

PYTHON

----- PRVI KORACI -----



RADNA VERZIJA 0.25.04

Dino Isanović

sekcijainfor@gmail.com

NOVU VERZIJU KNJIGE I PROGRAMA ZA
VJEŽBU MOŽETE NAĆI NA OVOM LINKU

[https://github.com/Dinolsanovic/Python-
prvi-koraci](https://github.com/Dinolsanovic/Python-prvi-koraci)

Šta nam je potrebno da bi smo pisali programe u Python programskom jeziku.

1. Potrebno je instalirati programski jezik python na vaš računar. [film za instalaciju python programskog jezika i radnog okruženja VS Code](#)

2. Alternativni editor za pisanje python programa je "Pycharm" pisan specijalno za Python programski jezik.

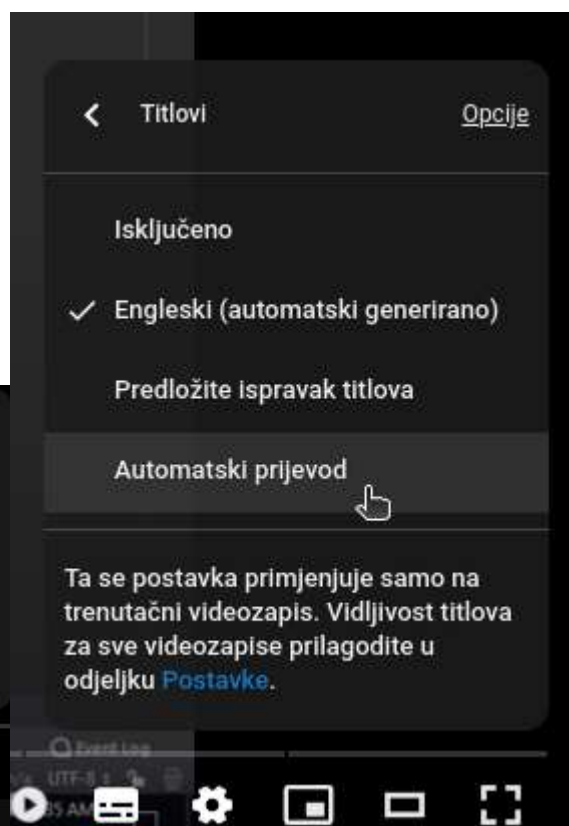
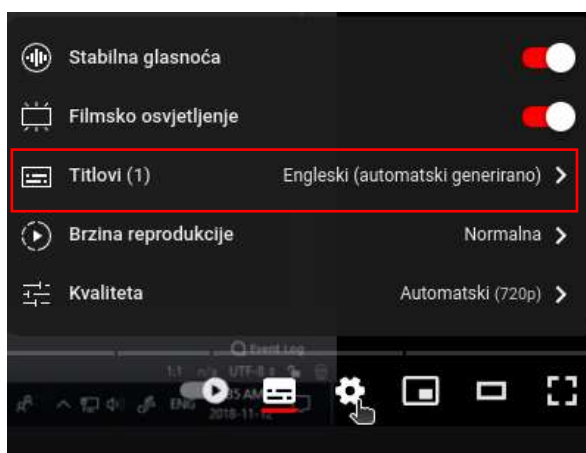
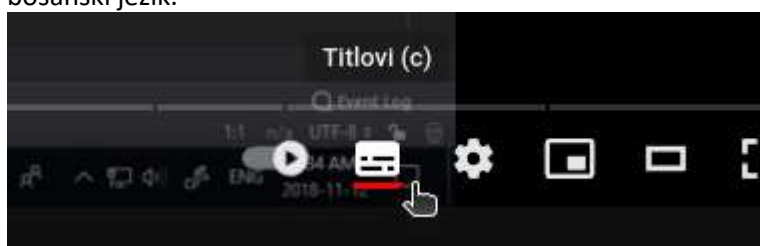
1. [Film za instalaciju Python jezika i Pycharm editora](#)

2. [Video uputstvo za korištenje Pycharm editora](#)

Izbor programa za pisanje programa u python-u je po slobodnom izboru. Na gore navedenim linkovima se nalaze i kompletni video tutorijali za učenje programiranja u python programskom jeziku. Link za kompletan tutorijal je ispod.

<https://www.youtube.com/watch?v=OFrLs22MDAw&list=PLzMcbGfZo4-mFu00qxl0a67RhjjZj3jXm>

Prilikom gledanja filmova na engleskom jeziku potrebno je uključiti titlove i prevod titlova na bosanski jezik.



Odabrati bosanski jezik u ponuđenoj listi jezika koji će se automatski prevoditi.

Za učenje uz pomoć video tutorijala preporučujem korištenje drugog monitora tako da na jednom monitoru gledate film a na drugom monitoru pišete kod ili koristite neku aplikaciju.

NAREDBE UNOSA I ISPISA

PRINT

Naredba za ispisivanje poruka i vrijednosti varijabli na ekranu

računara. Naredba za ispis aktivira se ključnom riječi **print**. Nakon te riječi otvaraju se i zatvaraju zagrade. Unutar zagrada, moguće je ispisati tekst, ali i vrijednosti varijabli. **Tekst koji želimo ispisati na ekranu uvijek se stavlja unutar navodnika.**

Print („pozdrav“)

Kada računar pokrene ovaj jednostavan program izvršit će samo jednu instrukciju koja će na ekranu računara ispisati poruku koja se nalazi između znakova navodnika. U našem slučaju će se pojaviti riječ pozdrav. Nakon čega će se završiti izvođenje programa koji je imao samo jednu komandu print.

Kako bi smo naredili računaru da ispiše tekst “kraj igre”?



Šta je varijabla u odnosu na obični tekst koji ispisujemo na ekranu. Varijabla je sadržaj memorije računara koji će nestati tek kada se program završi. A tekst „pozdrav“ koji smo ispisali je samo parametar funkcije (komande) *print*. Taj tekst neće biti zapamćen u memoriji računara.

Za izvođenje kompleksnih programa potrebni su nam podaci koje pamte VARIABLE . Zbog toga su varijable jedan od najvažnijih dijelova programa. Kada igramo neke igrice u kojima imamo neku životnu energiju “health” ta vrijednost se pamti u nekoj varijabli i kada vrijednost te varijable padne na nulu tada nam igrica to jest program ispiše poruku da je kraj igre (Game over).

Varijabla je rezervisana memorijska lokacija koja pamti podatke i rezultate koji su potrebni za rad programa.

Svaka varijabla ima **svoje ime** i **svoju vrijednost**. Mi u programu strogo koristimo samo ime varijable. A za vrijeme izvršavanja programa računar umjesto imena u svojim operacijama koristi vrijednost varijable. Da bi smo u memoriju računara upisali neku vrijednost moramo rezervisati mjesto pisanjem imena varijable te dodajemo operator dodjeljivanja “=” te upisujemo vrijednost varijable.

$a=10$ Ime varijable je “a” znak dodjeljivanja “=” a vrijednost varijable je 10
U prevodu na naš jezik to bi značilo varijabla “a” dobija vrijednost 10.

znak dodjeljivanja

a = **10**
ime varijable vrijednost varijable

Kada se ova instrukcija izvrši računar će rezervisati memorijsku lokaciju pod nazivom “a” a zatim će u tu lokaciju upisati broj 10.

Za pisanje jednostavnih programa bit će nam potrebne 2 vrste varijabli.

1. Znakovne varijable (slova, brojevi i specijalni znaci ^_^)
2. Numeričke varijable (brojevi)

Kako bi to izgledalo u programu koji u sebi sadrži 2 varijable, a je znakovna varijabla koja ima vrijednost 10 i b je numerička varijabla koja ima vrijednost 10

```
a="10" #vrijednost znakovnih varijabli se upisuju između navodnika
```

```
b=10 # vrijednost numeričke varijable
```

```
print (a+a) #komanda koja će na ekranu ispisati zbir vrijednosti ovih varijabli
```

```
print (b+b)
```

```
# Kasnije u programu pokušajte sabrati varijable a i b
```

```
# print (a+b)
```

```
# komentarisati grešku koja se pojavi
```

Ovaj program proširite sa dodatnim komandama

```
a=b+b
```

```
print(a+b)
```

```
print (a)
```



#komentarisati vrijednosti i greške koje ste dobili na ekranu.

Napisati program koji računa zbir brojeva 10 i 7 koristeći varijable

Komanda Print ima više oblika i formata i primjena ovdje su neki od njih.

1. Osnovni ispis teksta

```
print ("Pozdrav!")
```

Objašnjenje: Funkcija ispisuje string "Pozdrav!" na ekranu.

2. Ispis brojeva

```
print (123)  
print (3.14)
```

Objašnjenje: print() može direktno da ispiše cijeli broj (int) i decimalni broj (float).

3. Ispis kombinovanih vrijednosti

```
ime = "Ana"  
godine = 25  
print("Moje ime je", ime, "i imam", godine, "godina.")
```

Objašnjenje: Upotrebom zareza (,) možemo da kombiniramo više vrijednosti različitih tipova u jednom print() pozivu.

4. Upotreba formatiranih stringova (f-string)

```
ime = "Haso"  
godine = 30  
print(f"Moje ime je {ime} i imam {godine} godina.")
```

Objašnjenje: F-string omogućava umetanje varijabli unutar stringa pomoću {}.

5. Upotreba sep parametra

```
print("Python", "je", "moćan", sep="-")
```

Objašnjenje: Parametar sep definiše separator između vrijednosti. Ovde je separator crtica (-).

6. Upotreba end parametra

```
print("Ovo je prva linija.", end=" ")  
print("Ovo je nastavak.")
```

Objašnjenje: Parametar end definiše šta će biti dodano na kraj ispisane linije. Podrazumjevano je novi red (\n), ali ovdje koristimo razmak.

7. Ispis sa escape karakterima

```
print("Ovo je novi red:\nI ovo je drugi red.")  
print("Ovo je tab:\tTabulacija.")
```

Objašnjenje: Escape karakteri kao što su \n (novi red) i \t (tabulacija) omogućavaju posebne efekte u tekstu.

8. Ispis sa specijalnim znakovima

```
print("Ovo su navodnici: \" i apostrof: \'.")
```

Objašnjenje: Specijalni znakovi poput navodnika (") ili apostrofa (') mogu se ispisati pomoću escape karaktera (\).

9. Ispis na više linija

```
print("""Ovo je  
tekst  
na više linija.""")
```

Objašnjenje: Trostruki navodnici (""" ili ''') dozvoljavaju ispis teksta na više linija.

10. Ispis vrijednosti uz računanje

```
print("5 + 3 =", 5 + 3)
```

Objašnjenje: print() može direktno da prikaže rezultat matematičkih izraza.

11. Upotreba file parametra za ovaj primjer morate u folderu gdje se nalazi ovaj programčić imati tekstualni fajl izlaz.txt kojeg možete napraviti sa notepad aplikacijom.

```
with open("izlaz.txt", "w") as fajl:  
    print("Ovo će biti upisano u fajl.", file=fajl)
```

Objašnjenje: Parametar file omogućava ispisivanje u fajl umjesto na standardni izlaz (ekran).

12. Ispis bez razmaka između vrijednosti

```
print("Prvi", "Drugi", "Treći", sep="")
```

Objašnjenje: Upotrebom parametra sep="" uklanjamo razmake između vrijednosti.

13. Kombinovanje više parametara

```
print("Prva linija", end=" *** ")  
print("Druga linija", sep=">", end="\n\n")
```

Objašnjenje: Kombinovanjem end i sep možemo prilagoditi izgled izlaza.

15. Unicode karakteri

```
print("Srce: \u2665")
```

Objašnjenje: `\u2665` je Unicode kod za simbol srca (♥).

Za ispis brojeva sa tačno dvije decimale u Pythonu koristimo formatiranje stringova.

16. Upotreba `format()` metode

```
broj = 3.14159  
print("Broj na dvije decimale: {:.2f}".format(broj))
```

Objašnjenje: Placeholder `{:.2f}` formatira broj tako da ima dvije decimale. Ovdje je `:.2f` oznaka za formatiranje gdje:

- `.` označava decimalnu tačku,
 - `2` označava broj decimala,
 - `f` označava da je u pitanju broj sa pomičnim zarezom (float).
 - Vitičasta zagrada se dobija kombinacijom tastera `Alt Gr + b` i `Alt gr + n` pod uslovom da koristite tastaturu sa slovima ččž.
-

17. Ispis brojeva u različitim formatima

```
broj = 255  
print(f"Cijeli broj: {broj}")  
print(f"Binarni: {broj:b}")  
print(f"Oktalni: {broj:o}")  
print(f"Heksadecimalni: {broj:x}")
```

Objašnjenje: Formatiranjem pomoću:

- `:b` prikazuje binarni zapis,
 - `:o` oktalni,
 - `:x` hexadecimalni zapis.
-

18. Ispis sa bojama (koristeći ANSI kodove)

```
print("\033[31mOvo je crveni tekst\033[0m")  
print("\033[32mOvo je zeleni tekst\033[0m")
```

Objašnjenje: Koriste se ANSI escape sekvence za postavljanje boje teksta:

- `\033[31m` postavlja crvenu,
- `\033[32m` postavlja zelenu,
- `\033[0m` resetuje boju.

19. Ispis riječi unazad

```
rijec = "Python"  
print("Riječ unazad:", rijec[::-1])
```

Objašnjenje: `[::-1]` je sintaksa za obrtanje stringa.

Srednja zagrada se dobija kombinacijom tastera AltGr + F i AltGR + G

20. Ispis prva dva slova riječi

```
rijec = "Python"  
print("Prva dva slova:", rijec[:2])
```

Objašnjenje:

- `[:2]` uzima podstring od početka stringa do drugog indeksa (ne uključujući ga).
 - Rezultat će biti prva dva slova stringa.
 - Obratite pažnju na dvotačku, ako je ispred broja to znači da se slova broje od početka riječi.
-

21. Ispis zadnja dva slova riječi

```
rijec = "Python"  
print("Zadnja dva slova:", rijec[-2:])
```

Objašnjenje:

- `[-2:]` uzima podstring koji počinje od prethodnog slova i ide do kraja.
- Dvotačka je iza broja to znači da se broje 2 slova od kraja riječi

I još mnogo toga, što će te naučiti kada vam bude potrebno ...

INPUT

Komandu INPUT koristimo kada želimo da korisnik preko tastature unese vrijednost željene varijable (unos podataka koji su potrebni za rješenje problema).



```
ime=input("unesite ime")  
broj1= int(input("unesite cijeli broj"))  
broj2= float(input("unesite decimalni broj"))
```

VARIJABLA = tip podataka (input („prateći tekst“))

Tip podataka
ukazuje kakve
podatke pamti
varijabla:
int – cijeli broj
float – decimalni
broj
boolean – logička
vr.

Prateći tekst obično daje upute korisniku koju
podatak da unese.

a=tip podataka(input (“prateći tekst”))

Ime varijable

Komanda input koju koristimo za unos podataka preko
tastature.

Primjeri:

Želimo da korisnik unese svoje ime :

```
ime1= input(„unesite svoje ime“)
```

Kao što ste vidjeli ovdje nismo naveli tip podataka jer kod unosa običnog teksta to nije potrebno.

ime1 je ime varijable koja će pamtit i ime upisano preko tastature.

Kada radimo sa brojevima moramo razlikovati 2 vrste brojeva cjelobrojni i decimalni (`int` , `float`) što zahtjeva od nas da unaprijed znamo koje vrste brojeva ćemo koristiti. Ako niste sigurni možete koristiti tip **float**.

Cjelobrojne brojeve označavamo sa **int**

```
A= int ( input („unesite duzinu kvadrata“))  # kada računar izvrši ovu komandu čekat će da korisnik preko tastature unese vrijednost varijable A
```

Brojeve sa decimalnim zarezom označavamo sa „**float**“

```
D1= float ( input („unesite djeljenik“))
```

```
D2= float ( input („unesite djelilac“))
```

Kada dođe do izvršavanja funkcije `input` računar staje i čeka da korisnik unese vrijednost označene varijable i kada pritisne tipku ENTER u memoriju računara se zapisuje napisana vrijednost varijable. Koju mi koristimo u programu uz pomoć imena varijable.

Primjeri:

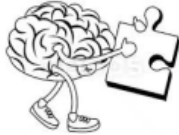
Unesite ova 2 programa i uočite razliku i komentarišite šta rade ovi programi.

```
a = int(input(„unesite vrijednost a“))
b = int(input(„unesite b: “))
s = a + b
print(s)
```

```
a = input()
b = input()
s = a + b
print(s)
```

NAPISATI PROGRAM KOJI RAČUNA ZBIR 2 BROJA

KAKO RAZMIŠLJA ČOVJEK



VAŠ MOZAK ĆE POKUŠATI ZAMISLITI NEKE BROJEVE ZBOG TOGA ŠTO SE U NJEMU NALAZI VELIKA KOLIČINA PODATAK. NARAVNO REZULTAT NIJE TAČAN JER JE IZMIŠLJEN.

UNOS

PODACI U NAŠ MOZAK SE UNOSE POMOĆU ORGANA VIDA ILI SLUHA. PA ĆE MO ČUTI ILI VIDJETI BROJEVE KOJE MORAMO SABRATI.

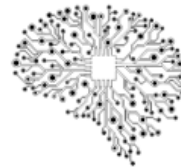
OBRADA

KAKO JE NAŠ MOZAK VEĆ NAUČIO SABIRATI, ON ĆE ODMAH SABRATI BROJEVE KOJE JE ZAPAMTIO U SVOJOJ MEMORIJI.

ISPIS

NAŠ MOZAK ĆE AUTOMATSKI REZULTAT PRETVORITI U GOVOR ILI ĆE KONTROLISATI NAŠU RUKU DA NAPIŠE REZULTAT.

KAKO RAZMIŠLJA RAČUNAR



U MEMORIJI RAČUNARA NEMA NIŠTA JER MU NISMO UNIJELE PODATKE..

PRVI ZADATAK JE UNOS POTREBNIH PODATAKA U MEMORIJU RAČUNARA. PODATCI U RAČUNARU SE PAMTE UZ POMOĆ VARIJABLI.

PODATAKA

PODACI U RAČUNAR SE UNOSE UZ POMOĆ KOMANDE "INPUT". POTREBNO JE UNIJELE 2 BROJA. TE BROJEVE ĆE PAMTITI 2 VARIJABLE.

PODATAKA

RAČUNARU MORAMO NAREĐITI DA IZVRŠI SABIRANJE TIH BROJEVA TO JEST VARIJABLI KOJE PAMTE TE BROJEVE. TAKO DA ĆE RAČUNAR PRILIKOM SABIRANJA PROVJERITI SVOJU MEMORIJU I KORISTITI VRIJEDNOSTI VARIJABLI.

REZULTATA

RAČUNAR NEĆE URADITI NIŠTA ŠTO MU NISMO NAREĐILI TAKO DA JE OBAVEZNO NAREĐITI RAČUNARU DA ISPIŠE VRIJEDNOST VARIJABLE KOJA PAMTI ZBIR SA KOMANDOM PRINT

***Napisati program koji računa zbir 2 broja

Prije izrade ovog programa moramo razmisliti koji podaci su nam potrebni za izradu ovog programa.

Kada u obzir uzmemo matematičku formulu zbira brojeva $z=a+b$ vidimo da su nam potrebna 2 broja kako bi dobili rezultat koji se od nas traži. Za svaki podatak u programu potrebna nam je varijabla koju ćemo imenovati na svoj način kako bi nas podsjećale na podatke koje one pamte. Pa ćemo nazvati:

prvi sabirak sa „a“

drugi sabirak sa „b“

zbir će biti izračunat i zapamćen pod varijablom „c“.

Svaki program ovog karaktera ima 3 faze

1. Unos podataka
2. Obrada podataka
3. Ispis rezultata



Znak # (hashtag) u programu ima funkciju pisanja komentara i nije obavezan ali kada pišete velike programe bit će vam potrebno da komentirate pojedine dijelove programa kako bi ste kasnije mogli pronaći željeni dio programa.

prilikom pokretanja svaki program bi trebao da se prijavi

šta radi i koja njegova funkcija te kratko uputstvo kako koristiti taj program

to ćemo uraditi korištenjem funkcije print

print ("PROGRAM RAČUNA ZBIR 2 BROJA")

1. KORAK UNOS PODATAKA POTREBNIH ZA DOBIJANJE REZULTATA

U OVOM SLUČAJU TREBAMO UNIJETI 2 BROJA KOJA ĆEMO NAZVATI a i b

UNOS PODATAKA VRŠIMO UZ POMOĆ
FUNKCIJE INPUT

a=int(input("unesite prvi broj "))

b=int(input("unesite drugi broj"))

UNOS
PODATAKA

2. KORAK OBRADA PODATAKA

VRLO JEDNOSTAVNO POTREBNO JE UPISATI
ODGOVARAJUĆI MATEMATIČKI IZRAZ
(FORMULU)

c=a+b

OBRADA
PODATAKA

3. KORAK ISPIS REZULTATA U OVOM
SLUČAJU VARIJABLA c PAMTI NAŠ REZULTAT

print ("zbir brojeva iznosi ", c)

varijable se nikada ne stavljaju pod znake

ISPIS
REZULTATA

navodnika

**** Program koji računa površinu i obim kvadrata :

predstavljanje programa

Print(„program računa površinu kvadrata“)

#unos podataka

a = int(input(„unesite dužinu stranice a “))

obrada podataka

p = a2** *#kvadriranje varijable a može se pisati i $p=a*a$*

o = 4*a

ispis rezultata

print („površina kvadrata iznosi“)

print(p)

print („obim kvadrata iznosi“, o)

podaci se mogu ispisati i sa jednom komandom print

print („površina kvadrata iznosi „ , p)

kada se startuje program
će ispisati šta radi

Za računanje površine i
obima računar treba preko
tastature unijeti dužinu
stranice a

Kada se izvrše računske
operacije varijable "p" i "o" će
dobiti neku brojčanu vrijednost
i bit će zapamćene u

Ispis rezultata uz pomoć
komande print. Varijable nisu
pod znacima navoda. Što znači da
će biti ispisani brojevi

Zadaci :

- 1) Napisati program koji računa zbir 3 broja sa decimalnim zarezmom (**float**).
- 2) Napisati program koji računa površinu pravougaonika.
- 3) Napisati program koji računa površinu i obim pravougaonika.
- 4) Napisati program koji računa jačinu struje u kolu koristeći Ohmov zakon $I=U/R$ (koristiti varijable koje mogu imati decimalni zarez **float**)

Komande za donošenje odluka u programu

Komanda IF (prosti oblik)

Svako inteligentno biće donosi odluke u zavisnosti od situacije u kojoj se nalazi uzimajući u obzir stvari iz svoje okoline. Tako kada pada kiša mi koristimo kišobran. U ovom slučaju da bi smo koristili kišobran treba da pada kiša. Znači „**kiša**“ je uslov za vršenje radnje „**nošenje kišobrana**“.

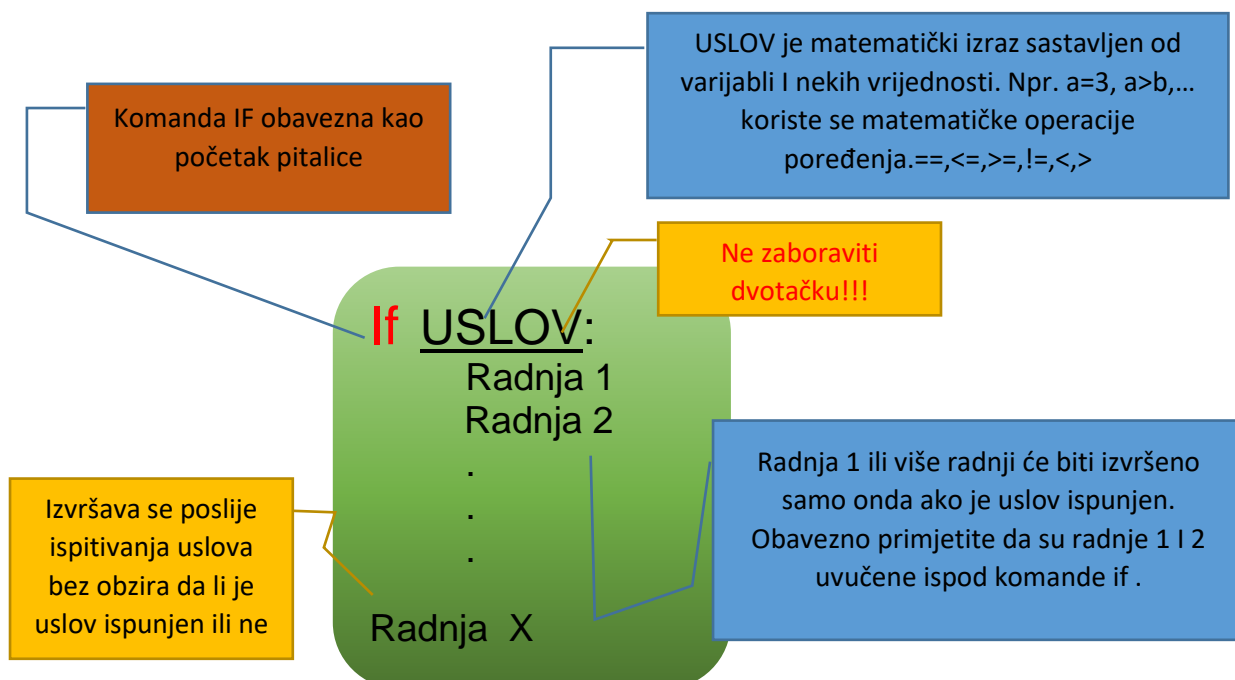
Kod računara situacija je slična postoji uslov i radnja koja se izvršava te naravno i komanda koja sadrži ova dva elementa. Komanda se naziva IF (ako) . Pa često ovu komandu nazivamo **pitalica**. Šta ispituje pitalica? Kada pokrenemo program njegova memorija je prazna što znači da moramo u memoriju računara unijeti neke podatke , rekli smo da su **varijable** one koje pamte podatke. Što znači da **pitalica** ispituje vrijednosti **varijabli** i na osnovu tih vrijednosti dešavaju se neke radnje.

Komandi IF koristimo kada želimo da uslovimo vršenje jedne ili više radnji.

Komanda if se sastoji od varijable i matematičkih znakova poređenja.

MATEMATIČKA OZNAKA	OZNAKA U PROGRAMU	ZNAČENJE
=	==	JEDNAKO (IDENTIČNO)
<	<	MANJE
>	>	VEĆE
>=	>=	VEĆE ILI JEDNAKO
<=	<=	MANJE ILI JEDNAKO
≠	!=	RAZLIČITO

OPŠTI OBLIK KOMANDE IF:



*** Primjer:

Napisati program koji će od korisnika tražiti da unese ispisati koji je od tih brojeva veći.

2 broja i

PAZITE NA PRAVOPIISNE GREŠKE

```
Print(„program upoređuje 2
broja“)
Print(„*****“)
#unos brojeva
a=int(input(„unesite 1. broj:“))
b= int(input(„unesite 2. broj:“))
# provjera unešenih brojeva
if a<b:
    print(„drugi broj je veći od
    prvog“)
    print(„*****“)
if a>b:
    print(„prvi broj je veći od
    drugog“)
    print(„*****“)
if a==b:
    print(„brojevi su jednaki“)
    print(„*****“)
print („kraj programa“)
```

Unos brojeva preko tastature.
Unešene brojeve će pamtit
varijable a i b

Ako je $a < b$ onda ispiši poruku „drugi broj je veći od prvog“ i ispiši zvjezdice. Ako ne nastavi dalje

Slijedi sljedeća provjera da li je $a > b$ i ako jeste ispiši poruku koja je uvučena a ako nije idi na sljedeću komandu

Sljedeća komanda je ispitivanje da li su varijable a i b identične. Ako jesu ispiši poruku a ako nisu uradi sljedeću komandu. Obratite pažnju na komandu print koja nije uvučena

PITANJA:

Koliko imamo varijabli ?
Kako se zovu ?
Šta pamte ?
Koliko imamo uslova (pitalica) ?
Kako su napisani ?
Gdje se određuje gdje će biti ispisana poruka da je prvi broj veći od drugog ?
Koji je uslov da se ispiše poruka **kraj programa** ?

*** Napisati program koji će od korisnika tražiti da unese šifru te ispisuje kada je šifra tačno upisana.
(Tačna šifra je 123).

Da razmislimo prije rada, šta bi trebalo da se desi :

1. Tražiti od korisnika da unese šifru
2. Provjeriti da li je upisana šifra identična sa tačnom šifrom
3. Ispisati odgovarajuću poruku
4. Završiti program

uvodni dio

Print („program provjerava tačnost upisane šifre“)

unos podataka

x=int(input(„unesite šifru“))

provjera šifre

#ako je vrijednost varijable x identična 123 to jest pravoj šifri tada ispiši poruku

If x==123:

Print(„unešena je prava šifra“)

Print („kraj programa“)



* Napomena obratite pažnju na zadnje 2 print komande i gdje se nalaze u programu.
Naravno ovaj program se može napisati na ljepši način, jer očigledno fali i poruka da je upisana pogrešna šifra. Dovršiti program sa odgovarajućom if pitalicom.

*** Napisati program koji će provjeriti da li je unešeni broj paran.

print ("program provjerava da li je unešeni broj paran")

unos broja

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

provjeriti da li je ostatak dijeljenja

sa brojem 2 jednak nuli a%2 modul koji računa ostatak dijeljenja broja a sa 2

if a%2==0:

print ("broj je paran")

provjera da li je ostatak dijeljenja

sa brojem 2 nije jednak (!=) nuli

if a%2!=0:

print("broj nije paran")

print ("kraj programa")

*** jednostavniji i brži način sa korištenjem IF ELSE

print ('program provjerava dali je broj paran')

a=int(input('Upisi broj: '))

if a%2==0: _____
print ('Broj je paran')

else:
print ('Broj je neparan')

a%2 ova funkcija daje ostatak dijeljenja broja a sa 2. Ako je ostatak dijeljenja varijable a sa 2 jednak 0 .

**** Napisati program koji će po želji korisnika računati

- 1- Jačinu struje u električnom kolu $I=U/R$
- 2- Jačinu napona u električnom kolu $U=I \cdot R$
- 3- Jačinu otpora u električnom kolu $R=U/I$

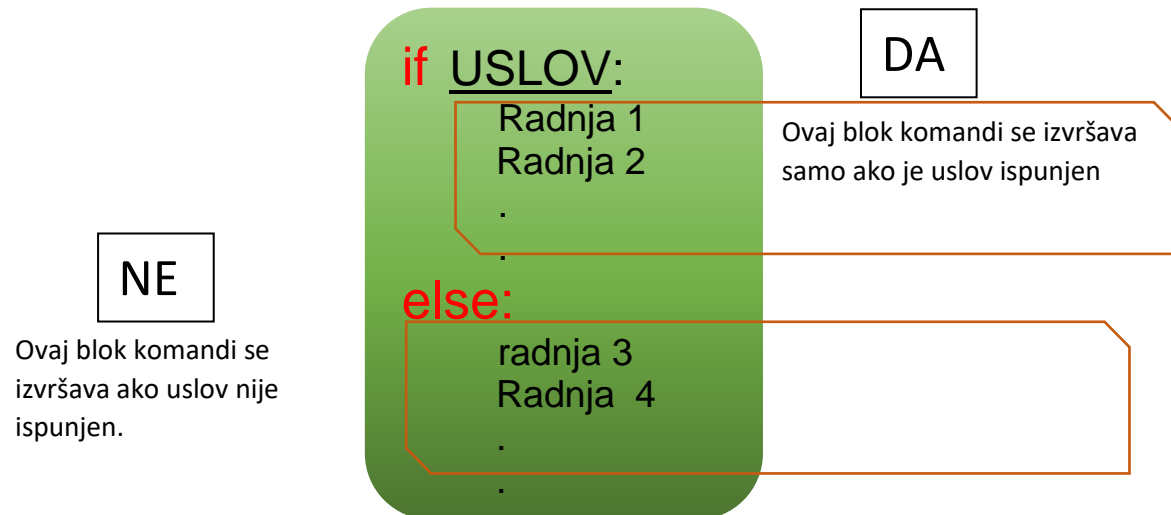
Ovo je program koji u sebi sadrži više malih programa čije će pokretanje kontrolisati korisnik sa unosom brojeva 1,2 ili 3 .

```
# uvodni izbornik
print("unesite broj 1 za računanje jačine struje (I)")
print("unesite broj 2 za računanje jačine napona (U)")
print("unesite broj 3 za računanje jačine otpora (R)")
#tražit ćemo od korisnika da unese željenu operaciju
i=input("Unesite željenu operaciju 1-3 ")
#provjera šta je korisnik unio
if i=="1":
    print("program računa jačinu struje (I)")
    U=float(input("Unesite jačinu napona u voltima"))
    R=float(input("unesite jačinu otpora u omima"))
    I=U/R
    print("Jačina struje iznosi I=",I," A")
if i=="2":
    print("program računa jačinu napona (U)")
    #dovršite sami ovaj program
# po mogućnosti upozoriti korisnika da unio pogrešne
vrijednosti
# 1,2,3
#if i!="1" or i!="2" or i!="3":
#    print("unijeli ste pogrešan broj ")
#    print("možete izabrati samo 1,2 ili 3")
#kraći oblik bez provjere
#else:
#    print(„Unijeli ste pogrešan izbor 1-3“)
```

IF ELSE

Prevedeno na jezik ljudi, **AKO** je uslov ispunjen radi 1. radnju **INAČE** ako nije ispunjen uslov radi radnju 3. Ovdje smo u mogućnosti da sa 1 pitalicom donesemo odluke u 2 ili više mogućih situacija. Sa ovim oblikom se treba pažljivo postupati i koristiti samo u situacijama gdje imamo izbor TAČNO ili NETAČNO. Inače može dovesti do pogrešnih radnji u programu.

OPŠTI OBLIK KOMANDE IF - ELSE



*** Napisati program koji će od korisnika tražiti da unese šifru te ispisuje kada je šifra tačno unešena. (Tačna šifra je 123).

Da razmislimo prije rada, šta bi trebalo da se desi :

5. Tražiti od korisnika da unese šifru
6. Provjeriti da li je unešena šifra identična sa tačnom šifrom
7. Ispisati odgovarajuću poruku
8. Završiti program

uvodni dio

Print („program provjerava tačnost unešene šifre“)

unos podataka

x=int(input(„unesite šifru“))

provjera šifre

#ako je vrijednost varijable x identična 123 to jest pravoj šifri tada ispiši poruku

if x==123:

print(„unešena je prava šifra“)

else:

print(„unijeli ste neispravnu šifru“)

Print („kraj programa“)

**** Napisati program koji će od korisnika tražiti da unese šifru te ako šifra nije ispravna program će ugaziti računar a ako je šifra ispravna pokrenut će facebook stranicu.

Napomena za pisanje ovog programa možemo koristiti kod iz prethodnog programa (copy/paste) te ga jednostavno proširiti sa određenim funkcijama. Kako u samom programskom jeziku ne postoji funkcija za gašenje računara niti funkcija za otvaranje web stranice morat ćemo koristiti programske biblioteke koje omogućuju željene funkcije. Python je programski jezik koji ima preko 137 000 programskih biblioteka koje omogućuju mnoštvo dodatnih funkcija iz raznih oblasti života. Prije upotrebe u programu biblioteke se moraju prethodno instalirati na računar sa komandom "pip install" u konzoli ili direktno iz editora kao je što je Pycharm koji omogućava direktnu instalaciju biblioteka bez korištenja pip install. Za gašenje računara koristit ćemo biblioteku "os" a za otvaranje stranice koristit ćemo biblioteku "webbrowser"

```
#učitavanje potrebnih biblioteka
import webbrowser
import os
# tražimo od korisnika da unese šifru
# tačna šifra je "a123"
a=input("Unesite šifru: ")
# provjeravamo šifru to jest vrijednost varijable a
if a=="a123":
    webbrowser.open("facebook.com")
    # nova komanda iz biblioteke webbrowser (open)
else:
    os.system('shutdown /p /f')
    # nova komanda iz biblioteke os gasi windows OS
```

*** Napisati program koji unosi jedan broj te ispisuje poruku je li učitani broj djeljiv s 5 ili ne.

```
a=int(input('Upisi broj: '))

if a%5==0:

    print ('Broj je djeljiv s 5')

else:

    print ('Broj nije djeljiv s 5')
```


*** Napisati program koji će postaviti 2 pitanja i provjeriti da li je odgovor tačan ili ne (kviz).

```
# kviz
print("program postavlja pitanja i provjerava odgovore")
print("*****")

#postavljanje 1. pitanja i provjera odgovora
#pitanje ispisujemo sa komandom print
print("Koji je glavni grad BiH")
a=input("unesite odgovor: ")

#provjera unešenog odgovora
if a=="Sarajevo":
    print("vaš odgovor je tačan")
else:
    print("vaš odgovor je netačan")

#postavljanje i provjera 2. pitanja
b=input("kako se zove najviši vrh BiH: ")
if b=="Maglić":
    print("vaš odgovor je tačan")
else:
    print("vaš odgovor je netačan")
print("kraj kviza")
```

*** napisati isti ovaj program koji će na kraju ispisati broj tačnih odgovora te procenat tačnih odgovora. Kasnije može se dati i ocjena za određeni postotak tačnih odgovora.



U principu program će biti isti kao i gore napisani program samo što ćemo uvesti nešto novo a to su brojači. Brojači imaju veliku primjenu u programiranju stoga su veoma važni. Brojači se obično pišu u obliku $b=b+1$... uvećavamo stanje brojača za 1. $b=b+1$ u kraćem obliku se piše $b+=1$.

U našem slučaju bit će 2 brojača

Brojač tačnih odgovora (bt)

Brojač netačnih odgovora (bn)

Tako da ćemo za svaki dat tačan odgovor brojač bt povećati za jedan $bt+=1$ a u suprotnom slučaju brojač netačnih odgovora se povećava za $bn+=1$.

Kada saberemo broj tačnih i broj netačnih odgovora dobijamo ukupan broj pitanja (bp) . $bp=bn+bt$
to će nam kasnije biti potrebno da izračunamo procenat (p) tačnih odgovora po formuli
 $p=bt/bp*100$.

```
# kviz
print("program postavlja pitanja i provjerava odgovore")
print("*****")
```

```
#postavljanje 1. pitanja i provjera odgovora
#pitanje ispisujemo sa komandom print
print("Koji je glavni grad BiH")
a=input("unesite odgovor: ")
```

```

#provjera unešenog odgovora
if a=="Sarajevo":
    print("vaš odgovor je tačan")
    bt+=1
else:
    print("vaš odgovor je netačan")
    bn+=1
#postavljanje i provjera 2. pitanja
b=input("kako se zove najviši vrh BiH: ")
if b=="Maglić":
    print("vaš odgovor je tačan")
    # dopunite nedostajuće komande
else:
    print("vaš odgovor je netačan")
    # dopunite nedostajuće komande

# računanje ukupnog broja pitanja bp
# ovaj dio programa ide na kraj kviza
bp=bn+bt
# računajne procenta tačnih odgovora
p=bt/bp*100

print („imate „ , p , “ tačnih odgovora“)
print ("kraj kviza")

```

Svaki put moramo povećati brojač.

*** Napisati program koji će na osnovu procenta tačnih odgovora ispisati odgovarajuću ocjenu.

Maksimalan broj poena je 100.

Ocjena	1	2	3	4	5
Procentat	0 - 40	41 - 55	56 – 70	71 - 85	86 -10

Ovaj problem se može riješiti na više načina i specifičan je po tome što vrijednost varijable procenat provjeravamo na drugačiji način. Da bi učenik dobio ocjenu 1 on mora imati vrijednost varijable procenat veći ili jednak (\geq) od 0 i takođe manji ili jednak (\leq) od 40. IF pitalice koje provjeravaju da li vrijednost jedne varijable zadovoljava 2 ili više uslova se tvore uz pomoć logičkog operatera AND. U našem slučaju pitalica bi se napisala ovako *if procenat \geq 0 and procenat \leq 40*. Kako imamo 5 ovakvih slučajeva za ocjene od 1 do 5 moramo provjeriti 5 puta vrijednost varijable procenat.

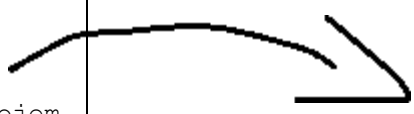
PROGRAM 1

```
#unos podataka
procenat=int(input("Unesite procenat
tačnih odgovora"))

#obrada podataka
# u našem slučaju ispitujemo u kojem
se opsegu nalazi varijabla procenat
# nužno je ispitati sve moguće
situacije
```

```
#provjera za ocjenu 1
if procenat>=0 and procenat<=40:
    print ("ocjena: nedovoljan (1)")
#provjera za ocjenu 2
if procenat>=41 and procenat<=55:
    print ("ocjena: dovoljan (2)")
#provjera za ocjenu 3
if procenat>=56 and procenat<=70:
    print ("ocjena: dobar (3)")
#provjera za ocjenu 4
if procenat>=71 and procenat<=85:
    print("ocjena: vrlo dobar (4)")
#provjera za ocjenu 5
if procenat>=86 and procenat<=100:
    print ("ocjena: odličan (5)")
print ("kraj programa")
```

PROGRAM 2



```
#provjera za ocjenu 1
if procenat>=0 and procenat<=40:
    print ("ocjena: nedovoljan
(1)")
#provjera za ocjenu 2
elif 41 <= procenat <= 55:
    print ("ocjena: dovoljan (2)")
#provjera za ocjenu 3
elif procenat>=56 and procenat<=70:
    print ("ocjena: dobar (3)")
#provjera za ocjenu 4
elif procenat>=71 and procenat<=85:
    print("ocjena: vrlo dobar (4)")
#provjera za ocjenu 5
else:
    print ("ocjena: odličan (5)")
print ("kraj programa")
```

Korištenjem komande elif umjesto if u drugom programu smo ubrzali izvođenje programa. Zbog toga što će se u prvom programu 5 puta ispitati varijabla procenat i biti će ispisana poruka koja odgovara vrijednosti varijable procenat. U drugom slučaju varijabla procenat će biti ispitana sve dok ne zadovolji kriterije i daljnje provjere neće biti urađene nego će se odmah ispisati poruka kraj programa.

Razmislite: kako još da ubrzate ovaj program? Izmjenom mjesta elif komandi uzimajući u obzir broj ocjena 4 i 3 te 5 i 1 na testovima. Komentarišite tekst markiran žutom bojom.

Ovaj programski zadatak se može riješiti na još načina ali to će te naučiti kasnije.

*** Napisati program koji će simulirati kalkulator. Program će tražiti da korisnik unese 1. broj a zatim operator (+, -, *, /) te nakon toga će računar tražiti da korisnik unese 2. broj. I na kraju će ispisati rezultat. Ovaj problem ćemo također riješiti na više načina gdje će te upoznati neke nove metode unosa podataka te funkciju case. Na ovaj način u jednom programu možemo po izboru korisnika pokrenuti više programa.

```

# Unos prvog broja
broj1 = float(input("unesite prvi broj"))

# Unos operatora (+, -, *, /)
operator = input()

# Unos drugog broja
broj2 = float(input("unesite drugi broj"))

# Izvršenje odgovarajuće operacije i ispis rezultata
if operator == '+':
    print(broj1 + broj2)
elif operator == '-':
    print(broj1 - broj2)
elif operator == '*':
    print(broj1 * broj2)
elif operator == '/':
    if broj2 != 0:
        print(broj1 / broj2)
    else:
        print("Greška")
else:
    print("Greška")

```

*** Drugi način, u kojem ćemo primjeniti nove metode. Prva metoda je unos više varijabli u jednom redu. Brojevi će biti unešeni i zapamćeni kao znakovna varijabla i u toku izvršavanja programa taj broj će biti pretvoren u decimalni broj (float). Druga metoda jeste korištenje funkcije *CASE* koja je ubačena u python tek od verzije 3.10. Tu funkciju koristimo kada želimo ispitati vrijednosti jedne varijable koja može imati više stanja. I naravno izvršiti više radnji ili programa.

```

# Unos varijabli i operatora u jednom redu
unos = input("Unesite izraz (npr. 3 + 5) sa razmacima : ")
# Razdvajanje unosa na delove
broj1, operator, broj2 = unos.split()
# Konvertovanje brojeva u float
broj1 = float(broj1) #zašto nije int umjesto float?
broj2 = float(broj2)
#komentarisati varijablu broj2 i broj1 šta se dešava sa prijašnjim
stanjem varijabli broj1 i broj2

```

```

# Izvršenje odgovarajuće operacije koristeći match-case
match operator:

```

```

    case '+':
        print(broj1 + broj2)
    case '-':
        print(broj1 - broj2)
    case '*':
        print(broj1 * broj2)
    case '/':
        if broj2 != 0:
            print(broj1 / broj2)
        else:
            print("Greška")
    case _:
        print("Greška")

```

Ovaj način provjere više stanja jedne varijable je poželjan zbog veće preglednosti napisanog koda.

Ovo je jedan od načina da se izbjegne situacija da neki broj podijelimo sa nulom. Kada se to desi program se odmah zatvara i prijavljuje grešku. To nije poželjno.

Kada malo bolje naučite programirati ovaj isti program će te napisati ovako.

```
izraz = input("Unesite matematički izraz (npr. 2+3): ")
rezultat = eval(izraz)
print("Rezultat je:", rezultat)
```

Ovaj program će raditi čak i složene izraze sa zagradama i više brojeva npr. $(10+3)/4+15$ ali kada upišete $10/0$ doći će do pada programa i greške **ZeroDivisionError: division by zero**.

Da bi smo izbjegli tu neugodnu situaciju poslužit ćemo se naprednim funkcijama za kontrolu grešaka koje mogu nastati prilikom unosa pogrešnih podataka program.

Za to ćemo koristiti try / except kao sredstvo za iznimno ponašanje programa prilikom nastanka nekih grešaka. **Potražite na internetu malo više o tome.**

```
izraz = input("Unesite matematički izraz (npr. 2+3): ")

try:
    rezultat = eval(izraz)
    print("Rezultat je:", rezultat)
except ZeroDivisionError:
    print("nije moguće izračunati ovaj izraz")
```

Ako dođe do situacije da korisnik upiše $10/0$ računar će pokušati da izračuna (**try:**) i ako se pojavi greška dijeljenja sa nulom (**except ZeroDivisionError:**) on će ispisati poruku: **nije moguće izračunati ovaj izraz**.

Ovaj program je skoro pa odličan, problem je u tome što ga moramo svaki put startovati da bi smo izračunali neki matematički izraz. Da bi smo postigli da naš program radi kao digitron i da stalno čeka da se unesu brojevi i da računa bez stalnog pokretanja programa iz početka, mi ćemo uraditi sljedeće. Staviti ćemo ovaj gore napisani kod u beskonačnu petlju koja će stalno ponavljati napisani kod unutar petlje. U ovom slučaju namjerno pravimo grešku koju ćemo ispraviti kasnije. Beskonačna petlja počinje sa **while True:** ostatak koda se nalazi uvučen u petlju.

```
while True:
    izraz = input("Unesite matematički izraz (npr. 2+3): ")
    try:
        rezultat = eval(izraz)
        print("Rezultat je:", rezultat)
    except ZeroDivisionError:
        print("nije moguće izračunati ovaj izraz")
```

Kao što je rečeno ovaj program će se odvijati beskonačno i jedini način da ga prekinemo jeste da pritisnemo enter prilikom unosa izraza. Tada će se pojaviti greška da program ne može izračunati taj izraz. (**novi bug (greška) u našem programu**). Pokušajte samo pritisnuti tipku enter.... Ovo je odlična situacija gdje je nije ništa upisano u varijablu **izraz**, to ćemo iskoristiti kao sredstvo za izlaz iz beskonačne petlje.

Postoji više načina kako provjeriti da li je neka znakovna varijabla prazna to jest da se u njoj ne nalazi ništa. Mi ćemo koristiti način da provjerimo koliko znakova se nalazi u varijabli. Funkcija koja nam daje broj znakova u varijabli se zove **len(izraz)**. Ova funkcija će izbrojati koliko se znakova nalazi u varijabli **izraz**. To ćemo provjeriti tako što u programu provjerimo u pitalici (if).

```
if len(izraz)==0:
    break          #izlaz iz beskonačne petlje
```

Ovaj kod ćemo ubaciti odmah nakon unosa varijable **izraz** ... **pokušajte ga ubaciti na kraj programa.**

```

while True:
    izraz = input("Unesite matematički izraz (npr. 2+3): ")
    if len(izraz)==0:
        break
    try:
        rezultat = eval(izraz)
        print("Rezultat je:", rezultat)
    except ZeroDivisionError:
        print("nije moguće izračunati ovaj izraz")
print ("kraj programa")

```

Skoro pa savršen program, ostalo nam je da spriječimo korisnika da unosi slova umjesto brojeva. Ako unesemo a+b program će pasti i ispisati grešku. I program jednu manu a to je kada napišemo 10/3 kao rezultat će se pojaviti ovaj broj Rezultat je: 3.3333333333333335 ova decimala sa brojem 5 nije tačna. Trebale bi biti sve trojke. To je greška u Pythonu koja se povlači već godinama i ne može se ispraviti bez korištenja specijalnih biblioteka.

```

from decimal import *
print(Decimal(10)/Decimal(3))

```

Ovaj naš program ima dvije mane koje moramo ispraviti kako bi program bio savršen.

Problem dijeljenja 10/3 ćemo riješiti sa time da rezultat ispisujemo sa samo 3 decimalna mjesta.

```
print(f"Rezultat je: {rezultat:.3f}")
```

Problem unosa pogrešnih podataka u izraz riješit ćemo tako, što ćemo ignorirati sve moguće greške tako da program neće pokušavati da izračuna ono što se ne može računati, nego će samo ispisati vrstu greške.

```

except Exception as e:
    print("Greška u izrazu: ",e)

```

#e je vrsta greške

Tako da naš program postaje skoro pa savršen ...

```

while True:
    izraz = input("Unesite matematički izraz (npr. 2+3): ")

    if len(izraz) == 0: # Ako je unos prazan, prekini program
        break

    try:
        rezultat = eval(izraz)
        # Formatiranje rezultata sa 3 decimale 3f
        print(f"Rezultat je: {rezultat:.3f}")
    except ZeroDivisionError:
        print("Nije moguće izračunati ovaj izraz (dijeljenje sa nulom).")
    except Exception as e:
        print("Greška u izrazu: ",e)

print("Kraj programa")

```


Postoji još jedan mali problem a to je kako pokrenuti ovu aplikaciju kod prijatelja koji nema instaliran python.

Oni koji koriste Linux OS neće imati takvih problema jer sa instalacijom Linux-a na vaš računar se instalira i Python programski jezik. Tako da možete slobodno kopirati svoje python skripte i prenositi na druge Linux računare.

Korisnici Windows OS-a će morati napraviti izvršni fajl (exe) kako bi mogli pokrenuti svoj Python program na svakom Windows računaru.

Postoji više metoda za izradu exe fajla, ovdje je jedna od tih koja je vrlo jednostavna i slikovita.

<https://pypi.org/project/auto-py-to-exe/>

Na internetu potražite tutorijal za auto-py-to-exe

Ovako izgleda tok izrade jedne aplikacije od planiranja do kodiranja i ispravljanja grešaka te uklanjanje potencijalnih problema.

Sada možemo napraviti i grafičku verziju istog programa.

Uputstvo za pravljenje GUI kalkulatora u Pythonu koristeći tkinter

Kako napraviti python program koji se ne izvršava u terminalu, nego liči na klasični program koji se otvara u prozoru i ima polja za unos i komandna dugmad te dugme za zatvaranje aplikacije. Za programe koji izgledaju tako bit će nam potrebna **tkinter** biblioteka koja u sebi sadrži potrebne komande za izradu radnih prozora. U suštini naš programski dio će ostati isti samo što ćemo ga uvrstiti u dio programa koji

1. Crta radni prozor
2. Definiše polje za unos podataka
3. Određuje polje za ispis rezultata
4. Definiše dugme izračunaj koje aktivira funkciju za računanje (naš kod programa)

1. Šta je tkinter?

tkinter je standardna Python biblioteka za pravljenje GUI (grafičkih) aplikacija. Pomoću nje možete kreirati prozore, dugmad, polja za unos, etikete i druge grafičke elemente za izvršavanje programa .

2. Instalacija

Ako koristite Python 3, tkinter je već instaliran. Nema potrebe za dodatnim koracima.

Da biste provjerili da li je sve ispravno:

1. Otvorite Python interaktivni terminal.
2. Ukucajte:

```
import tkinter
```

```
print("tkinter radi!")
```

Ako ne dobijete grešku, sve je spremno.

Korak 1: Uvoz biblioteka

Prvo uvozimo potrebne biblioteke:

```
import tkinter as tk
from tkinter import messagebox
# tk je skraćenica za tkinter.
```

#messagebox omogućava prikazivanje dijaloga za obavještenja, poput upozorenja.

Korak 2: Kreiranje glavnog prozora

Dodajte sledeći kod za pravljenje glavnog prozora aplikacije:

```
root = tk.Tk()
root.title("Kalkulator matematičkih izraza")
root.geometry("500x300")
```

#tk.Tk() kreira glavni prozor.

#title() postavlja naslov prozora.

#geometry() postavlja veličinu prozora na **400x200 piksela**.



Korak 3: Određivanje vrste i veličine fonta

Dodajemo zajednički font koji ćemo koristiti:

```
font = ("Helvetica", 20)
#"Helvetica" je naziv fonta.
#20 je veličina slova.
```

Korak 4: Dodavanje elemenata

Dodajemo osnovne elemente korisničkog interfejsa.

Polje za unos:

```
unos_label = tk.Label(root, text="Unesite izraz:", font=font)
unos_label.pack(pady=20)
```

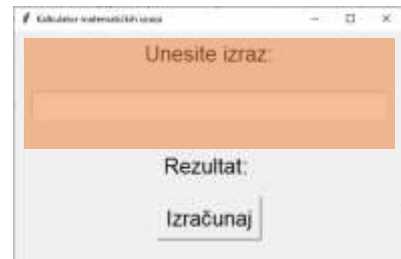
```
unos = tk.Entry(root, font=font, width=30)
unos.pack(pady=20)
```

#Label: Koristi se za tekstualne oznake.

#Entry: Omogućava unos teksta.

#pack(): Postavlja elemente jedan ispod drugog.

#pady= :vertikalni razmak od sljedećeg elementa



Polje za prikaz rezultata:

```
izlaz = tk.Label(root, text="Rezultat: ", font=font)
izlaz.pack(pady=20)
```

#izlaz je natpis koji prikazuje rezultat računanja.



Korak 5: Dugme za izračunavanje

Dodajemo dugme i funkciju „*izracunaj*“ koja se pokreće kada korisnik klikne na dugme. U funkciji koja se zove „*izracunaj*“ je smješten kod našeg programa koji radi u konzoli. Umjesto komande input imamo funkciju `unos.get` koja povlači tekst matematičkog izraza iz polja unos

Funkcija:

```
def izracunaj():
    izraz = unos.get()
    try:
        rezultat = eval(izraz)
        izlaz.config(text=f"Rezultat: {rezultat:.3f}")
    except Exception as e:
        messagebox.showerror("Greška", f"Došlo je do greške: {e}")
```

- **get():** Dobija unos korisnika iz polja **unos**.
- **eval():** računa matematički izraz (npr., $2+3*4$).
- **try-except:** Hvata greške poput dijeljenja nulom ili nevažećih izraza.
- **config():** Ažurira tekst etikete rezultat_label.

Dugme Izračunaj:

```
dugme_izracunaj = tk.Button(prozor, text="Izračunaj",
                             command=izracunaj, font=("Arial", 20))
dugme_izracunaj.pack(pady=10)
```

- Button kreira dugme.
- command povezuje dugme sa funkcijom izracunaj.



Korak 6: Pokretanje aplikacije

Dodajte sledeći kod na kraj fajla:

```
prozor.mainloop()
```

- **mainloop()** pokreće izvršavanje napisanog koda kao aplikaciju.

Kada pokrenete program otvorit će se prozor aplikacije. Zatim unesite matematički izraz (npr., $5+3*2$) i kliknite na dugme **Izračunaj**.

5. Testiranje

Svaki program se mora testirati na ispravnost rada.

Probajte slijedeće izraze:

- Ispravni izrazi:
 - $2+3$
 - $5/2$
 - $10*(2+3)$

- Greške:
 - Prazno polje: dobijate upozorenje da unesete izraz.
 - Dijeljenje nulom (1/0): Prikazuje poruku *Greška: Dijeljenje nulom.*
 - Pogrešan unos (abc+2): Prikazuje grešku.

6. Moguća poboljšanja

- Dodajte dugme za resetovanje polja.
- Kada pročitate malo više o funkciji **eval** doći će te do saznanja da ta funkcija može pokrenuti zlonamjerni kod koji je unaprijed pripremljen. Tako da možete zamijeniti **eval** sigurnijom bibliotekom kao što je **sympy**.

Kod cijelog programa bez komentara:

```
import tkinter as tk
from tkinter import messagebox

prozor = tk.Tk()
prozor.title("Kalkulator matematičkih izraza")
prozor.geometry("500x300")

font = ("Helvetica", 20)
unos_label = tk.Label(prozor, text="Unesite izraz:",
font=font)
unos_label.pack(pady=10)

unos = tk.Entry(prozor, font=font, width=30)
unos.pack(pady=20)

izlaz = tk.Label(prozor, text="Rezultat: ", font=font)
izlaz.pack(pady=20)
def izracunaj():
    izraz = unos.get()
    try:
        rezultat = eval(izraz)
        izlaz.config(text=f"Rezultat: {rezultat:.3f}")
    except Exception as e:
        messagebox.showerror("Greška", f"Došlo je do greške:
{e}")

dugme_izracunaj = tk.Button(prozor, text="Izračunaj",
command=izracunaj, font=font)
dugme_izracunaj.pack()

prozor.mainloop()
```

ZA ONE KOJI ŽELE NAUČITI VIŠE

Ako želite naučiti više o pisanju GUI programa u pythonu pogledajte ovdje:

<https://pythonbasics.org/>

Skrolujte do **Tkinter** dijela.

Naravno youtube i mnoge druge stranice će vam ponuditi pregršt informacija i tutorijala za vaše GUI programe pisane u python programskom jeziku.

Pored tkinter biblioteke python podržava i druge biblioteke za pisanje GUI aplikacija. Na ovom linku možete naći opise tih biblioteka.

<https://www.geeksforgeeks.org/best-python-gui-frameworks-for-developers/>

Među njima je i Pygame biblioteka za pravljenje jednostavnih igrica.

Da bi ste napravili igricu morate ovladati petljama i klasama te komandama koje sadrži biblioteka Pygame.

Detaljno uputstvo za izradu jednostavne igrice:

<https://github.com/Dinolsanovic/Python-prvi-koraci/tree/main/pong>

Ako budete imali problema sa dodavanjem programskih linja u template ovdje se nalazi kompletan kod igrice. <https://github.com/Dinolsanovic/Python-prvi-koraci/tree/main/pong>

PETLJE

Petlje koristimo kada želimo jednu ili niz radnji ponoviti više puta.

Petlje u stvarnom životu možemo opisati kao puhanje gume na biciklu, sipanje 3 kašike šećera u šolju sa čajem, hodanje uz stepenice, vježbanje 10 sklekova. U stvari ponavljanje istih radnji tačno određeni broj puta ili dok se ne ispuni neki uslov.

Tako u Pythonu postoje 2 vrste petlji:

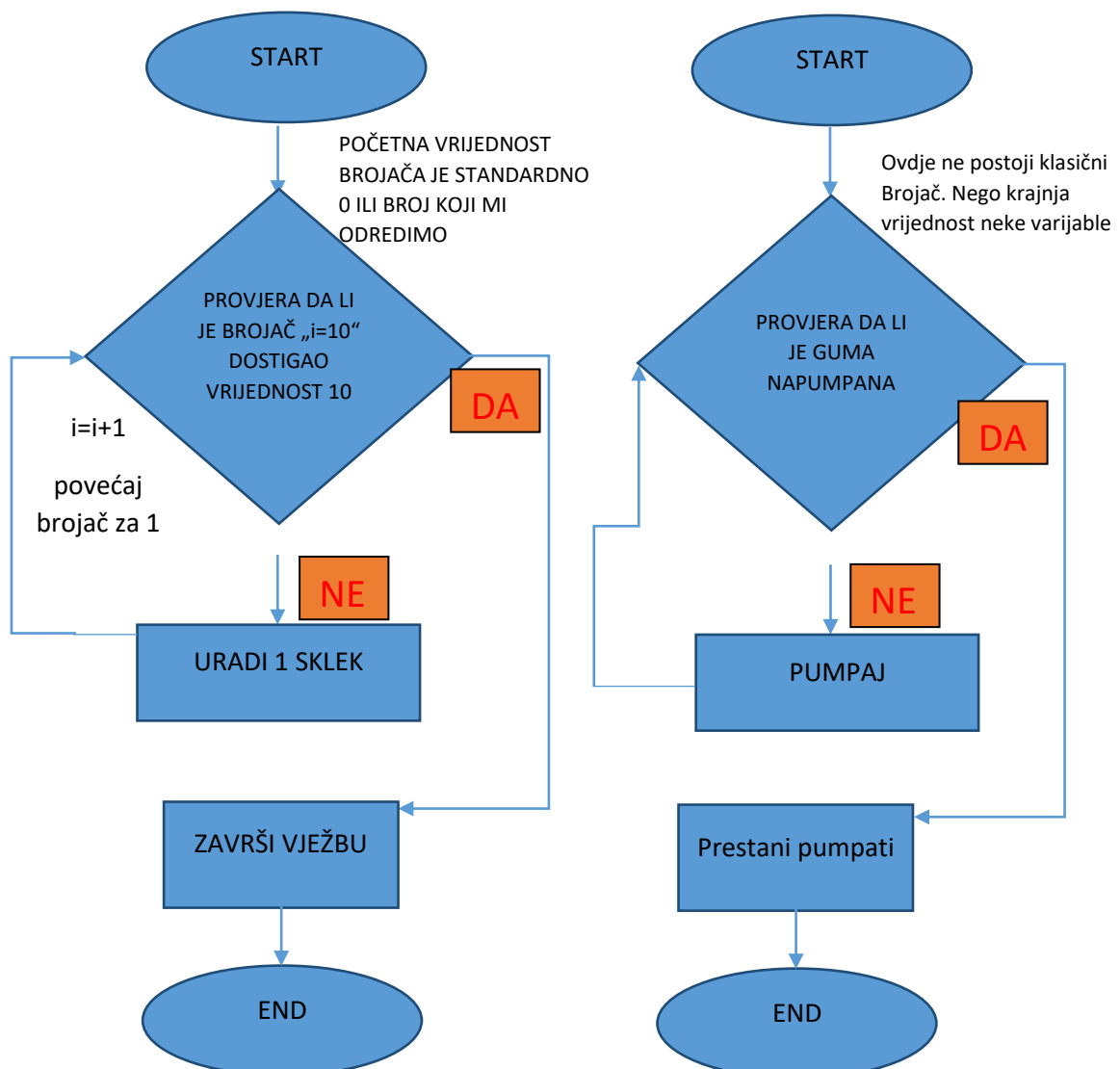
Petlje koje se ponavljanju tačno određeni broj puta. **For petlje**.

Petlje koje se ponavljaju dok se ne ispuni neki uslov. **While petlje**.

Sa kojom petljom bi opisali pumpanje gume?

Kao i kod IF pitalice ključnu ulogu igraju vrijednosti varijabli. Koje određuju broj ponavljanja ili uslov da se prekine ponavljanje.

Grafički prikaz petlje koja radi 10 sklekova i petlje koja pumpa gumu. Iz samog teksta vidimo da petlja koja radi 10 sklekova je for petlja a petlja koja pumpa gumu je while petlja.



Ako uporedimo grafički prikaz primjetit ćemo veliku sličnost u principu rada ove dvije petlje. Razlika je u tome što se brojač kod FOR petlje uvijek uvećava za istu vrijednost (1*). Kod WHILE petlje nemamo tačno određen broj koraka. Zato se WHILE petlja prevodi u smislu RADI DOK SE NE ISPUNI USLOV. Jer dok pumpamo gumu ne znamo tačno koliko puta moramo utisnuti zrak u gumu.

FOR PETLJA

For petlje koristimo kada neku radnju želimo ponoviti tačno određeni broj puta.

For petlje se koriste u više situacija. Prva situacija jeste prosto ponavljanje jedne ili više radnji.

```
for i in range(početak, kraj, korak):
```

** **i** je brojač koji će se uvećavati svaki put za vrijednost **korak** (ako ne napišemo onda je 1)

** **početak** je početna vrijednost brojača i. (ako ne napišemo onda je nula)

** **kraj** završna vrijednost petlje (obavezna) kada brojač dostigne **kraj** onda se petlja završava.

** **korak** je broj za koji se uvećava vrijednost brojača ako ne napišemo ovaj treći parametar onda se smatra da se brojač u svakom krugu uvećava za 1.

PRIMJERI:

*** napisati program koji 5 puta ispisuje riječ učiti. (*napomena u python-u postoji jednostavnije rješenje*).

U ovoj situaciji koristimo najprostiji oblik for petlje koja od parametara ima samo parametar kraj, kao krajnju vrijednost.

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

```
    print("Učiti")
```

```
print("Svaki dan")
```

Naziv funkcije range koja ima samo stop parametar 5 gdje će krajnja vrijednost brojača i dostići 5. „kraj“

Tijelo for petlje gdje se sve uvučene komande ponavljaju 5 puta.

1. Kada pokrenete ovaj program početna vrijednost brojača i će biti 0. Program će ispisati tekst **Učiti** zatim će povećati vrijednost brojača za 1 i provjeriti da li je brojač i=5. Ako nije onda ponavlja.

2. Vrijednost brojača je sada 1. program ispisuje riječ Učiti. Provjeriti da li je i=5 ako nije onda uvećava brojač za 1 (i=i+1 ili kraće napisano i+=1)

3. Petlja će se obrtati sve dok **i** (*brojač*) ne bude 5 ... Tada će se izvršiti prva komanda ispod tijela petlje. Print(„Svaki dan“).

*** Napisati program koji će ispisati brojeve od 1 do 5.

U ovoj situaciji koristit ćemo vrijednosti brojača i (*situacija gdje su nam potrebni nizovi brojeva*). To jest ispisat ćemo vrijednost brojača i na ekran sa komandom print (i). Ako se vratite na prethodni program i pročitate detaljno tekst vidjet ćete da vrijednost brojača počinje sa 0. Što nama ne odgovara. Zato koristimo proširenu verziju for range(start,end). Logički parametar start bio bi 1 a parametar end bio bi 5.

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

*pokrenite program te pronađite i objasnite grešku u ovom programu.

*Napišite ispravnu verziju ovog programa.

*** Napisati program koji će ispisati sve parne brojeve od 0 do 10.

Nula je paran broj, tako da ćete koristiti puni oblik for petlje. For i in range(start,end,step). Startna vrijednost bi bila 0, krajnja vrijednost bi bila 11 a ne 10 (*zaključak iz prethodnog zadatka*), step ili korak bi bio ??????. step je vrijednost za koju se uvećava brojač i. U prethodnim slučajevima step je standardno 1. gdje se brojač uvećava za 1. U ovom zadatku moramo imati drugačije povećanje brojača.

```
for i in range(0,11,2):  
    print (i)  
print ("Kraj programa")
```

Ovdje vidite da je treći parametar range funkcije 2 a to je korak ili step. Tako da će početna vrijednost brojača biti start a to je 0. A povećanje brojača će biti za 2. Tako da će u prvom prolazu biti ispisan broj 0 a u sljedećem prolazu brojač će se povećati za 2 (i=i+2 ili i+=2) tako da će u sljedećem krugu biti ispisan broj 2 i sve do broja 11 gdje vrijednost brojača neće nikada dostići krajnju vrijednost to jest broj 11 jer će se zaustaviti na broju 10.

Neki primjeri for petlji

Range funkcija	opis	primjer	izlaz
For i in range() print (i)			
range(kraj)	<ul style="list-style-type: none"> Ispisuje niz brojeva od 0 do 3. korak: 1 	range(4)	0, 1, 2, 3
range(početak, kraj)	<ul style="list-style-type: none"> Ispisuje niz brojeva od početka do kraja korak: 1 	range(0, 3)	0, 1, 2
		range(2, 6)	2, 3, 4, 5
		range(-13, -9)	-13, -12, -11, -10
range(početak, kraj, korak)	<ul style="list-style-type: none"> Ispisuje brojeve od početka do kraja sa različitim korakom korak: step 	range(0, 4, 1)	0, 1, 2, 3
		range(1, 7, 2)	1, 3, 5
		range(3, -2, -1)	3, 2, 1, 0, -1
		range(10, 0, -4)	10, 6, 2

Pogledajte ovaj film.

<https://www.youtube.com/watch?v=YhDI2btcWeU>

U nekim slučajevima nije potrebno štampati nizove brojeva nego koristiti numeričku vrijednost brojača kako bi dobili željeni rezultat.

*** Napisati program koji će izračunati zbir svih brojeva u intervalu koji odredi korisnik.

Napomena: Kako se u for petlji nalaze konkretne brojčane vrijednosti koje određuju način na koji se izvodi petlja. Tada konkretne vrijednosti petlje možemo zamijeniti varijablama. To jest vrijednostima koje unese korisnik preko tastature koristeći funkciju „input“.

1 - U našem slučaju moramo pitati korisnika da preko tastature unese dvije vrijednosti, prva vrijednost „start“ je startna vrijednost petlje a druga „stop“ je krajnja vrijednost petlje.

2- Ukupan zbir postavljamo da ima vrijednost 0. zbir=0. Ovu varijablu zbir koristimo da na nju dodajemo vrijednost brojača. Možemo je zamisliti kao vreću u koju ubacujemo brojeve. Matematički

prikazano izgledat će kao $zbir = zbir + i$. Gdje je i vrijednost brojača i u petlji a $zbir$ je prethodna vrijednost varijable $zbir$. Prije početka izvršavanja petlje vrijednost će bit $zbir=0$.

3- Definišemo petlju sa start i stop parametrima gdje uzimamo u obzir da petlja nikada ne ide do zadnje vrijednosti tako da ako želimo da koristimo i krajnju vrijednost stop parametar za petlju moramo povećati za 1. $stop = stop + 1$.

4- U svakom obrtanju petlje vrijednost brojača će se uvećavati za 1. i tu vrijednost brojača ubacujemo u vreću $zbir = zbir + i$ ($zbir += i$). *Zapamtite ovu metodu, često će vam biti potrebna.*

```
# Korisnik unosi početak i kraj intervala
pocetak = int(input("Unesite početak intervala: "))
kraj = int(input("Unesite kraj intervala: "))

# Inicijalizujemo promjenljivu za zbir
zbir = 0

# Petlja koja prolazi kroz sve brojeve u intervalu
for i in range(pocetak, kraj + 1): # kraj+1 da uključimo i zadnji broj
    zbir += i # Dodajemo broj u zbir ovo je identično zbir=zbir+i

# Ispis rezultata
print(f"Zbir brojeva od {pocetak} do {kraj} je: {zbir}")
```

Probajte u tijelu for petlje ubacite komandu `print(i)` kako bi ste vidjeli kako se vrijednost brojača i mijenja u svakom krugu.

*** Napisati program koji će od korisnika tražiti da unese 5 brojeva koje će računar sabrati i ispisati zbir tih brojeva.

1. Za unos određenog broja podataka koristimo for petlju koja će se okrenuti 5 puta.
2. U tijelu petlje pišemo komandu za unos jednog broja i dodajemo ga na ukupan zbir koji prije početka petlje ima vrijednost 0.
3. Ispisujemo rezultat zbir.

```
# postavljamo zbir na 0
zbir = 0

# Petlja za unos 5 brojeva
for i in range(5):
    broj = int(input("Unesite broj: ")) # Unos broja
    zbir += broj # Dodajemo broj u zbir

# Ispis rezultata
print("Zbir unešenih brojeva je:", zbir)
```

Ovaj program možemo poboljšati tako da korisnik može unijeti po želji koliko brojeva on želi sabrati. Takođe možemo poboljšati stabilnost programa tako što ćemo ubaciti dio koda koji provjerava da li je korisnik unio pogrešan broj ili slovo. Jer program radi samo sa cijelim brojevima (int) što mi možemo promijeniti u float ako želimo sabirati decimalne brojeve.

```
# Korisnik unosi koliko brojeva želi da sabere
n = int(input("Koliko brojeva želite da unesete? "))

# Postavljamo zbir na 0 obavezno inače greška u programu
zbir = 0

# Petlja za unos brojeva varijabla n je stop vrijednost
for i in range(n):
    while True:
        try:
            broj = int(input(f"Unesite {i+1}. broj: ")) # Unos broja
            break # Ako je unos validan, izlazimo iz petlje
        except ValueError:
            print("Molimo unesite ispravan broj.")
    zbir += broj # Dodajemo broj u zbir

# Ispis rezultata
print(f"Zbir unešenih brojeva je: {zbir}")
```

→ `while True` je petlja koja se ponavlja sve dok ne unesemo broj između 1 i 5. Može trajati vječno, tu vrstu petlji smo spomenuli na početku. While petlje ćete naučiti kasnije.

Varijabla `zbir` se na početku postavlja da ima vrijednost 0, iz razloga što je za računanje izraza `zbir += broj` (`zbir=zbir+broj`) potrebna neka početna vrijednost varijable `zbir`. Ako bi smo množili brojeve tada bi koristili `zbir=1` jer bi rezultat bio uvijek nula. Funkcija `try` nam je već poznata i koristi se za rad sa greškama koje se dešavaju prilikom unosa pogrešnih informacija u našem slučaju (decimalni brojevi i slova) tako da program radi iako će neko pokušati namjerno ili slučajno da unese pogrešne podatke što bi dovelo do pada programa.

*** Napisati program koji računa prosječnu ocjenu za unešeni broj predmeta.

Princip je isti gdje se koristi `zbir unešenih brojeva` i na kraju se samo podijeli sa brojem unešenih ocijena. Rezultat se ispisuje na 2 decimalna broja.

```
# Korisnik unosi broj ocena
n = int(input("Koliko ocjena želite da unesete? "))

zbir = 0

# Unos ocjena i računanje zbira
for i in range(n):
    ocjena = float(input(f"Unesite {i+1}. ocjenu: "))
    zbir += ocjena

# Računanje prosjeka na 2 decimale .2f
prosjeck = zbir / n
print(f"Prosječna ocjena je: {prosjeck:.2f}")
```

Poboljšajte ovaj program sa provjerom unosa pogrešnih informacija. Moguće unaprijeđenje je da korisnik samo unese koji je razred a program će odrediti broj ocjena koje se trebaju unijeti (koristiti `case` ili `if-elif`). Program se može proširiti i sa ispisom zaključne ocjene na osnovu dobijene prosječne ocjene funkcija `round` *****zakljucna=round(prosjeck)*****.

Do sada ste koristili petlje koje su bile vezane za neki niz brojeva. Sada prelazimo na for petlje koje će za vrijednosti brojača imati slovo, riječ ili brojeve koji nisu u nekom nizu.

*** Program koji ispisuje string riječ="Python" slovo po slovo jedno ispod drugog.

Objašnjenje: u ovoj situaciji petlja se okreće unutar stringa „riječ“ te će brojač u svakom krugu imati vrijednost jednog slova iz stringa, naravno kreće se od prvog slova.

```
riječ = "Python"

for slovo in riječ:
    print(slovo)  # Ispisuje trenutno slovo u stringu
```

Idemo naučiti nešto novo a to je funkcija `enumerate` koja će nam ispisati indeks svakog slova u stringu. Šta je indeks ? To je numerička vrijednost koja nam govori na kojoj se poziciji nalazi odgovarajuće slovo u stringu.

string	P	y	t	h	o	n
indeks	0	1	2	3	4	5

Okako python obrađuje svaki string gdje svaki element stringa pored svoje vrijednosti ima i vrijednost indeksa. To će nam biti korisno kada budemo htjeli zamjeniti neko slovo ili ubaciti neko slovo na određenu lokaciju u stringu, pronaći koja se slova ponavljaju u stringu itd... Pa idemo ispisati i redne pozicije svakog slova u stringu riječ.

```
riječ = "Python"


for indeks, slovo in enumerate(riječ):  # Start je 0 po defaultu
    print(f"{indeks}: {slovo}")
```

Kako ste primjetili da koristim sve češće formatirani ispis umjesto klasičnog ispisa koji bi izgledao ovako `print(indeks, ":", slovo)`. Formatirani način ispisa je lakše čitljiv i razumljiv

Ako nam bude nekada trebalo da ispišemo 1. slovo umjesto 0. slovo dodati ćemo parametar `start=1`.

```
riječ = "Python"

for indeks, slovo in enumerate(riječ, start=1):  # Start je sada 1
    print(f"{indeks}: {slovo}")
```



***Napisati program koji broji koliko slova ima u jednoj rečenici ili tekstu, razmaci između slova se ne računaju.

Objašnjenje: funkcija koja daje brojčanu vrijednost dužine stringa je `LEN`. N.P.R. `print len(a)` ispisuje broj koji pokazuje koliko slova i razmaka se nalazi u stringu `a`. Ali kako se u zadatku traži da ispišemo samo broj slova bit će nam potrebna petlja koja će ispitati svaki znak u stringu. Te ako je dati znak prazno polje onda ga neće brojati. Kako se u zadatku nešto broji trebat će nam brojač slova, možemo ga nazvati „brojač“ čija će vrijednost na samom početku biti 0. Naravno vrijednost znaka ispitivat ćemo uz pomoć if pitalice. If znak!=" " (ako znak nije prazno polje) povećavamo brojač za jedan (brojač+=1)


```
tekst = "Python je lagan"
brojac = 0

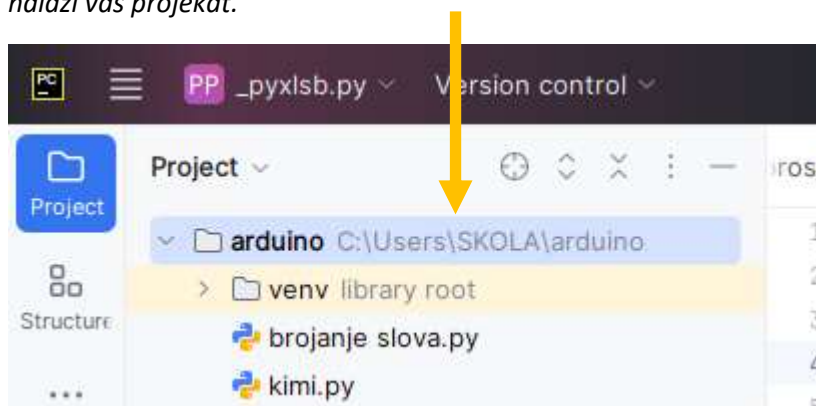
for slovo in tekst:
    if slovo != " ": # Preskačemo razmake
        brojac += 1

print(f"Broj slova (bez razmaka): {brojac}")
```

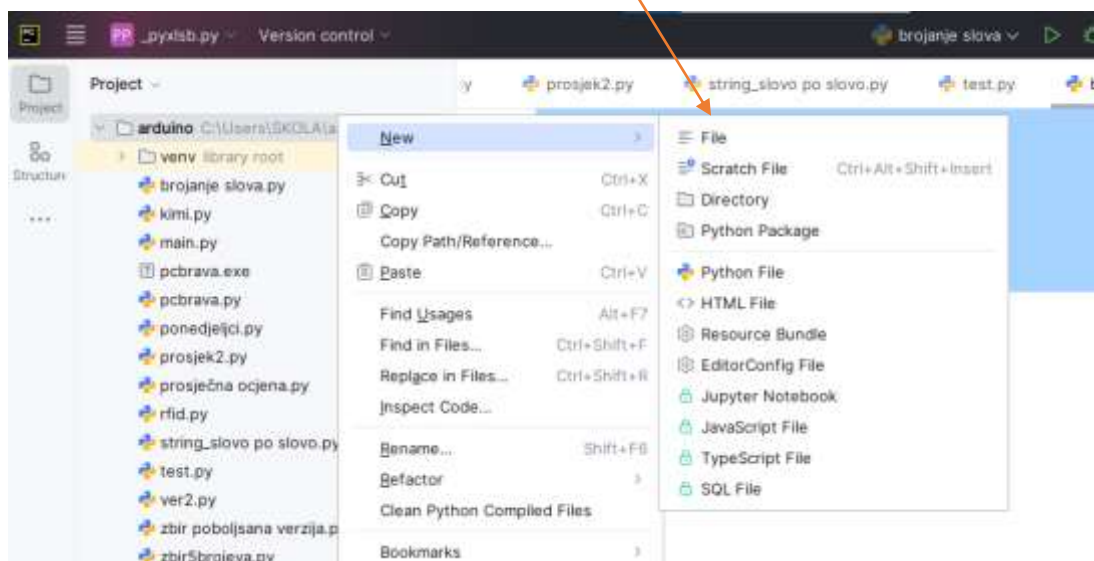
Pokušajte napraviti program koji ispisuje koliko slova „a“ se nalazi u rečenici.

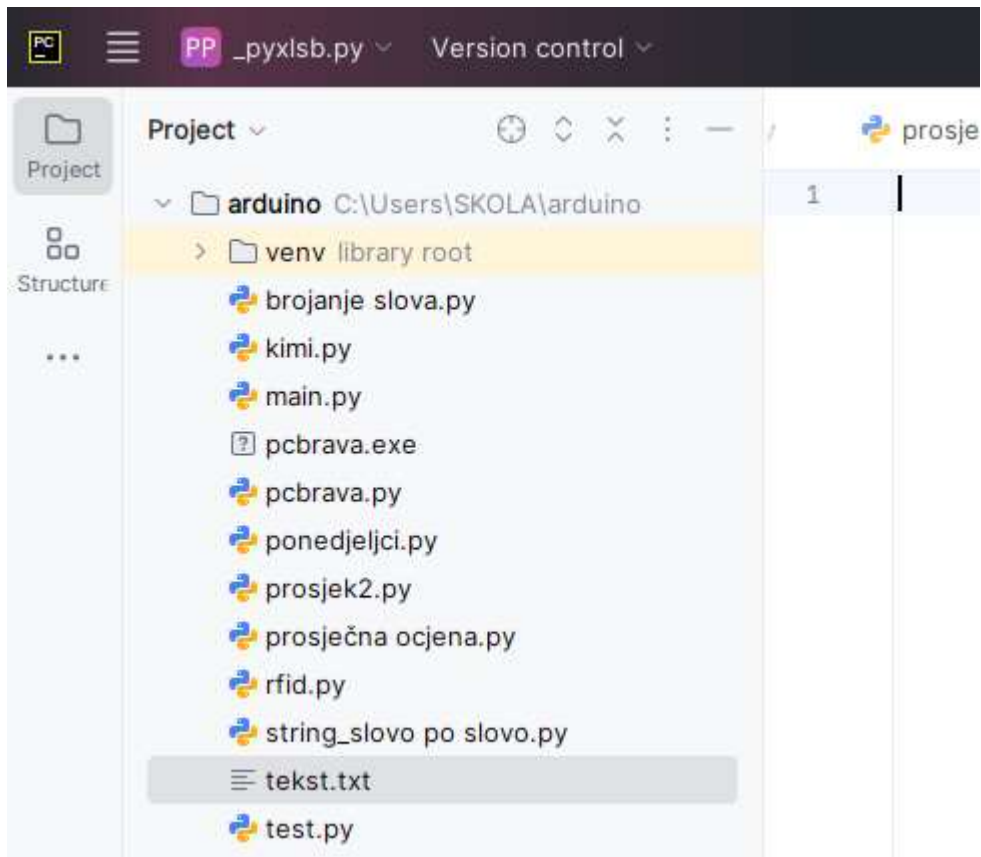
***** Napisati program koji će pretražiti datoteku koji se zove „tekst.txt“ za tačno određenom riječi koju će unijeti korisnik.**

Objašnjenje: Za ovaj zadatak morate pronaći folder gdje se nalazi vaš program. Ako koristite Pycharm lako možete pronaći lokaciju vašeg programa koji se nalazi u lokaciji gdje ste odredili da se nalazi vaš projekat.



*U ovom slučaju „ja“ bih morao kreirati tekstualni fajl u folderu C:\Users\SKOLA\arduino. Tekstualni fajl kreiram koristeći notepad aplikaciju u koju ću kopirati ili napisati neki tekst i snimiti u datom folderu. Oni koji koriste pycharm lako mogu kreirati tekstualni fajl ako desnim dugmetom miša kliknu na ime svog projekta i odaberu opcije New → File. I u polju napisati **tekst.txt**. Kasnije možete taj fajl pronaći u listi gdje se nalaze programi i unijeti neki tekst.*





Kada ste napravili tekstualni fajl i unijeli neki tekst u njega (kopirati neki tekst sa interneta). U ovom slučaju je procedura malo drugačija. Prvo je potrebno učitati datoteku u memoriju a zatim je pretražiti i ispisati odgovarajuće poruke.

```
pojam = input("Šta tražite? ")

with open('tekst.txt') as f:
    if pojam in f.read():
        print("Pronađeno!")
    else:
        print("Nije pronađeno.")
```

f.read() - učitava cijeli sadržaj datoteke kao jedan string

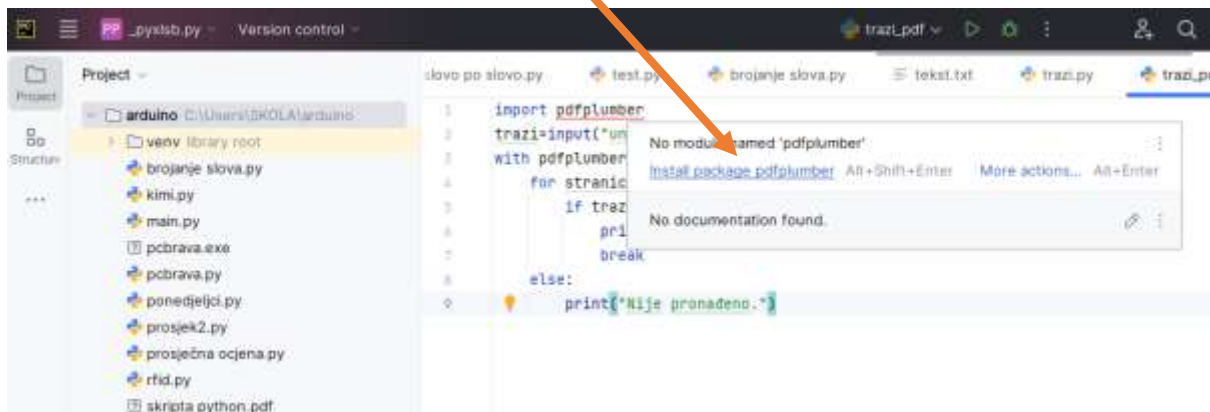
pojam in - provjerava postoji li traženi pojam bilo gdje u tom stringu

if/else - ispisuje odgovarajuću poruku

Ovakav način pretrage je dobar za male fajlove. Za velike fajlove moramo koristiti drugačije algoritme pretrage.

*** Napisati program koji će pretražiti pdf dokument. *** [Za one koji žele više.](#)

Objašnjenje: za rad sa dokumentima kao što su pdf ili word potrebne su nam dodatne biblioteke koje nam daju dodatne komande za rad sa specifičnim dokumentima. Za ovaj zadatak koristimo biblioteku koja se zove „pdfplumber“. Kako to nije standardna biblioteka potrebna je instalacija te datoteke kako bi smo pokrenuli program bez greške. Dodatne biblioteke se instaliraju tako što u terminalu pokrenemo komandu „pip install pdfplumber“. Ili direktno iz programa Pycharm u kojem je dovoljno dovesti pokazivač miša na ime biblioteke i poslije 2 sekunde dobijamo opciju za automatsku instalaciju date biblioteke.



Naravno potrebno je u folder direktno kopirati fajl koji želimo pretražiti. U ovom programu pretražujemo fajl koji se zove „skripta python.pdf“. Ako ne želimo kopirati fajl onda moramo opisati njegovu destinaciju u obliku C:\imefoldera\lokacija što komplikuje izvođenje programa.

```
import pdfplumber
```

```
# 1. Unos traženog teksta
```

```
trazeni_tekst = input("Unesite tekst koji želite pronaći u PDF-u: ")
```

```
# 2. Otvaranje PDF dokumenta umjesto „skripta python.pdf“ vi unosite ime vašeg pdf dokumenta.
```

```
with pdfplumber.open('skripta python.pdf') as pdf:  
    pronadjeno = False
```

```
# 3. Prolazak kroz sve stranice
```

```
for stranica in pdf.pages:
```

```
    # 4. Ekstrakcija teksta sa stranice
```

```
    tekst_stranice = stranica.extract_text()
```

```
# 5. Provjera postojanja traženog teksta
```

```
    if trazeni_tekst in tekst_stranice:
```

```
        pronadjeno = True
```

```
        break
```

```
# 6. Ispis rezultata
```

```
    print("Tekst je pronađen u dokumentu." if pronadjeno else  
        "Tekst nije pronađen u dokumentu.")
```

Ovdje se vidi par novih detalja koje ćemo objasniti u narednom tekstu.

1. Unos traženog teksta:

- Korisnik unosi tekst koji želi pronaći u PDF-u
- Unesena vrijednost se sprema u varijablu `trazeni_tekst`

2. Otvaranje PDF dokumenta:

- `pdfplumber.open()` otvara PDF dokument
- `with` blok osigurava pravilno zatvaranje dokumenta nakon upotrebe
- vrijednost varijable `pronadjeno` je `False` jer još nije pronađen `trazeni_tekst`

3. Iteracija (for petlja) kroz stranice:

- `pdf.pages` sadrži listu svih stranica u dokumentu
- Petlja `for` prolazi kroz svaku stranicu jednu po jednu

4. Ekstrakcija teksta:

- `stranica.extract_text()` pretvara sadržaj stranice u običan tekst
- Ova metoda pokušava sačuvati strukturu i redoslijed teksta

5. Pretraga teksta:

- Operator `in` provjerava postoji li traženi tekst u ekstrahiranom sadržaju
- Ako se tekst pronađe, varijabla `pronadjeno` postaje `True` (tačno) i petlja se prekida

6. Ispis rezultata:

- Na kraju se ispisuje odgovarajuća poruka ovisno o tome je li tekst pronađen
- `If` `pronadjeno` --- ako je vrijednost varijable `pronadjeno` `True` ispisat će se poruka tekst je pronađen u dokumentu `else` (inače) nije pronađen. Jer smo prije pretrage varijablu `pronadjeno` postavili na `False` (`pronadjeno=False`).

`True` i `False` su **logičke (boolean) vrijednosti** u Pythonu koje predstavljaju stanje **istinitosti** nekog izraza ili uvjeta.

- **True** znači "**tačno**" ili "**istina**" – uvjet je ispunjen.
- **False** znači "**netačno**" ili "**laž**" – uvjet nije ispunjen.

`For` petlje možemo koristiti i sa podacima koji se nalaze u posebnim varijablama koje se nazivaju liste. Ukratko rečeno liste su varijable koje u sebi sadržavaju niz podataka. Kasnije ćemo detaljno objasniti liste. Za sada ih koristimo da bi smo zapisali neki niz podataka u jednu varijablu.

*** Napisati program koji će ispisati koje zadatke danas trebam uraditi.

Objašnjenje: U program moram unijeti jedan niz zadataka. Niz srodnih vrijednosti mogu ubaciti u listu. I kasnije uz pomoć for petlje ispisati sadržaj te liste sa zadacima koji se ispisuju jedan ispod drugog.

```
# lista koja se zove zadaci u koju upisujemo zadatke
zadaci = ["Ići u školu", "Uraditi zadaću", "Programirati"]

print("Današnji zadaci:")
#for petlja koja će po redu ispisati svaki element liste
for zadatak in zadaci:
    print(f"- {zadatak}")
```

Idemo ispisati zadatke obrnutim redoslijedom

Za to nam je potreban parametar reversed

```
zadaci = ["Ići u školu", "Uraditi zadaću", "Programirati"]

print("Zadaci unatrag:")
for zadatak in reversed(zadaci):
    print(f"- {zadatak}")
```

Hajdemo još zakomplicirati, tako što ćemo pored imena zadatka napisati redni broj. Za to koristimo funkciju koja broji članove liste i vraća njihovu brojčanu vrijednost. enumerate() ali kako u listama prvi element ima indeks 0 (nula) mi ćemo dodati još jedan startni parametar 1. I još nam treba jedan brojač koji će se okretati u for petlji. Po tom pitanju Python je veoma fleksibilan tako da ga jednostavno možemo ubaciti u petlju koja će učitivati članove liste. Dobiti će ime **broj**.

```
zadaci = ["Uzeti lijek", "Čistiti stan", "Platiti račune"]

print("🚀 DANASNJI ZADACI:")
for broj, zadatak in enumerate(zadaci, 1):
    print(f" {broj} : {zadatak}")
```

Obratite pažnju na for petlju gdje se u jednoj for petlji nalaze 2 brojača broj i zadatak. Prvi mijenja vrijednosti od 1 do 3 a drugi kupi zadatke iz liste po redoslijedu.

Prolaz	broj	zadatak	Ispis
1.	1	"Uzeti lijek"	" 1 → Uzeti lijek"
2.	2	"Čistiti stan"	" 2 → Čistiti stan"
3.	3	"Platiti račune"	" 3 → Platiti račune"

Znak pribadače dobijamo tako što u print komandu dodamo unikod za taj znak.

```
print("\U0001F4CC DANASNJI ZADACI:") # 🚀 DANASNJI ZADACI:
```

*** Ispisati sve kodove za specijalne znakove u datom rasponu. Unikodovi se pišu u heksadecimalnom obliku koristeći escape parametar „\“.

```
# Ispis znakova iz određenog raspona (npr. U+1F300 do U+1F5FF)
for code in range(0x1F300, 0x1F5FF):
    print(f"{chr(code)} U000{code:04X}", end=" ")
    if code % 10 == 0: print() # Novi red svakih 10 znakova
```

Emotikone možemo direktno kopirati sa neke veb stranice i ubaciti ih u svoje programe sa copy/paste. <https://getemoji.com/>

*** Napisati program koji će iz liste pitanja čitati pitanja a odgovor će provjeravati iz liste koja se zove odgovori.

Kod ovakvog pristupa gdje istovremeno očitavamo podatke iz više listi koristimo funkciju zip() koja priprema podatke iz više listi za paralelnu obradu podataka. Gdje u svakom prolazu istovremeno čitamo podatke iz liste pitanja i liste odgovori. Kao i u predhodnoj verziji kviza trebamo znati broj pitanja koji dobijamo korištenjem funkcije len(pitanja) i broj tačnih odgovora koje postavljamo na nulu na samom početku programa.

```
# Podaci za kviz
pitanja = [
    "Glavni grad Francuske?",
    "Koliko je 2 + 2?",
    "Kako se zove najviši vrh BiH?"
]
odgovori = ["Pariz", "4", "Maglić"]

broj_pitanja = len(pitanja)
bodovi = 0

print("Dobrodošli u kviz! Odgovarajte tačno na pitanja.\n")

# Glavna petlja kviza
for i, (pitanje, tacan_odgovor) in enumerate(zip(pitanja,
odgovori), 1):
    print(f"{i}. {pitanje}")
    odgovor = input("Vaš odgovor: ").strip().capitalize()

    if odgovor == tacan_odgovor:
        print("Tačno!\n")
        bodovi += 1
    else:
        print(f"Netačno! Tačan odgovor je: {tacan_odgovor}\n")

# Rezultati

print(f"Procenat tačnih odgovora: {(bodovi / broj_pitanja) *
100:.0f}%")
```

Dodatna objašnjenja:

Podaci za kviz

```
pitanja = [  
    "Glavni grad Francuske?",  
    "Koliko je 2 + 2?",  
    "Kako se zove najviši vrh BiH?"  
]  
odgovori = ["Pariz", "4", "Maglić"]
```

Ovdje vidimo 2 potpuno ispravna načina zapisivanja elemenata listi (bitno je da iza svakog elementa stoji zarez). Lista pitanja gdje unosimo pitanja i lista odgovori.

```
broj_pitanja = len(pitanja)  
bodovi = 0
```

`len(pitanja)` je funkcija koja broji koliko ima elemenata u listi pitanja i daje konkretan broj. U našem slučaju varijabla `broj_pitanja` dobija vrijednost 3. Brojač bodova se postavlja na nulu. `bodovi=0`

```
for i, (pitanje, tacan_odgovor) in enumerate(zip(pitanja, odgovori), 1):
```

u ovoj petlji se dešava mnogo toga.

1. `zip(pitanja, odgovori)`

- **Šta radi:** Spaja dvije liste element po element iz svake liste u odgovarajuće parove

```
pitanja = ["Pitanje 1", "Pitanje 2"]
```

```
odgovori = ["Odgovor 1", "Odgovor 2"]
```

```
list(zip(pitanja, odgovori)) # [("Pitanje 1", "Odgovor 1"), ("Pitanje 2", "Odgovor 2")]
```

2. `enumerate(..., 1)`

- **Šta radi:** Dodaje brojač (indeks) ponavljanja, počevši od 1
- **Parametar 1:** Označava da brojanje počinje od 1 (umjesto podrazumijevanog 0)

```
list(enumerate(["a", "b"], 1)) # [(1, "a"), (2, "b")]
```

3. `for i, (pitanje, tacan_odgovor) in ...`

- **Razdvajanje parova:** Raspakuje parove u pojedinačne varijable
- `i` - redni broj pitanja (počinje od 1)
- `pitanje` - tekst pitanja
- `tacan_odgovor` - tačan odgovor za to pitanje

Kompletan tok podataka:

1. `zip` kombinuje pitanja i odgovori u parove

2. enumerate dodaje redne brojeve tim parovima
3. Petlja raspakuje svaki element u tri varijable

```
odgovor = input("Vaš odgovor: ").strip().capitalize()
```

U ovom dijelu vršimo više operacija

1. input unos varijable odgovor preko tastature
2. .strip uklanja nepotrebne razmake
3. capitalize pretvara svaku riječ u riječ koja počinje velikim slovom tako da „maglič“ postaje „Maglič“. Tako da iako pišemo sve malim slovima u obzir se uzima tačan odgovor jer inače maglič nije isto kao Maglič.

```
if odgovor == tacan_odgovor:
    print("Tačno!\n")
    bodovi += 1
else:
    print(f"Netačno! Tačan odgovor je: {tacan_odgovor}\n")
```

U ovom dijelu iz pomoć if pitalice provjeravamo da li je unešeni odgovor identičan sa podatkom iz liste odgovori. Ako je odgovor identičan onda ispisujemo poruku Tačno i uvećavamo broj bodova za

1. ELSE (inače) ispiši poruku Netačno i ispiši tačan odgovor.

```
print(f"Procenat tačnih odgovora: {(bodovi / broj_pitanja) * 100:.0f}%")
```

U ovom dijelu programa se ispisuje poruka procenat tačnih odgovora a u maloj zagradi se prvo računa procenat po principu broj tačnih odgovora podijeljen sa ukupnim brojem pitanja i pomnožen sa 100 gdje .0f računa na 0 decimalnih mjesta to jest procenat je uvijek cijeli broj.

Moguća poboljšanja programa.

1. ispisati odgovarajuću ocjenu na osnovu procenta tačnih odgovora (if pitalice ili case)
2. Na početku kviza tražiti od korisnika da unese svoje ime te na kraju zapisati ime i procenat tačnih odgovora u tekstualni fajl ili excel tabelu. Evo primjera zapisivanja varijable ime i procenta u tekstualni fajl. Vi pokušajte sami da ubacite ovaj kod i prilagodite vaš program. Po završetku kviza u folderu će se pojaviti tekstualni fajl rezultati.txt.

```
with open("rezultati.txt", "a", encoding="utf-8") as f:
    f.write(f"{ime}: ({int((bodovi/broj_pitanja)*100)}%) \n")
```

Dodatne informacije:

1. with open("rezultati.txt" otvara datoteku rezultati.txt
2. "a" ----- a znači append (dodaj) mod korištenja ovog fajla gdje na kraj fajla dodaje neki tekst. Postoje i drugi parametri „r“ read (čitaj) koristimo samo za očitavanje podataka iz tekstualnog fajla. „w“ write ... briše predhodni sadržaj iz fajla i upisuje novi
3. encoding="utf-8" ---- podrška za naša slova

4. f.write ---- funkcija koja zapisuje podatke iz zagrade u otvoreni fajl

5. int ---- pretvara rezultat proračuna u cijeli broj

6. \n ----- parametar koji pravi novi prazan red za upis narednih podataka.

Kasnije možemo podatke upisivati u excel tabelu, sortirati i napraviti module za unos novih pitanja i mnogo toga. Za takve zadatke moramo ovladati funkcijama, listama i još nekim bibliotekama za rad sa tabelama.

LISTE

Lista je vrsta varijable koja može da sadrži više podataka.

Osnovne karakteristike lista:

- **Redoslijed:** Elementi su poredani određenim redom
- **Promjenjivost:** Možemo mijenjati, dodavati ili uklanjati elemente
- **Raznovrsnost:** Mogu sadržavati različite tipove podataka
- **Indeksiranje:** Elementima se pristupa preko indeksa (počinje od 0)

Kako računar vidi listu?

Vidi je u obliku elementa liste i indeksa liste. Index je redni broj elemnta (podatka) u listi. Prvi element liste ima indeks 0.

lista [„prvi“, „drugi“, „treći“]

Ova lista sadrži tekstualne podatke jer se podaci nalaze pod znakom navoda. Brojeve u listama pišemo bez znakova navoda.

lista			
podatak	prvi	drugi	treći
indeks	0	1	2

Uvijek imati u vidu činjenicu da prvi podatak ima indeks 0 a NE 1 što bi bilo logično. Tako da u ovisnosti od zadatka koristimo samo podatke a nekada i indekse. Indeksi su nam potrebni da iz liste dobijemo specifičan podatak koji se nalazi na nekom mjestu u listi.

1. Ispis elemenata liste

Najprostiji način ispisa elemenata liste jeste uz pomoć komande `print(lista)`. Ovaj način ispisuje listu sa zagradama, navodnicima i zarezima. To je standardni način ispisa listi.

```
lista=["prvi","drugi","treći"]
print (lista)
```

Rezultat izvođenja ovog programa je:

```
['prvi', 'drugi', 'treći']
```

Nekima se neće svidjeti zagrade ili možda navodnici pa možemo koristiti poseban parametar za ispis „*“.

```
lista=["prvi","drugi","treći"]
print (*lista)
```

Rezultat izvođenja ovog koda je:

```
prvi drugi treći
```

Ako želimo ispisati elemente liste jedan ispod drugog onda moramo koristiti petlje.

```
lista=["prvi","drugi","treći"]
for element in lista:
    print(element)
```

Rezultat izvođenja ovog koda je: pogledajte ovaj film

<https://www.youtube.com/watch?v=0cyBRRYcSJY>

```
prvi
```

```
drugi
```

```
treći
```

Ako želimo ispisati samo neki element liste moramo koristiti indeks tog elementa. Naredni program ispisuje drugi element liste.

```
lista = ["prvi", "drugi", "treći"]
print(lista[1]) # Ispisuje drugi element
```

Rezultat izvođenja ovog koda je:

```
drugi
```

kako indeks elemenata liste počinje sa 0 drugi element ima indeks 1. Ako kažemo 2. podatak u listi indeks je uvijek jedan broj manje (1). `indeks=podatak_broj - 1`.

Nekada je potrebno ispisati redni broj elementa liste. Za to koristimo funkciju `enumerate` koju smo već spominjali u predhodnim zadacima.

```
lista = ["prvi", "drugi", "treći"]
for i, element in enumerate(lista, start=1):
    #print(f"{i}. element: {element}")
    print(i, "element:", element)
```

Ovdje vidite primjer formatiranog i standardnog python ispisa. Obije print funkcije daju isti rezultat. Dovoljno je ukloniti # ispred funkcije print.

*** Ispis zadnjeg elementa liste

```
lista = ["prvi", "drugi", "treći"]  
print(lista[-1])  # Ispisuje zadnji element
```

Kako su liste mutabilne (promjenjive) tako postoji niz načina kako vršiti razne operacije sa podacima

OPERACIJE SA PODACIMA KOJI SE NALAZE U LISTAMA:

1. Zamjena podataka u listi
 2. Dodavanje podataka u listu
 3. Brisanje podataka iz liste
 4. Sortiranje po raznim kriterijima (manji-veći, veći – manji)
 5. Kopiranje listi (*važno za čuvanje originalnih podataka radi kasnijeg upoređivanja istih*)
- Sve ove navedene operacije se mogu kombinovati.

u listama.

1. ZAMJENA PODATAKA U LISTI :

```
lista = [1, 2, 3]  
# zamjena podataka u listi se vrši uz pomoć  
# indeksa podatka a ne vrijednosti broj 2 posatje 20  
lista[1]=20  
print (lista)  
.....  
[1, 20, 3]
```

2. DODAVANJE PODATAKA U LISTU

```
lista = [1, 2]  
lista2=[4,5]  
# Dodaj broj 3 na kraj liste  
lista.append(3)          # [1, 2, 3]  
print(lista)  
# Dodaj na poziciju 1 vrijednost 10  
# pozicija 1 odgovara broju 2 tako da umjesto 2 ubacujemo 10  
lista.insert(1, 10)      # [1, 10, 2, 3]  
print(lista)  
# Spajanje lista (lista i lista2)  
lista.extend(lista2)     # [1, 10, 2, 3, 4, 5]  
print (lista)
```

append dodaje podataka na kraj liste

insert ubacuje poadtak na određenu poziciju. Pozicija je određena indeksom

extend spaja jednu listu sa listom koja se nalazi u zagradi. lista+lista2

3. BRISANJE PODATAKA IZ LISTE

```
lista = [10, 20, 30, 40, 50]

# Ukloni po vrijednosti
lista.remove(20)      # [10, 30, 40, 50] (uklanja vrijednost u
                        # zagradi)
print("izbrisan broj 20 ")
print(lista)
# Ukloni po indeksu
lista.pop(2)           # [10, 30, 50] (uklanja 3. element na
                        # poziciji 2 a to je broj 40)
print("izbrisan 3. član liste koji ima indeks 2")
print (lista)
# Obriši cijelu listu
lista.clear()          # [] ovako se označava prazna lista
print("prazna lista")
print (lista)
```

remove(20) -- briše podatak iz liste koji ima vrijednost 20. Brisanje po vrijednosti ili imenu člana liste

pop(2) -- briše 3. podatak iz liste uvijek imajte u vidu da indeks počinje sa 0. tako da 1. podatak ima indeks 0.

clear – briše sve podatke iz liste

Razlika između **remove** i **pop** metode jeste u tome što **remove** koristi vrijednost podatka koji se može nalaziti bilo gdje u listi. **Pop** metoda briše podatak bez obzira na njegovu vrijednost nego u obzir uzima tačno određenu lokaciju u listi (indeks).

Pogledajte ovaj film:

<https://www.youtube.com/watch?v=agu3sLn4j5I&list=PLRG4EiombR6ndB7FW9ujmCGcrD9o-Ptlw&index=4>

Programi za zabavu

[HTTPS://WWW.YOUTUBE.COM/PLAYLIST?LIST=PLGG3WGCSILI5DUVBQBQ8XVA1DN2B_Q487L](https://www.youtube.com/playlist?list=PLGG3WGCSILI5DUVBQBQ8XVA1DN2B_Q487L)

4. SORTIRANJE I OBRATANJE

Sortiranje veće količine podataka se obavlja u listama i veoma je jednostavno. Nije potrebno da poznajemo neke od algoritama za sortiranje podataka.

```
lista_brojeva = [3, 1, 4, 2]
lista_imena=["Ana", "Denis", "Adnan" ]
# Sortiranje brojeva
lista_brojeva.sort()          # [1, 2, 3, 4]
print("sortiranje brojeva",lista_brojeva)
#sortiranje imena
lista_imena.sort()
print("sortiranje imena", lista_imena)
print()                        #prazan red

#sortiranje unatrag
lista_brojeva.sort(reverse=True) # [4, 3, 2, 1]
print("sort reverse",lista_brojeva)

lista_imena.sort(reverse=True)
print("sort reverse",lista_imena)
print()

# Obrtanje redoslijeda
lista_brojeva.reverse()        # [1, 2, 3, 4] → [4, 3, 2, 1]
print(lista_brojeva)

lista_imena.reverse()
print (lista_imena)
```

Pogledati film (uključite prevod po potrebi): https://www.youtube.com/watch?v=3Tcq_dBhIrQ

4. DUŽINA LISTE I KOPIRANJE

Kada budemo koristili indekse elemeneata liste veoma važan detalj jeste odrediti dužinu liste.

