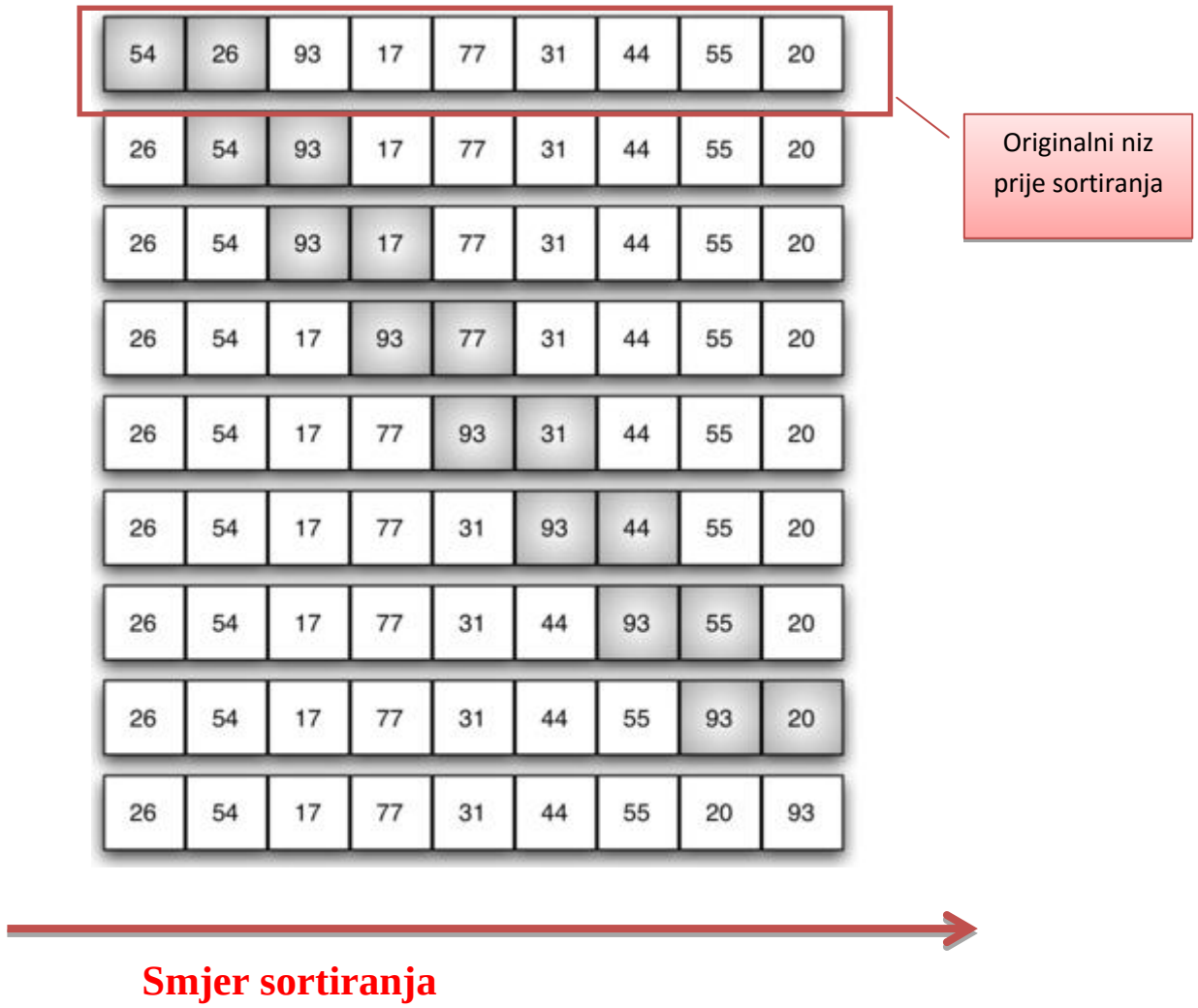
Koristili smo naredbe **min** i **max**. One automatski provjeravaju sve elemente niza uspoređujući ih i na kraju utvrđuju koji je od njih najveći odnosno najmanji. Naredbe min i max rade usporedbu s elemenitma niza. Kada utvrde najmanji ili najveći, dohvate tu vrijednost i smjeste u varijablu ili u našem slučaju niz. **Min>niz[i] i Max<niz[i]** znači da se uspoređuju elementi unutar raspona niza.

8.1. BUBBLE SORT (Mjehuričasto sortiranje)

Mjehuričasto sortiranje je algoritam sortiranja jednodimenzionalnih nizova na temelju usporedbe susjednih elemenata niza. Rad algoritma je vrlo jednostavan. S usporedbom elemenata niza, kreće se s lijeve strane. Kod ovog se sortiranja uspoređuje svaki element sa svojim sljedbenikom i ako je veći od njega, mijenjaju im se vrijednosti.

Nakon prvog prolaska kroz polje najveća će se vrijednost nalaziti u posljednjem elementu polja, tako da u sljedećem koraku možemo postupak ponoviti za polje umanjeno za posljednji element Postupak staje dok ostane samo jedan element u polju ili pak dok u jednom prolasku kroz polje nema ni jedne zamjene.

Algoritam mjehuričastog sortiranja radi pomoću **dodatne privremene varijable.**



Na slici iznad nalazi se originalni jednodimenzionalni niz brojeva **[54, 26, 93, 17, 77, 31, 44, 55, 20].** Niz je potrebno sortirati od najmanjeg elementa prema najvećem. Algoritam kreće s usporedbom elemenata s lijeva na desno. Uspoređuje dva po dva elementa niza. Na primjer prvo uspoređuje zasjenjena polja na slici iznad s brojevima 54 i 26. Uspoređuje te brojeve i utvrđuje koji je od njih manji. Kako je 26 manji od 54, brojevi mijenjaju pozicije.s Broj 26 dolazi na mjesto broja 54. Nakon toga se broj 54 uspoređuje sa susjednim elementom niza.

Uspoređuje se svaki element sa svakim tako dugo dok se ne dobije niz elemenata od najmanjeg do najvećeg. Nakon svih usporedbi niz treba izgeldati ovako: [17, 20, 26, 31, 44, 54, 55, 77, 93].

U programskom kodu, mjehuričasto sortiranje bi izgledalo ovako:

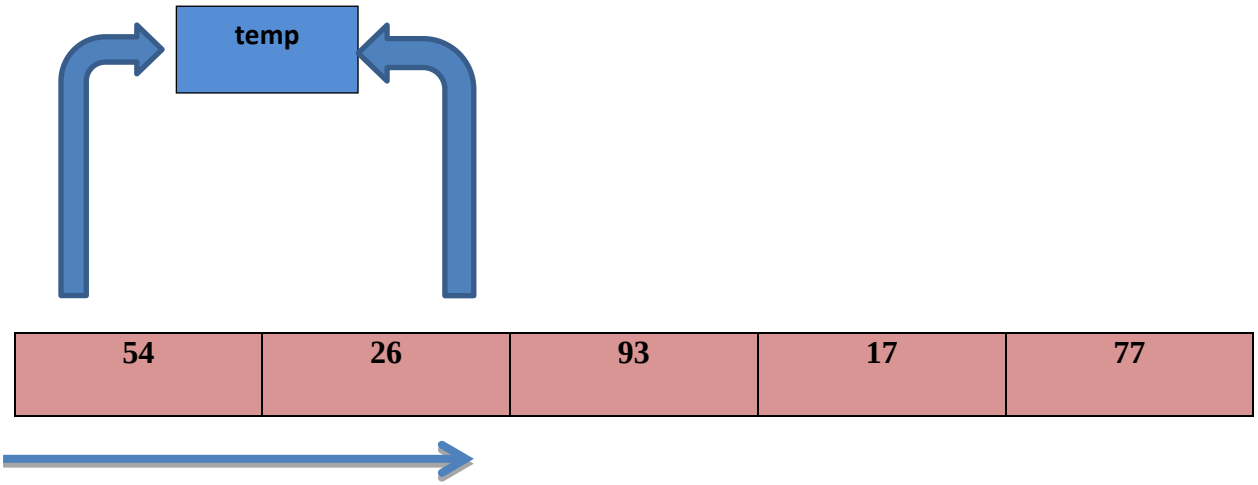
```
niz = [54,26,93,17,77,31,44,55,20]
for i in range(len(niz)):
        for j in range(i):
            if niz[j]>niz[j+1]:
                temp = niz[j]
                niz[j] = niz[j+1]
                niz[j+1] = temp

print(niz)
```

Kreirali smo niz elemenata koji nisu posloženi po redoslijedu od manjeg prema većem. Imamo dvije petlje. Prva for petlja učitava kreirani niz. Druga petlja radi provjeru za svaki element **niza i** pomoću **if uvjetovanja.** Iz if-a vidimo da ukoliko je trenutni element **niz[j]** niza veći od susjednog (sljedećeg) **niz[j+1]** onda napravi zamjenu.

Ključan dio programskog koda je:

```
temp = niz[j]
niz[j] = niz[j+1]
niz[j+1] = temp
```

Za zamjenu je potrebno deklarirati dodatnu varijablu (u našem primjeru **temp**).Ta varijabla je privremena jer služi samo u svrhu usporedbe. Na primjer ako uspoređujemo brojeve 54 i 26, privremena variajbla služi kao dodatno mjesto unutar niza preko kojeg se radi zamjena.

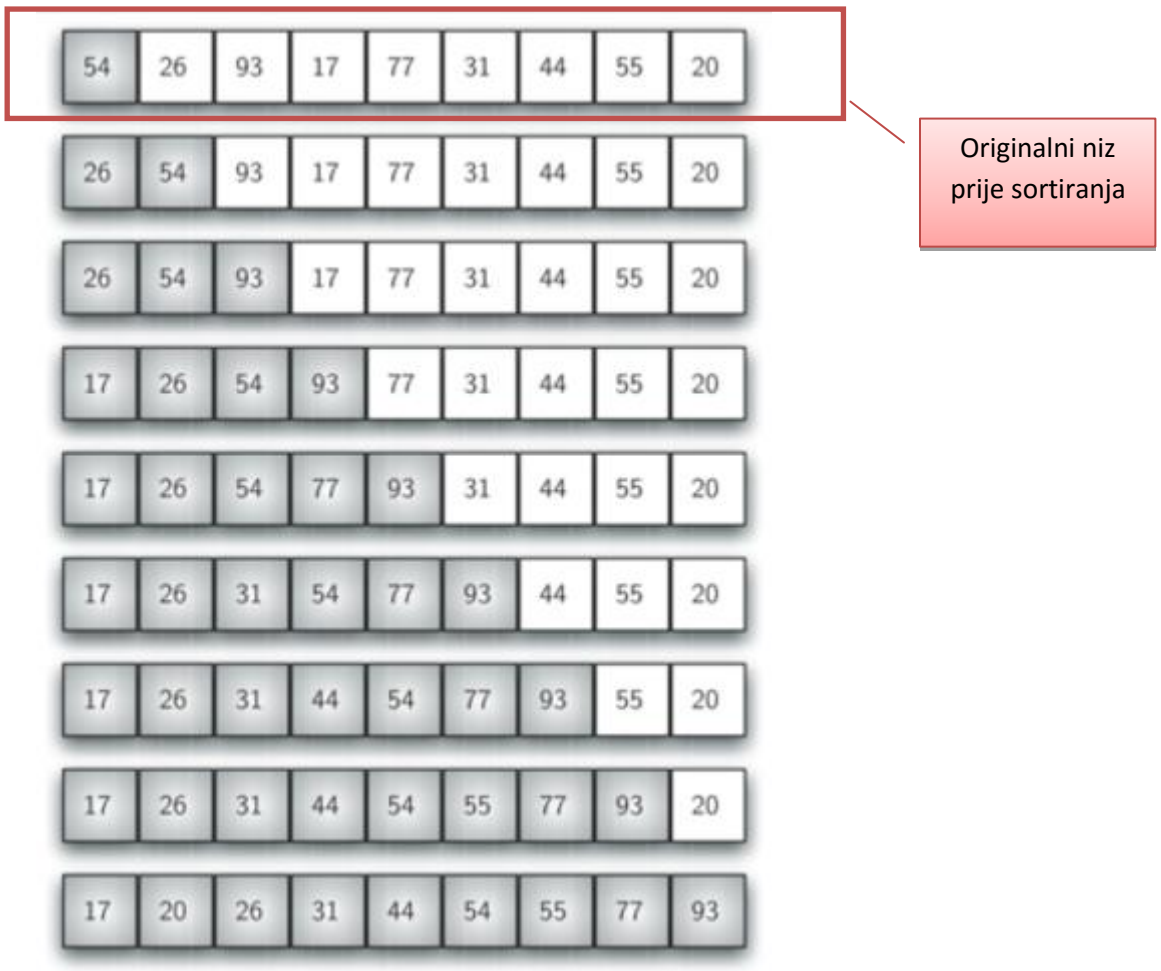


Preko privremene varijable radimo zamjenu brojeva 26 i 54 na primjer. Ona predstavlja privremeno mjesto u koje se sprema broj 26, dok se broj 54 pomiće jedno mjesto prema desno.

8.2. INSERTION SORT (Sortiranje umetanjem)

Sortiranje umetanjem je algoritam sortiranja jednodimenzionalnih nizova koji radi na principu usporedbe elemenata niza i umetanja manje vrijednosti naodgovarajuću poziciju niza. Sortiranje se provodi od lijeva na desno. Sortirani niz, sortiran je od najmanjeg elementa prema najvećem. Kod sortiranja umetanjem se polje dijeli na dva dijela – na sortirani i nesortirani dio. Na početku samo prvi element polja čini sortirani dio polja i u svakom se koraku sortirani dio smanjuje za jedan element, a nesortirani dio povećava za 1, prebacujući prvi element iz nesortiranog dijela polja u sortirani dio. Prebacivanje se vrši tako da se promatrani element iz nesortiranog dijela uspoređuje s posljednjim elementom u soriranom dijelu, te ako je promatrani element manji, posljednji se element iz sortiranog dijela prebacuje za jedno mjesto dalje u polju. Nakon toga se promatrani element uspoređuje s pretposljednjim elementom u polju itd, sve dok se ne naiđe na element u sortiranom dijelu polja koji je manji od promatranog elementa ili pak dok se i prvi element sortiranog dijela polja ne pomakne za jedno mjesto dalje u polju.

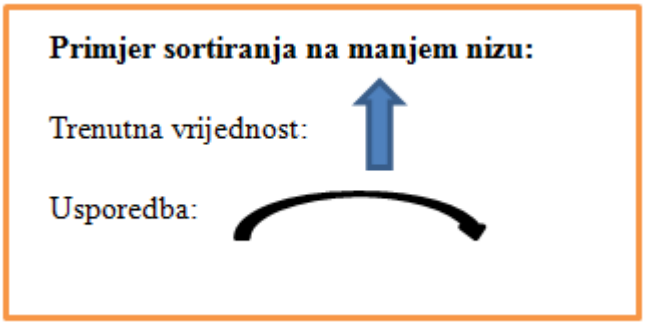
Na taj se način oslobodilo mjesto u polju na koje se umeće novi element i povećava sortirani dio polja za jedan element.

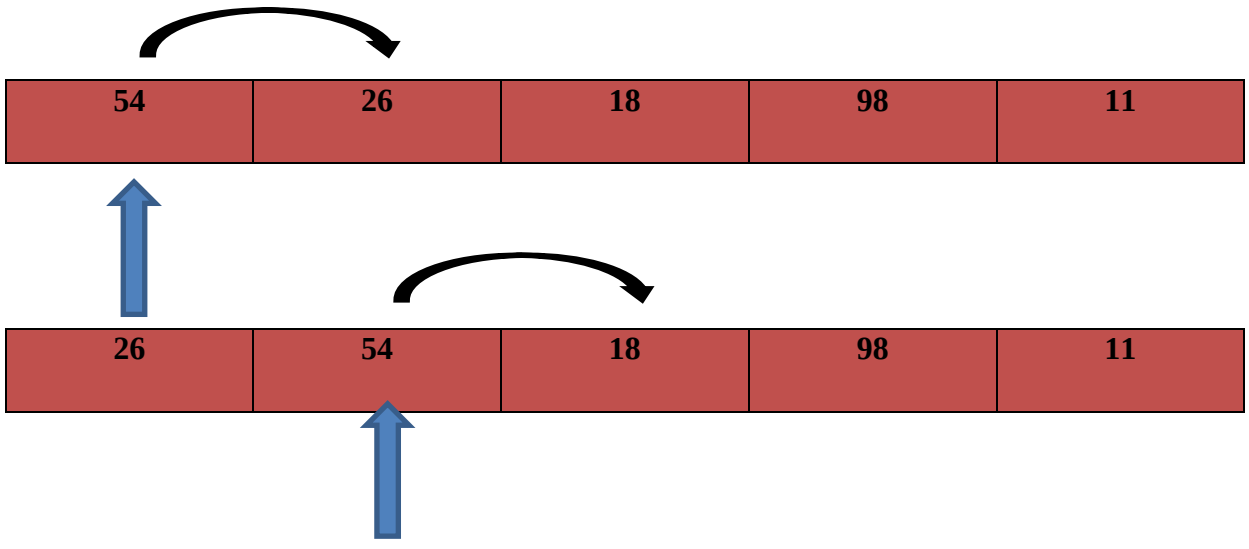


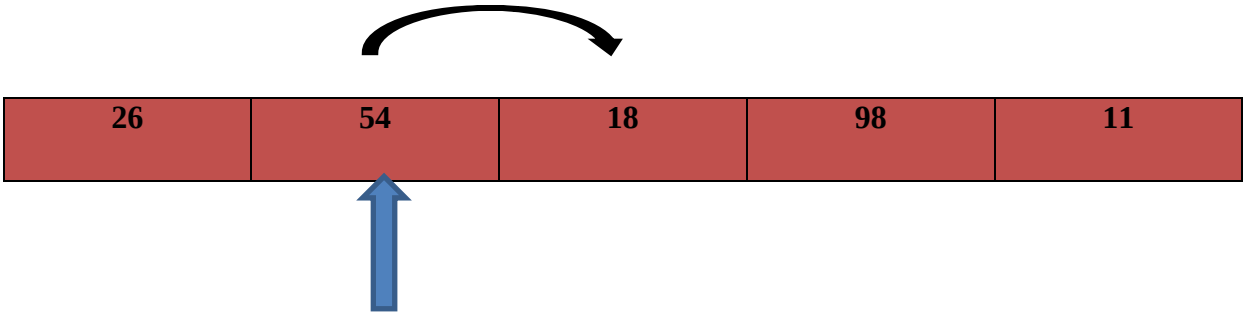
Na primjer u originalnom nizu **[54, 26, 93, 17, 77, 31, 44, 55, 20]** svaki element niza se uspoređuje sa svakim. Trenutni element niza se eusporeduje sa slejdećim. Na primjer prvi element se uspoređuje sa susjednim. Dakle 54 se usporedi s 26. Broj 54 je veći od 26 pa se broj 26 umeće direktno na poziciju broja 54. Drugim riječim zamijene mjesta. U idućem krugu imamo trenutnni broj 54 i idući broj 93. Nema potrebe za umetanjem jer je 93 veći od 54. U idućem koraku trenutna vrijednost je broj 93 i on se uspoređuje sa susjednim brojem 31. Utvrđeno je d aje 93 veće od 31; znači broj 31 treba omaknuti ulijevo za jedno mjesto. Nakon toga se taj broj 31 uspoređuje s brojem 54. I kada se utvrdi da je broj 31 manji od 54, zamijene mjesta. Nakon te zamjene vraćamo se na broj 91 koji postaje trenutnom vrijednosti i uspoređujemo ga sa sljedećim susjedom. Sljedeći susjed je broj 44 i njega treba umetnuti na odgovarajuću poziciju da slijedi niz od najmanjeg do najvećeg.

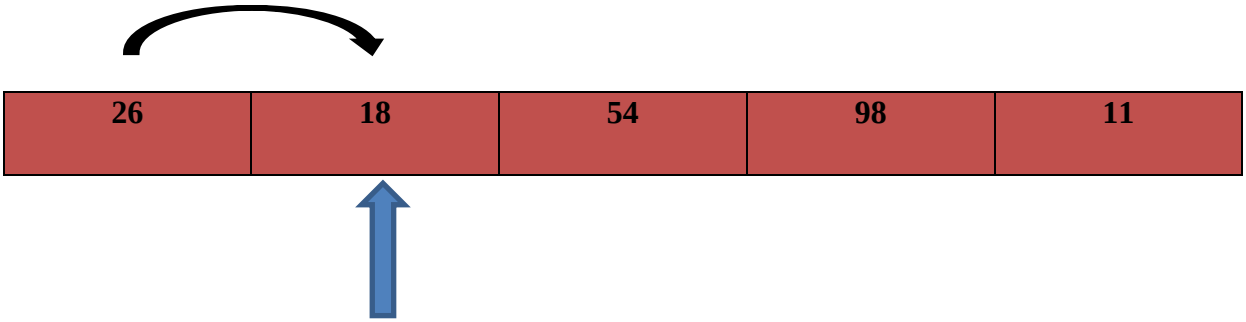
```
niz = [54,26,93,17,77,31,44,55,20]
for i in range(len(niz)):
  trenutna_vrijednost = niz[i]
  pozicija = i

  while pozicija>0 and niz[pozicija - 1]>trenutna_vrijednost:
         niz[pozicija]=niz[pozicija-1]
         pozicija = pozicija-1
         niz[pozicija]=trenutna_vrijednost
print(niz)
```

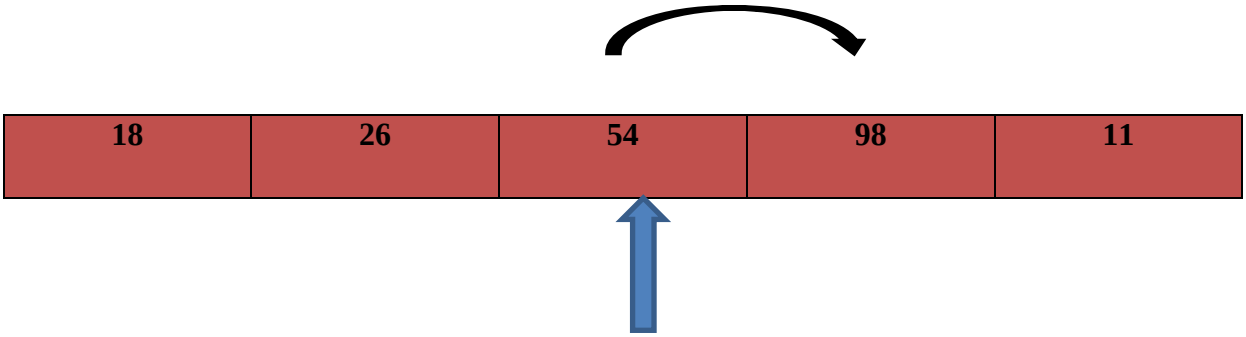
Programski kod je vrlo jednostavan za realizaciju ovog algoritma. Sastoji se od trenutne vrijednosti koja se uspoređuje sa susjednom te pozicijom koja se umanjuje za jedan **pozicija-1** svaki puta kada se naiđe na element čija je vrijednost manja od trenutne vrijednosti.

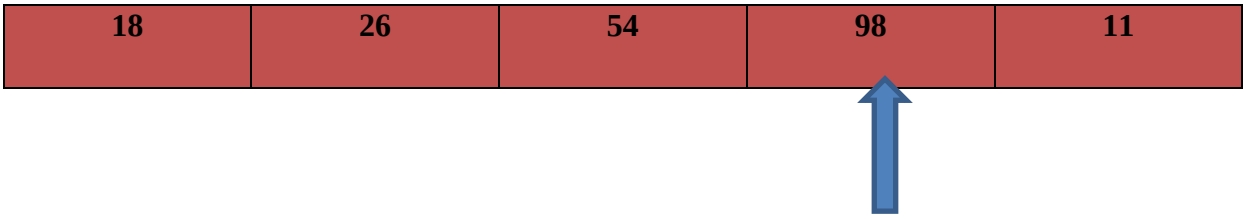


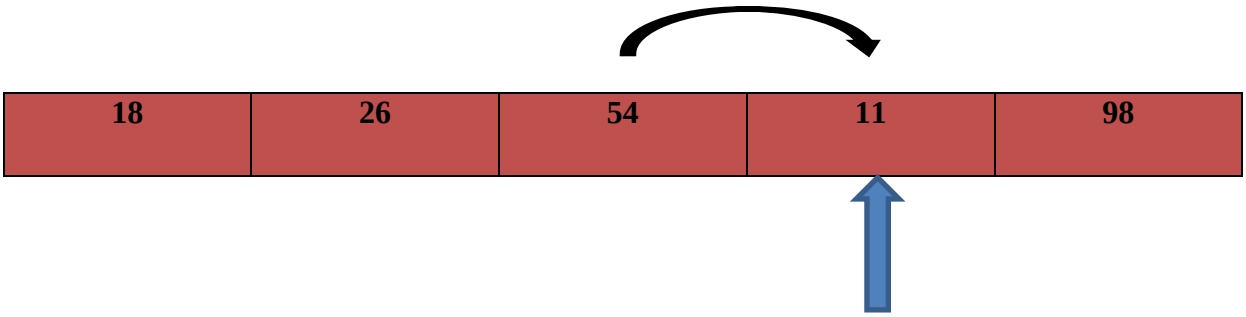


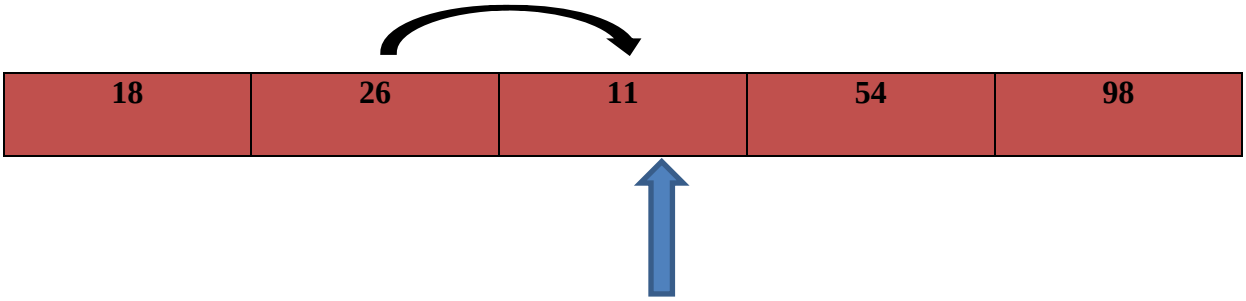

26
54
18
98
11

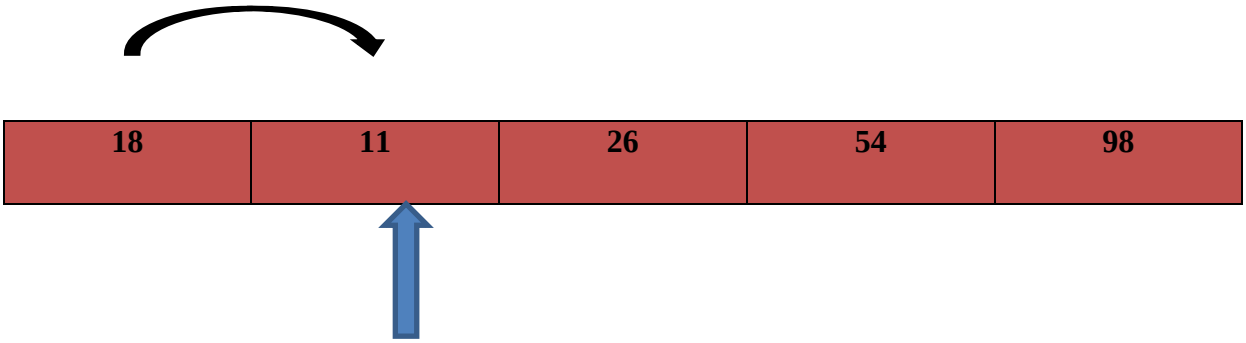

26
18
54
98
11

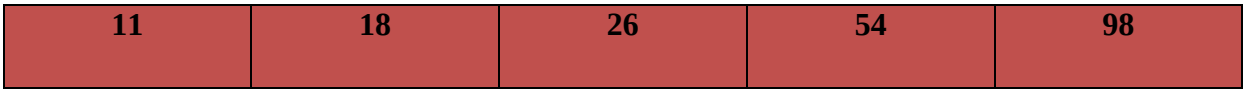

18
26
54
98
11


18
26
54
98
11


18
26
54
11
98



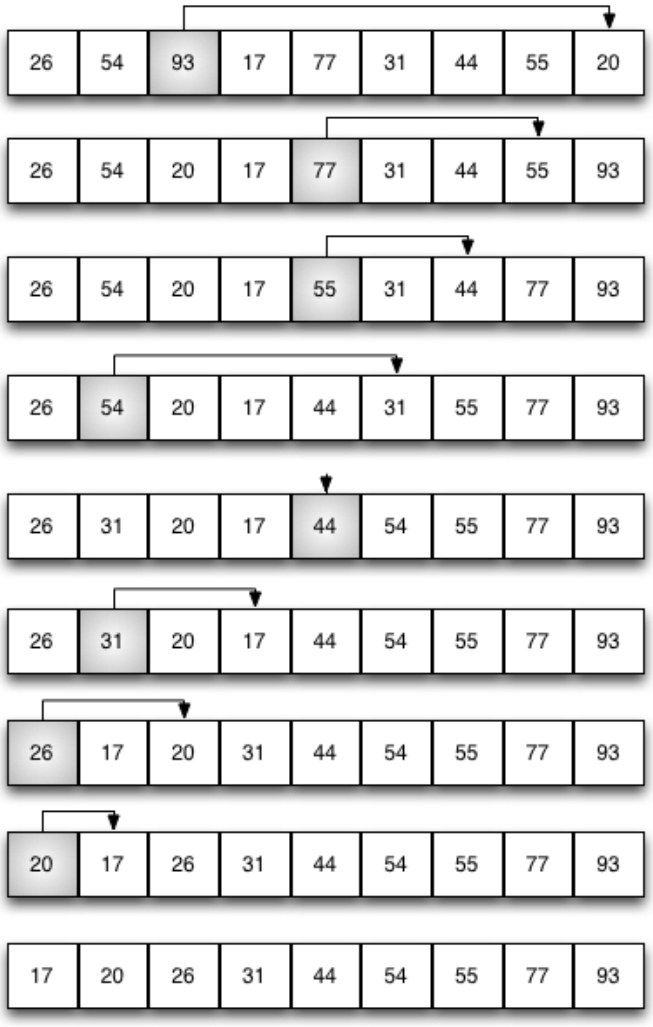




Kraj sortiranog niza

8.3. SELECTION SORT (Sortiranje izborom)

Algoritam sortiranja izborom je jedan on najjednostavnijih algoritama sortiranja. U polju se nađe element s najvećom vrijednošću i njegova se vrijednost zamijeni sa posljednjim elementom. Nakon toga se postupak ponavlja za sve elemente osim posljednjeg. Nakon toga za sve osim posljednja dva itd. sve dok se ne dođe do jednog elementa.



Na slici iznad program je pretražio cijeli niz međusobno uspoređujući vrijednosti elemenata. Naišao je na najveći element niza **broj 93.** Nakon toga je broj 93 smjestio na posljednju poziciju niza. Posljednja pozicija niza s brojem 93 sada ne ulazi u usporedbu. Program ponovno međusobno uspoređuje vrijednosti elemenata nizova izuzev vrijednost 93. Sljedeća najveća vrijednost je **broj 77.** Taj broj smješta na pretposljednju poziciju niza odmah do broja 93. Sljedeća najveća vrijednost je **broj 55** i njega algoritam smješta na pretposljednju poziciju niza i tako sve dok se ne dobije sortirani niz od manjeg prema većem.

Odsječak programskog koda za ovaj algoritam izgleda ovako:

```
niz = [54,26,93,17,77,31,44,55,20]

for i in range(len(niz)-1,0,-1):
  najveci=0
  for j in range(1,i+1):
    if niz[j]>niz[najveci]:
      najveci = j

  temp = niz[i]
  niz[i] = niz[najveci]
  niz[najveci] = temp


print(niz)
```

Ključan dio programa je s usporedbom trenutne najveće vrijednosti **niz[najveci]** i idućeg elementa niza **niz[j].** Zatim je bitan odsječak koda koji omogućava zamjenu elemenata niza. Ponovno uvodimo dodatnu privremenu varijablu **temp** koja služi kao međuprostor za zamjenu vrijednosti s pozicija niza. Na primjer kada smo mijenjali pozicije **brojeva 93 i 20** broj 20 se zapisao u varijablu **temp** i time je upraznio poziciju na koju će se zapisati **broj 93.**

9. DVODIMENZIONALNI NIZOVI

Dvodimenzionalni niz se može predočiti tablicom s zadanim brojem redaka i stupaca. Položaj člana unutar dvodimenzionalnog polja označen je sa dva cjelobrojna indeksa. Prvi indeks određuje redak, a drugi stupac.

Prvi član dvodimenzionalnog polja (prvi redak, prvi stupac) označen je indeksom: **[0,0]** , a posljednji (posljednji redak, posljednji stupac) indeksom: **[(broj redaka [(broj redaka – 1) , (broj stupaca 1) , (broj stupaca – 1)].**

Primjer deklaracije dvodimenzionalnog polja:

a=[[],[]]

Prve zagrade označavaju broj retka, a druge broj stupca. Raspored elemenata unutar matrice određen je indeksima koji započinju sa $a_{0,0,}$ a završavaju sa $a_{i,j}$ što u prijevodu znači da krajnji indeksi matrice završavaju s korisnikovim odabirom stupca i retka. Primjerice ako korisnik odabere 3x3 odnosno tri retka i tri stupca. U Python programskom jeziku postoji više načina kreiranja matrice. Najčešće je to način da se matrica kreira u Python Shell-u. No kako bi u skripti kreirali i napunili matricu elemenitma, moramo se poslužiti for petljom odnosno dvjema for petljama od kojih se jedna odnosi na redak, a druga na stupac.

```
a = []
for i in range(3):#odredjivanje retka
    a.append([])
    for j in range(3):#odredjivanje stupca
        a[i].append(i+j)
print(a)
```

Prva for petlja sa elementom **i** odnosi se na određivanje broja redaka. Druga petlja sa **j** elementom, odnosi se na određivanje broja stupaca. Izraz **a.append** označava u svakom slučaju da se vrijednost stupca pripaja to jest dodaje u matricu.

Kako bi lakše mogli predočiti matricu, sljedeći prikaz popisuje retke i stupce matrice 3x3.

```
              [1, 2, 3]    -> 1. red (indeks 0)
matrica =     [4, 5, 6]    -> 2. red (indeks 1)
              [7, 8, 9]    -> 3. red (indeks 2)
               |  |  |
               |  |  -> 3. stupac (indeks 2)
               |  -> 2. stupac (indeks 1)
               -> 1. stupac (indeks 0)
```

ZADACI

Zadatak 1:

Napraviti fiksni unos stupaca i redaka.

Ispis:

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

Rješenje:

```
a = []
for i in range(3):#odredjivanje retka
    a.append([])
    for j in range(3):#odredjivanje stupca
        a[i].append(i+j)
print(a)
```

U ovom primjeru zadan je rang retka i stupca 3x3 (**range(3)**). Unutar tog ranga, određeno je automatsko popunjavanje elemenata matrice pa je tako dobiveno [0,1,2], [1,2,3], [2,3,4].

Zadatak 2:

Napraviti unos redaka i stupaca matrice preko tipkovnice.

Ispis:

```
Broj redaka:2
Broj stupaca:2
[[0, 1], [1, 2]]
```

Rješenje:

```
m=input("Broj redaka:")
n=input("Broj stupaca:")

a = []
for i in range(m):#odredjivanje retka
    a.append([])
    for j in range(n):#odredjivanje stupca
        a[i].append(i+j)

print(a)
```

U odnosu na prethodni zadatak, napravlejna je mala preinaka. U for petljama na mjestu **range()** su sada upisane varijable retka i stupca odnosno u našem slučaju **m i n.** Naravno koliko god stupaca i redaka unijeli, python će automatski popuniti pozicije elemenata u rangu od nula do vrijednosti koju smo preko tipkovnice odredili kao krajnju vrijednost retka i stupca.

Zadatak 3:

Omogućiti tipkovnički unos redaka i stupaca te elemenata matrice.

Ispis:

```
Broj redaka:2
Broj stupaca:3
unesite elemente matrice:1
unesite elemente matrice:2
unesite elemente matrice:3
unesite elemente matrice:12
unesite elemente matrice:3
unesite elemente matrice:4
[[1, 2, 3], [12, 3, 4]]
```

Rješenje:

```
m=input("Broj redaka:")
n=input("Broj stupaca:")

a=[]
for i in range(m):
    b=[]
    for j in range(n):
        b.append(input("unesite elemente matrice:"))
    a.append(b)

print(a)
```

Prethodni zadatak je nadopunjen naredbom unosa na mjestu for petlje. Na mjesto append umjesto a.append(i+j), upisano je input(„Unesite elemente matrice:“). Oznake i+j označavale su automatsko popunjavanje elemenata matrice. Sada na tome mjestu dodajemo ručno preko tipkovnice svoej elemente.

Zadatak 4:

Pomnožiti svaki element matrice s brojem 3

Ispis:

```
1. matrica
[[0, 3, 6], [3, 6, 9], [6, 9, 12]]
>>>
```

Rješenje:

```
a = []
print("1. matrica")
for i in range(3):#odredjivanje retka
    a.append([])
    for j in range(3):#odredjivanje stupca
        a[i].append((i+j)*3)
print(a)
```

Upotrijebljena je matrica s fiksnim retkom i sutpce te automatskim popunjavanjem elemenata matrice. Da bi svaki element matrice pomnožili s tri, na mjestu **a[i].append((i+j)*3)** dodali smo operaciju množenja i pomnožili taj izraz s tri. Tako smo fiksnu elemente fiksne matrice pomnožili s tri.

Zadatak 5:

Kreirati dvije matrice s fiksnim elementima i zbrojiti ih.

Ispis:

```
1. matrica
[[0, 1, 2], [1, 2, 3], [2, 3, 4]]
2. matrica
[[0, 1, 2], [1, 2, 3], [2, 3, 4]]
[[0, 1, 2], [1, 2, 3], [2, 3, 4], [0, 1, 2], [1, 2, 3], [2, 3, 4]]
```

Rješenje:

```
a = []
for i in range(3):#odredjivanje retka
    a.append([])

    for j in range(3):#odredjivanje stupca
        a[i].append((i+j))
print("1. matrica")
print(a)



b = []
for i in range(3):#odredjivanje retka
    b.append([])

    for j in range(3):#odredjivanje stupca
        b[i].append((i+j))
print("2. matrica")
print(b)
c=a+b
print(c)
```

Zadatak 6:

Preko tipkovnice unesite množitelj matrice s fiksno određenim elementima.

Ispis:

```
unesite mnozitelj matrice:56
[[0, 56, 112], [56, 112, 168], [112, 168, 224]]
```

Rješenje:

```
mnozitelj=int(input("unesite mnozitelj matrice:"))
a = []
for i in range(3):#odredjivanje retka
    a.append([])
    for j in range(3):#odredjivanje stupca
        a[i].append((i+j)*mnozitelj)
print(a)
```

Na mjestu **a[i].append((i+j)*mnozitelj)** dodali smo varijablu **mnozitelj** koja nam je omogućila tipkovnički unos broja s kojim ćemo pomnožiti elemente matrice.

Zadatak 7:

U matrici fiksnih redaka i sutpaca te automatskog unosa elemenata pomnožite redak i stupac.

Ispis:

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

Rješenje:

```
a = []
for i in range(3):#odredjivanje retka
    a.append([])
    for j in range(3):#odredjivanje stupca
        a[i].append((i*j))
print(a)
```

U retku **a[i].append** gdje gdje inače kod automatskog unosa elemenata matrice stoji **(i+j)** , promijenilo se u **(i*j)** čime smo omogućili množenje retka i stupca. Dobivena vriejdnost je zapravo rezultat množenja prvog retka s prvim stupcem.

```
#[0,1,2]            #0,1,2  *  #0     = 0  0  0
#[1,2,3]                       #1       0  1  2
#[2,3,4]    #i*j               #2       0  2  4
```

PRIMJERI

Sljedeći primjeri pokazuju konkretnu primjenu matrica u programiranju.

Primjer 1:

```
matrica = []
matrica.append([1, 2, 3])
matrica.append([4, 5, 6])
matrica.append([7, 8, 9])
print "Ispis cijele liste s podlistama:"
print matrica
print "Ispis red po red:"
for red in matrica:
    print red
print "Cisti ispis svih elemenata - bez zagrada, zareza i sl.: "
for red in matrica:
    for element in red:
        print element,
    print
print "Zadnji element u zadnjem retku: ", matrica[2][2]
```

U primjeru 1 matrica treba ispisati posljednji element koji se nalazi pod sjecištem retka i stupca (2,2) .Matrica ima 3 retka i 3 stupca. S obzirom da se retci i stupci broje od indeksa nule, krajnji element retka i stupca sadražava indeks 2.

```
                [1, 2, 3]    -> 1. red (indeks 0)
matrica =       [4, 5, 6]    -> 2. red (indeks 1)
                [7, 8, 9]    -> 3. red (indeks 2)
                 |  |  |
                 |  |  -> 3. stupac (indeks 2)
                 |  -> 2. stupac (indeks 1)
                 -> 1. stupac (indeks 0)
```

Ono što program treba ispisati jest zadnji element na lokaciji retka i stupca (2,2).

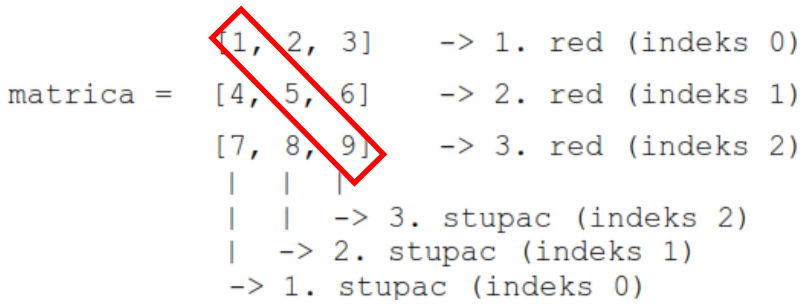
```
Cisti ispis svih elemenata - bez zagrada, zareza i sl.:
1 2 3
4 5 6
7 8 9
Zadnji element u zadnjem retku:  9
```

Primjer 2:

Sljedeća matrica 3x3 treba ispisati elemente koji se nalaze na dijagonali.

```
matrica = []
matrica.append([1, 2, 3])
matrica.append([4, 5, 6])
matrica.append([7, 8, 9])
print "Ispis cijele liste s podlistama:"
print matrica
print "Ispis red po red:"
for red in matrica:
    print red
print "Cisti ispis svih elemenata - bez zagrada, zareza i sl.: "
for red in matrica:
    for element in red:
        print element,
    print
#ispis po glavnoj dijagonali
print ("Ispis po glavnoj dijagonali: ")
print(matrica[0][0])
print(matrica[1][1])
print(matrica[2][2])
```

Ono što je potrebno kod ispisa vrijednosti po dijagonali je točno odrediti indekse retka i stupca.



Dakle moramo ispisati elemente matrice 3x3 pod indeksima: [0][0], [1][1], [2][2]. Dio programskog koda koji se odnosi na ispis po dijagonali je:

```
#ispis po glavnoj dijagonali
print ("Ispis po glavnoj dijagonali: ")
print(matrica[0][0])
print(matrica[1][1])
print(matrica[2][2])
```

Kroz zadane indekse prolazi dijagonala.

```
Ispis po glavnoj dijagonali:
1
5
9
```

Primjer 3:

Sljedeći program je konkretan prikaz primjene matrica u svrhu tabličnog skladištenja podataka. Sljedeća matrica skladišti podatke kao što su jmbg, ime, prezime. Ujedno ispisuje posljednji element matrice na indeksu [2][2].

```
datoteka = []
datoteka.append(["1234567890", "Pero", "Peric"])
datoteka.append(["0987654321", "Jozo", "Jozic"])
datoteka.append(["5432167890", "Ante", "Antic"])
print "Ispis cijele liste s podlistama (tj. datoteke):"
print datoteka
print "Ispis red po red (tj. slog po slog):"
for slog in datoteka:
    print slog
print "Cisti ispis svih polja - bez zagrada, zareza i sl.: "
print "JMBAG, Ime, Prezime"
print "--------------------"
for slog in datoteka:
    for polje in slog:
        print polje,
    print
print
print "Ljepse oblikovan ispis sadrzaja (po dva tabulatora medju stupcima): "
print "JMBAG\t\tIme\t\tPrezime"
print "-----\t\t---\t\t-------"
for slog in datoteka:
    for polje in slog:
        print polje, "\t\t",
    print

print "Zadnje polje u zadnjem slogu: ", datoteka[2][2]
```

U odnosu na prethodni primjer ovaj primjer je nadograđen an način da je u matricu dodan teksualni zapis, a ne brojevi. Ispis bi trebao izgledati ovako:

```
Ispis cijele liste s podlistama (tj. datoteke):
[['1234567890', 'Pero', 'Peric'], ['0987654321', 'Jozo', 'Jozic'], ['5432167890', 'Ante', 'Antic']]
Ispis red po red (tj. slog po slog):
['1234567890', 'Pero', 'Peric']
['0987654321', 'Jozo', 'Jozic']
['5432167890', 'Ante', 'Antic']
Cisti ispis svih polja - bez zagrada, zareza i sl.:
JMBAG, Ime, Prezime
-------------------
1234567890 Pero Peric
0987654321 Jozo Jozic
5432167890 Ante Antic

Ljepse oblikovan ispis sadrzaja (po dva tabulatora medju stupcima):
JMBAG           Ime             Prezime
-----           ---             -------
1234567890              Pero            Peric
0987654321              Jozo            Jozic
5432167890              Ante            Antic
Zadnje polje u zadnjem slogu:  Antic
```

Ujedno ispisan je zadnji element matrice **Antic** na lokaciji retka i stupca [2] [2]**.**

10. KLASE

Klasa - predložak u kojem se čuvaju podaci o nekom elementu: automobil i vozač, te ima implementirane neke radnje nad podacima

- **elementi klase:**
- svojstva (atributi) – čuvaju vrijednosti važne za neki element koji opisujemo (marka automobila, maksimalna brzina, ime vozača,....)
- metode – radnje koje izvodimo nad elementima (uključivanje motora, dodavanje gasa, kočenje,...)
- objekt – konkretizacija klase – nakon što klasi dodamo vrijednosti svojstava, konkretni element koji možemo pokazati

class je ključna riječ i obavezna je prilikom definiranja klase.

- **self** nije ključna riječ, nego je konvencija za ime prvog parametra metode. Prilikom poziva metode na instanci objektaPython će automatski postaviti taj objekt kao prvi parametar.
- Ako klasa ima metodu koja se zove **__init__** Python će je automatski izvršiti pri stvaranju novog objekta. Ova metoda je zgodna za inicijaliziranje atributa.

Deklaracija:

Class ImeKlase:

    Definiranje klase

Definiciju klase čini niz metoda (funkcije) te svojstva (varijable)

- svaka metoda obavezno ima parametar self – preko njega dolazimo do svih svojstava i metoda klase

PRIMJERI:

Primjer 1:

```
#deklaracija klase/nema atributa u tijelu klase
#nije potrebno unositi atribut u klasu Posuda()
class Posuda:
#poziv funkcije unosa biskvita
    def insertBiskvit(self, biskvit):
        self.biskvit = biskvit
#poziv funkcije ispisa biskvita
    def getBiskvit(self):
        print (self.biskvit)
```

Deklaracija podrazumijeva naziv klase **Posuda** s dvije funkcije koje omogućavaju unos i ispis (**insertBiskvit, getBiskvit**). Ova klasa nema inicijaliziranih parametara na početku funkcijom **def _init_.** U drugom dijelu programa je napravljen poziv klase.

```
#pozivi klase i funkcija
mojaPosuda=Posuda()
mojaPosuda.insertBiskvit('mojBiskvit')
mojaPosuda.getBiskvit()
```

Unutar iste skripte je napravljen poziv klase. Kao prvo kreiali smo neku varijablu **mojaPosuda** u nju smo smjestili naziv **Posuda()** i na taj način **kreirali objekt ili instancu klase**. Kada kod ispisa dobijemo:

```
<__main__.Pravokutnik instance at 0x0296C198>
```

To znači da je kreirana jedna instanca (objekt) klase. Budući da nemamo parametara unaprijed definiranih, moramo pozvati funkciju **insertBiskvit('parametar')** koja nam omogućava unos parametra **biskvit**, nakon toga pozivamo funkciju **getBiskvit()** bez upisa parametara i dobivamo ispis onog što smounijeli funkcijom **insertBiskvit('parametar').**

Primjer 2:

```
#deklaracija klase
class Auto:
#inicijalizacija objekta
    def __init__(self, model, color):
        self.model = model
        self.color = color
#prikaz objekta
    def display(self):
        print ('Auto je', self.color, self.model)
#prmjena boje
    def paint(self, new_color):
        self.color = new_color
```

Kod ovog primjera uz deklaraciju klase imamo i funkciju **def_init_** koja omogućava definiranje parametara klase: **model i color.** Nakon toga imamo funkciju **display** kojom ispisujemo parametre nekog objekta klase. Funkcijom **paint** omogućavamo mijenjanje boje kod **color** kao jednog parametra objekta klase **Auto.**

Poziv se nalazi u drugom dijelu programa:

```
#-------------------------------------
#poziv klase i metoda(funkcija)
#stvaranje 1. objekta klase
car_1 = Auto('WV', 'pink')
car_1.display()
#stvaranje 2. objekta klase
car_2 = Auto('Mercedes', 'white')
car_2.display()
car_2.paint('black')
car_2.display()
```

Primjer 3:

```
#deklaracija klase
class Dog:
#stvaranje objekta
    def __init__(self, ime, boja):
        self.ime = ime
        self.boja = boja
#prikaz objekta
    def display(self):
        print("Podaci o psu", self.ime, self.boja)
#promjena boje psa
    def farba(self, nova_boja):
        self.boja=nova boja
```

Klasa je potpuno identičen deklaracije kao i prethodni primjer (Primjer 2) . Imamo dva parametra za pojedini objekt, funkciju prikaza parametara obejkta i mogućnost promjene boje. Poziv se nalazi u drugom dijelu programa.

```
#poziv klase i metoda.
pas_1=Dog('fido','smedja')
pas_1.display()
#poziv metode za promjenu boje psa
pas_1.farba('ljubicasta')
pas_1.display()
```

Primjer 4:

```
class Pravokutnik:
#stvaranje objekta klase
    def insertStranice(self, a,b):
        self.a = a
        self.b = b
#racunanje povrsine
    def getPovrsina(self):
        print ("povrsina",self.a*self.b)
#racunanje opsega
    def getOpseg(self):
        print ("opseg",(2*self.a)+(2*self.b))
```

Sljedeća klasa je deklarirana s 2 parametra stranicom a i b za računanje opsega i površine pravokutnika. Imamo tri funkcije:**1. insertStranice, 2. getPovrsina i 3.getOpseg.** Prva omogućava proizvoljna unos brojeva stranica pravokutnika što ujedno predstavljaju parametre objekta klase. Zatim druge dvije funkcije računaju površinu i opseg pravokutnika. Kod poziva klase i kreiranja objekta klase, ne upisujemo parametre jer oni nisu ionako određeni u samoj klasi nego se pozivaju u funkciji **insertStranice.**

```
#pozivi klase
p=Pravokutnik()
#stvaranje objekta klase
p.insertStranice(12,4)
p.getPovrsina()
p.getOpseg()
```

ISBN: 978-953-59549-0-3