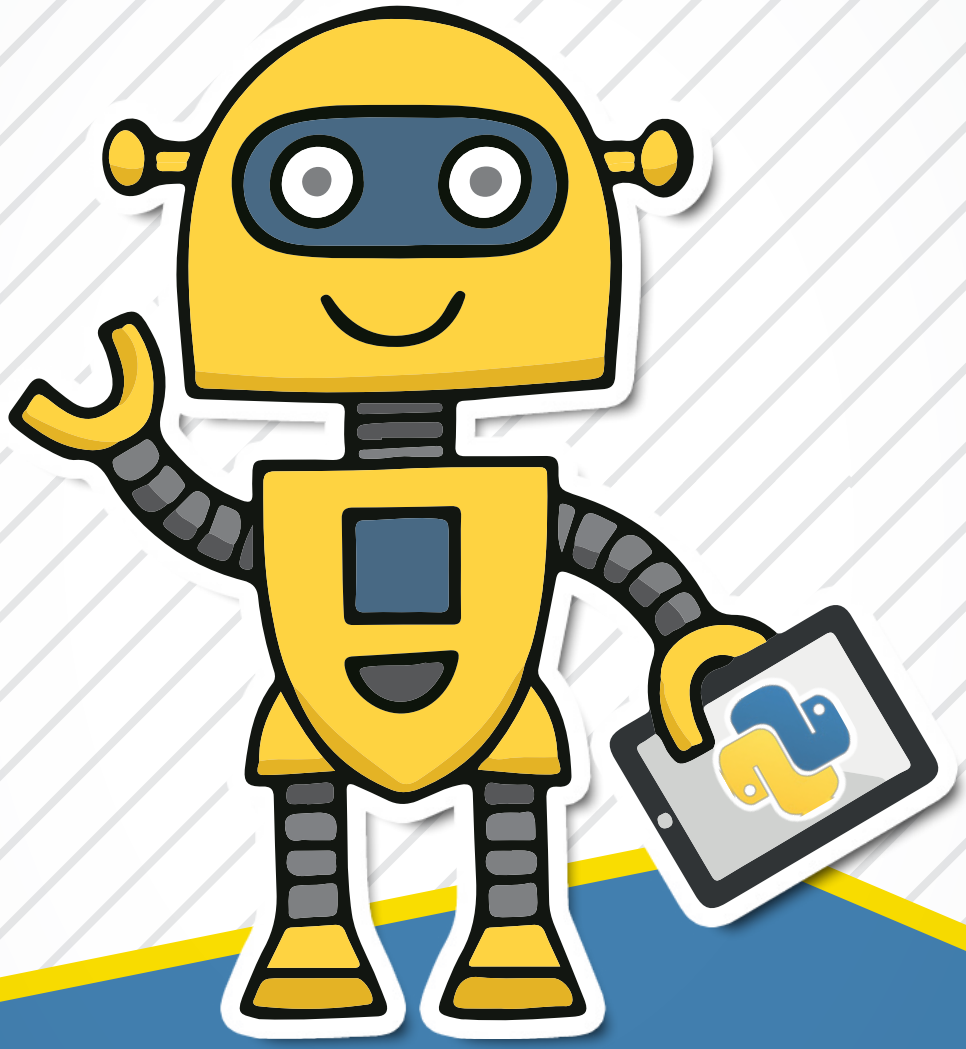
Tinker



Pajton za sve

sa zbirkom riješenih zadataka

Pajton za sve

sa zbirkom riješenih zadataka

Podgorica, 2020.

PAJTON ZA SVE
sa zbirkom riješenih zadataka

Izdavač:
NVO Tinker
Vasa Raičkovića 48, Podgorica

Autorka:
Jelena Milojković

Urednica:
Jelena Milojković

Saradnici:
Boris Milojković
Ivan Šaranović
Katarina Obradović
Dejana Ponoš

Štampa:
Studio Heber

Tiraž:
100

igramiranje.me
stemtinker.me

CIP - Каталогизација у публикацији
Национална библиотека Црне Горе, Цетиње

ISBN 978-9940-799-00-7

COBISS.CG-ID 15618820

Pod pokroviteljstvom

CRNA GORA
MINISTARSTVO NAUKE

Partneri

Arhimed

udruženje Roditelji

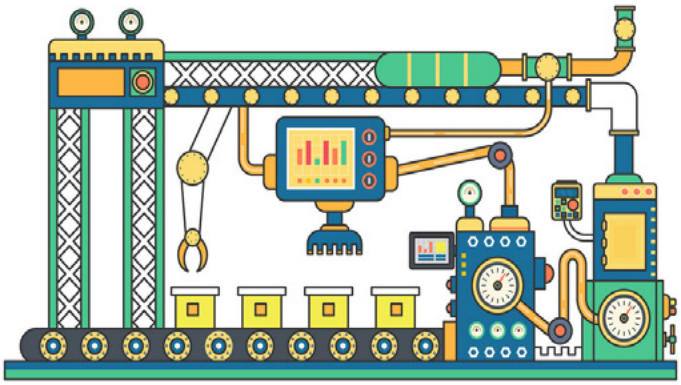
ZZi

SADRŽAJ

Kako je razvijan program rada sa djecom, tako je i ovaj priručnik obogaćivan novim primjerima i drugim pratećim materijalom.

Ovo izdanje nastalo je tokom ljeta 2020. godine, u okviru **Tinker online ljetnje škole programiranja "Pajton za sve!"** koja je, pod pokroviteljstvom **Ministarstva nauke**, realizovana za **polumaturante i srednjoškolce iz cijele Crne Gore.**

Kao i što sam naziv kaže, ova knjiga je **namjenjena svima, bez obzira na njihov uzrast ili prethodni nivo znanja.** Napisana je tako da omogućava upoznavanje sa programskim jezikom Pajton i uvježbavanje korišćenja stečenog znanja, korak po korak, tako da čitalac na kraju, upotrebljavajući sve elemente koje je postepeno uvježbavao, može samostalno napraviti dobro poznatu igru Vješala.



Imajmo na umu da rješavajući programerske izazove vježbamo i razvijamo našu sposobnost **logičkog promišljanja**. Pored toga, tokom realizacije svih Tinker aktivnosti, sa velikom ljubavlju i neprestano podstičemo, njegujemo i razvijamo **drugarstvo**. Gradimo sjajan tim djece, mladih, odraslih, koji su uvijek spremni da postave pitanje, daju konstruktivan prijedlog i pomognu jedni drugima. Tinker aktivnosti poput brojnih radionica programiranja, „Matematičkih mozgalica", matematičkih radionica za kadete i juniore,... i naravno, radionice Igramiranja, predstavljaju neke od naših načina da **oplemenimo zajednicu u kojoj živimo**.

Princip koji poštujemo u Tinkeru je da konkretan rad uvijek prilagođavamo uzrastu polaznica i polaznika, njihovom broju, prethodnom predznanju i sl. Takođe, dinamika rada može biti različita i u zavisnosti od toga da li radionice radimo online ili onsite.

Plan rada ljetnje škole napravljen je po radionicama, i materijal je grupisan po nedjeljama sa N1, N2, N3,... Uobičajeno vrijeme trajanja jedne radionice je minimalno 1h, najduže 1h i 30m.

Prvi serijal radionica u okviru Tinker ljetnje škole programiranja podrazumijeva 32h rada, i to :
- 8h intenzivnih online radionica i
- 32h praktičnog rada koji se sastoji od:
  - izrade zadataka
  - izrada finalnog projekta primjenom stečenog znanja: igre Vješala.

Materijal obrađen na intenzivnim radionicama, inače se obrađuje na troipomjesečnoj Tinker obuci.

Zadatke koji su obuhvaćeni ovim priručnikom kao i dodatne zadatke prikupljene tokom godina, pripremili smo i u Tinker repl.it virtuelnoj učionici u kojoj se, pored zadataka, nalaze i prateća teorijska uputstva. Kao što je ranije pomenuto, ova knjiga predstavlja priručnik za realizaciju **Prvog serijala intenzivnih online radionica programiranja „Pajton za sve!"** koji Tinker mentorima, vršnjačkim edukatorima i, naravno, polaznicama i polaznicima Tinker radionica, može biti od pomoći tokom realizacije samih radionica.

Osnovno izdanje ovog priručnika nastalo je 2018. godine, u okviru definisanja prve verzije programa za realizaciju Tinker radionica programiranja za djecu i mlade. Kasnija dopuna priručnika realizovana je 2019. godine, tokom realizacije prvog serijala Tinker radionica programiranja u programskom jeziku Python, koje su se, u saradnji sa Glavnim gradom, održavale u Centru kompetencija u Podgorici. Već tada smo pripremili online učionicu na repl.it platformi u kojoj smo kreirali naloge za polaznike radionica, zadatke, teorijska objašnjenja, testove za automatsku provjeru ispravnosti postavljenih zadataka, prijedloga rješenja i slično, kao i praćenje rada svih polaznika. U početku smo radionice realizovali na način da smo skoro uvijek svi bili fizički prisutni u učionici, a djeca su dobijala domaće zadatke koje su u našoj virtuelnoj učionici radila "od kuće", dok su mentori mogli pratiti izradu zadataka. Dopalo nam se to što je, i pored fizičke udaljenosti, svaki mentor imao jasan pregled informacija o tome koliko procenata trenutno objavljenih zadataka je svaki polaznik tačno uradio, koji zadaci su urađeni a koji ne. Pored toga što se testiranje rješenja moglo podesiti tako da se radi

automatski u okviru platforme repl.it, mentorima je omogućeno i da "ručno" pregledaju rješenja i pošalju poruku, komentar ili instrukcije direktno, svakom od polaznika, za svaki objavljeni zadatak pojedinačno.

Ovakav pristup nam je omogućio da početkom 2020. godine, u doba primjene intenzivnih mjera u okviru borbe sa pandemijom Covid-19, sa grupom djece koja su neposredno prije pandemijskih okolnosti otpočela sa serijalom radionica programiranja u Pajtonu, sastajući se u fizičkoj učionici, na jednostavan način nastavimo rad i to potpuno u online režimu. Zajedno smo radili na tome da vrijeme koje provodimo kod kuće obogatimo zajedničkim druženjem i učenjem, koje smo zaokružili 11. juna 2020. godine, kada je organizovana online provjera znanja za polaznike online radionica. Veoma smo ponosni na polaznike iz ove grupe koji za nas predstavljaju na neki način "ratne drugove". Zajedno smo programirali i družili se i iskoristili vrijeme kućne izolacije na najbolji mogući način.

Nakon realizacije prvog serijala online Tinker radionica, dobili smo veoma pozitivne povratne informacije od polaznika, odnosno djeca su odlično reagovala na novi način rada, a svi polaznici koji su uspješno savladali zadate ciljeve, pozvani su da nastave druženje na narednim radionicama. To nas je inspirisalo da planiramo Tinker ljetnju školu programiranja za djecu iz cijele Crne Gore, potpuno u online režimu.

Zahvalnica

Krug Tinker prijatelja je veliki i mnogi prijatelji, kolege, profesionalci, entuzijasti, naučnici,...,veliki i mali radoznalci, su svojim radom i idejama doprinijeli njenom jačanju. Trudimo se da o svima njima pišemo na našim internet stranicama www.igramiranje.me i www.stemtinker.me.

Posebno se zahvaljujem svima koji su svojim radom, idejama, motivacijom doprinijeli da ovaj priručnik bude bolji i korisniji čitaocima spremnim da zajedre u svijet programiranja:

Tinker mentoru Ivanu Šaranoviću, uz čiju pomoć smo, serijal radionica programiranja u Pajtonu započet u Centru kompetencija, po prvi put realizovali potpuno u online režimu i koji svojim temeljnim pristupom, posvećenošću i sjajnim humorom daje Tinker radionicama poseban karakter „stare škole" potpuno prilagođene savremenom načinu učenja. Pod njegovim budnim okom nije bilo teško kvalitetno raditi sa velikim brojem polaznika u onsite i online režimu. Posebno mu se zahvaljujem na pomoći u prikupljanju i obradi dijela materijala prezentovanog u ovom priručniku.

Tinker mentorki Katarini Obradović, koja je ojačala naš tim tokom online ljetnje škole programiranja realizovane ljeta 2020. godine, pod pokroviteljstvom Ministarstva nauke i koja je svojim nesebičnim radom pratila napredak velikog broja polaznica i polaznika, brižljivo i suptilno ih vodila svojim sjajnim idejama, komentarima, instrukcijama i nesebično prenosila svoje dragocjeno znanje i iskustvo.

Tinker mentoru Seidu Kršiću, sa kojim smo realizovali naše prve Tinker radionice programiranja u Pajtonu, na njegovoj posvećenosti matematici i programiranju i njegovom sjajnom pristupu u radu sa djecom.

Vršnjačkim edukatorima Milici M., Harisu Š., Tei B. i Jovanu Š., koji su svojim primjerom pokazali kako se prenošenjem znanja, znanje uvećava. Posebno hvala Milici M. koja se posvetila održavanju naših repl.it učionica: pisanju zadataka, pripremi i pisanju testova za automatsko testiranje rješenja, kao i unosu modela rješenja.

Naravno Tinkera ne bi ni bilo bez partnerske sinergije sa sjajnom Dejanom Ponoš, koja gaji ogromnu ljubav i razumijevanje prema svoj djeci i ima nepresušnu težnju da doprinosi zajednici u kojoj živi i time je učini prijatnijim i boljim mjestom za sve, u čemu veoma često uspijeva! Sve je počelo, krajem 2017. godine kada smo, razmišljajući kako da oplemenimo vrijeme naše, ali i druge djece, isplanirale, a u martu 2018.

godine, zajedno realizovale prvi serijal radionica Igramiranja: igre i programiranja, sa djecom uzrasta od 6 do 12 godina. Bilo je to nevjerovatno iskustvo, koje nas je uvelo u nove Tinker avanture. Hvala Dejani na istrajnosti, vremenu i neiscrpnim idejama!

I na kraju, hvala mojoj porodici na, za mene, najvažnijoj i najsnažnijoj podršci! Vaša ljubav i razumijevanje su neprocjenjiv i nepresušan izvor energije i inspiracije!

Posebnu zahvalnost dugujem svom suprugu i kolegi Borisu Milojkoviću, koji je, sa velikom ljubavlju i stručnošću, od samog početka, pomagao da Tinker pronađe najbolji kurs razvoja. On je, pored ostalog, sa velikom pažnjom i posvećenošću pripremio i realizovao Drugi serijal radionica u okviru Tinker ljetnje škole programiranja 2020, i to onaj najslađi dio, koji svi jako vole: programiranje grafičke igrice koristeći Pygame.

Bez svih vas Tinker Igramiranje ne bi bilo ono što sada jeste. Hvala vam!

Šta je naš cilj

Bilo bi nam drago da polaznike Tinker radionica podstaknemo da **prepoznaju i razvijaju ljubav prema nauci** i da ih, uklanjajući bilo kakve barijere i strahove, vodimo kroz dio ranije neistraženih ili nedovoljno istraženih „svijetova“. Pored toga što na Tinker radionicama polaznice i polaznici stiču nova znanja, oni poboljšavaju bistrinu logičkog rasuđivanja i pripremljenost kako za individualni, tako i za timski rad.

Organizovanje radionica programiranja, uz pomoć brojnih partnera, započeli smo prije nepune tri godine. Dosadašnje iskustvo nam je pokazalo da se veliki broj mladih, ali i iskusnih profesionalnaca, susrijeće sa nečim što bi smo mogli nazvati **neophodnošću današnjice**, a to je **potreba da u svojoj struci, bez obzira na to o kojoj struci je riječ, koriste programiranje.** Za koju god profesiju da se kasnije odluče, već sada, a pogotovu u veoma bliskoj budućnosti, vjerujemo da će im znanja stečena na Tinker radionicama, sistematičnost i razvijen pristup u rješavanju izazovnih problema i istraživački duh biti dobra osnova za dalji razvoj, istraživanje i stvaranje.

Ono što Tinker radionice čini posebnim nije ova knjiga, već energija i znanje koje međusobno dijelimo, druženje, međusobno poštovanje, želja da napredujemo i učinimo da napreduju i ljudi koji nas okružuju, odnosno cjelokupna zajednica u kojoj živimo. Stvaramo i gajimo snažnu Tinker ekipu mladih ljudi, koji su uvijek spremni da razmjenjuju ideje, našale se, postavljaju pitanja, daju odgovore, budu inventivni i stvaraju. I to je ono na šta smo ponosni!

Jelena Milojković, *direktorica NVO Tinker*

27.septembar 2020.
Herceg Novi

Mi znanjem mjenjamo svijet.
Mi učimo, mislimo i sanjamo.
Istražujemo neistraženo,
jer mnogo toga još ne znamo.

Mi volimo izazove
i njih se ne bojimo.
Snagom našeg uma,
zagonetke odgonetamo,

Mi postavljamo pitanja
i tragamo za odgovorima.
Veslamo uzvodno ka
vrijednom izvoru znanja.

Mijenjamo svijet, tako što sebe mijenjamo.

UVOD

Nije važno koliko imaš godina! Ukoliko želiš da se isprobaš u novim izazovima, stekneš nove vještine, programiraš i rješavaš logičke probleme, na pravom si mjestu!

RAČUNARI

Računari se mogu nalaziti svuda: u telefonu, automobilu, satu, konzoli za video igrice, robotu, u opremi koja se koristi u industriju, u spravama za teretane, medicinskim uređajima, čestitkama, usisivačima, šporetima, ...svemirskim letilicama, ... Računari se mogu nalaziti svuda i sve njih najčešće treba isprogramirati.

Prema posljednjim istraživanjima, skoro 70% programerskih poslova je van IT industrije, što znači da programiranje može biti korisno skoro bilo kojom profesijom da se bavimo.

PROGRAMSKI JEZICI. KAKO "RAZGOVARAMO" SA RAČUNARIMA?

Da bi kompjuter „razumio" šta želimo da mu kažemo, to mu trebamo napisati na jeziku koji i on „poznaje". Takav jezik nazivamo **programskim jezikom**.

KOLIKO IMA PROGRAMSKIH JEZIKA?

Prvo, da li znaš koliko ima različitih jezika koji služe za sporazumijevanje ljudi? Ima ih oko 7.000!

A programskih jezika? Postoji i jako puno programskih jezika. Preko 250! Neki od njih su : Java, Scala, Perl, T-SQL, Pascal, C, C++ i Fortran.

Mi ćemo učiti programski jezik Python (čitamo: Pajton).

ŠTA JE TO PROGRAMIRANJE?

Pisanje programa korišćenjem nekog od programskih jezika, nazivamo **programiranjem**.

U okviru svakog programskog jezika definisana su stroga pravila koja moramo poštovati prilikom programiranja. Program može pisati, odnosno isprogramirati, čovjek ali ne samo čovjek. Program može biti napravljen i od strane drugog programa koji se izvršava na računaru.

Kompjuterske programe ponekada nazivamo i aplikacijama.

Na primjer, programski jezik Pajton, koji ćemo ovdje upoznati, korišćen je za razvoj programa kao što su YouTube, Gmail, Google Maps, ali i mnogih drugih.

Programski jezik Pajton nastao je početkom devedesetih godina prošlog vijeka, a osmislio ga je Gvido van Rosum sa Univerziteta Stičing u Holandiji.

Iako Pajton nije više tako mlad jezik, izuzetno je popularan. Evo par razloga za njegovu popularnost:

- jednostavan je,
- lako se uči,
- u njemu se brzo programira.

Iz navedenih razloga, često se njime na svom poslu služe ljudi različitih profesija. Inače, veoma je uobičajeno i biće sve uobičajenije da profesionalni programeri nisu jedini koji programiraju kao što ni profesionalni pisci nisu jedni koji pišu.

Programi pisani u programskom jeziku Pajton se najčešće **interpretiraju**. Dakle, kako bi program koji smo napisali u Pajtonu mogao da se izvrši, potreban nam je program po imenu Pajton interpreter. Ovaj program prvo tumači (interpretira), a zatim i izvršava Pajton naredbe. Uz interpretator se obično isporučuje i veoma razvijena standardna biblioteka modula koja omogućava širok spektar primjene.

Pajton interpreteri mogu da preuzmu i izvrše cijele programe, ali mogu i da rade u interaktivnom režimu, tako da se svaka unijeta naredba odmah izvršava.

Pajton interpreter izvršava se u okruženju koje zovemo **školjka** (engl. shell) i postoje razna okruženja (školjke) u kojima Pajton interpreter može da se izvršava.

Kako bismo, za početak, preskočili instaliranja i podešavanja, počećemo sa programiranjem u Pajtonu u našoj **virtuelnoj učionici**, koristeći za programiranje online platformu www.repl.it.
Takođe, moguće je isprobati online školjku na internet stranici: www.python.org/shell.

Instaliranje i podešavanja ostavićemo za kasnije.

Napišimo prvi program

"Upoznaj se" sa računarom. Napišimo prvi program u kojem

- Računar "pita": "Kako se zoveš?".
- Korisnik odgovara na pitanje, upisujući svoje ime na tastaturi (kada ispiše svoje ime pritiska tipku "Enter").
- Računar "pozdravlja" korisnika programa tako što ispisuje njegovo ime, a u nastavku sljedeći tekst ", `bas lijepo ime. Zdravo!`")

U programskom jeziku Pajton, program bi mogao da izgleda ovako:

```
ime = input("Kako se zoves?")
print(ime,", bas lijepo ime. Zdravo!")
```

	PRIMJER
Ulaz	Milica
Izlaz	Milica, bas lijepo ime. Zdravo!

Kako se program izvršava?

Kada pokrenemo izvršavanje programa (klikom na dugme "run") vidjećemo da je startovan program jer će se ispisati tekst "Kako se zoveš?".
Nakon unošenja imena i pritiskanja tipke "Enter", program će ispisati unijeto ime iza kojeg će dodati ", `bas lijepo ime. Zdravo!`".

Pošto je ovo sve što smo u programu tražili od računara da uradi, program se završio.

Napiši svoju priču.
Po uzoru na prethodni program, napiši program za drugačiji "razgovor" sa računarom.

Luckaste rečenice
Napiši program koji na osnovu tri ulazne vrijednosti:

- pridjev,
- imenica i
- glagol

ispisuje rečenicu koja počinje upravo sa te tri unijete riječi (u redosljedu u kojem su unijete) a završava se sa tekstom "preko dugačke šarene duge."

U programskom jeziku Python, program bi mogao da izgleda ovako:

```
pridjev = input("Unesi pridjev: ")
imenica = input("Unesi imenicu: ")
glagol = input("Unesi glagol: ")
print("Tvoja luckasta recenica:")
print(pridjev, imenica, glagol, "preko dugačke šarene duge.")
```

Komentari

Ukoliko u programu želimo upisati tekst koji je nama koristan, ali računar ga, prilikom izvršavanja programa ne treba uzeti u obzir, takav tekst nazivamo komentarom. Da bi računaru naznačili da je reč o komentaru, možemo koristiti "tarabu" (#):

```
# Ovo je komentar
print("Naredba print će se izvršiti.")
# Ovo se neće izvršiti
```

Ukoliko želimo više redova staviti u komentar, to možemo uraditi na sljedeće načine:

```
def multiline_komentar():
    # Ovo je primjer
    # pisanja komentara
    # u više redova u Pajtonu
```

ili ovako:

```
"""
Ukoliko baš ne volite da pritiskate `enter`
i da kucate brojne tarabe, možete komentar
ispisati u više redova koristeći
trostruke navodnike.
"""
```

ili, pak, ovako:

```
'''
Ukoliko baš ne volite navodnike i
da pritiskate `enter`
i da kucate brojne tarabe, možete komentar
ispisati u više redova koristeci
trostruke apostrofe. :)
'''
```

PROMJENLJIVE

Vjerovatno su vam, na početku vaše programerske karijere, govorili da su promjenljive **imena koja smo dali međurezultatima u našim programima.** Sada već možemo malo preciznije pojasniti šta su to promjenljive.

Riječ **promjenljiva** u programiranju opisuje **naziv *mjesta u memoriji računara na kojem se čuvaju podaci*** kao što su brojevi, tekstovi, liste brojeva i tekstova i slično. Na tom mjestu se mogu čuvati različite vrijednosti, pa otuda potiče i sam naziv "promjenljiva".

Možemo reći da je promjenljiva kao oznaka, naljepnica sa nekim imenom, odnosno nazivom.

Da bismo napravili promjenljivu po imenu **rezultat**, koristimo znak jednakosti (=) kako bi Pajtonu "rekli" koju vrijednost podataka će promjenljiva označavati, odnosno čuvati. U programerskom žargonu bi rekli

da jednakost koristimo za "dodjeljivanje vrijednosti promjenljivoj".

Kada napišemo:

```
rezultat=100
```

time smo napravili promjenljivu *rezultat* koja označava broj 100.
Obratite pažnju da ovo **ne znači** da ne može postojati druga promjenljiva koja takođe označava istu vrijednost: 100.

Da bismo saznali koju vrijednost promjenljiva označava, unijećemo naredbu **print**, iza koje ćemo u zagradama ukucati **naziv promjenljive**. Kao rezultat izvršavanja naredbe print, ispisaće se vrijednost promijenljive rezultat, odnosno broj 100:

```
print(rezultat)
```

izlaz:
100

Ukoliko u Pajtonu želimo promijeniti vrijednost promjenljivoj koja se zove *rezultat*, npr. tako da promjenljiva više ne označava vrijednost 100, već ukazuje na vrijednost 200, učinićemo to na sljedeći način:

```
rezultat=200
print(rezultat)
```

izlaz:
200

U prvom redu naveli smo da promjenljiva rezultat označava broj 200. U drugom redu pitali smo šta promjenljiva rezultat označava, jer smo želeli da budemo sigurni da se vrijednost koju promjenljiva označava promijenila. Pajton ispisuje rezultat u posljednjoj liniji i mi smo sigurni da promjenljiva rezultat sada označava vrijednost 200.

Možemo koristiti više od jedne promjenljive za istu "stvar", odnosno za istu vrijednost.

```
rezultat=200
rekord= rezultat
print(rekord)
```

izlaz:
200

U ovom primjeru mi smo rekli Pajtonu da želimo da promjenljiva *rekord* označava istu "stvar" koju je označavala promjenljiva *rezultat*. To smo učinili tako što smo koristili znak jednakosti između promjenljivih *rekord* i *rezultat*:

```
rekord = rezultat
```

Promjenljivu *rezultat* smo mogli nazvati i drugačije, npr. *xyz*, ali takav naziv nam ne bi mnogo značio. Biramo imena promjenljivima koja imaju smisla i koja nam govore koju će vrstu podataka ta promjenljiva označavati. Pritom moramo poštovati sljedeće:

- Imena promjenljivih mogu se sastojati od **slova, brojeva i podvlake (_), ali ne mogu početi sa brojem**.
- Postoji **razlika između malih i velikih slova**.
- **Preporučuje** se da u imenima koristimo samo **slova engleske abecede.**
- Naziv promjenljive **može biti jedno slovo**, npr. **a**, ali može biti i duži niz karaktera.
- Ime **promjenljive ne može sadržati razmak**, ali možemo na primjer **koristiti podvlaku za razdvajanje riječi**.
- Ime promjenljive **ne može sadržati ni srednje crtice i ostale interpunkcijske i specijalne znake**.

Ponekad je dovoljno dobro birati kratka imena, ali ponekad je neophodno davati duža imena promjenljivima koja imaju više smisla.

Korišćenje promjenljivih

U Pajtonu možemo napisati program koji računa vrijednost izraza 20+10*365. Program izgleda veoma jednostavno:

```
20 + 10 * 365
```

izlaz:
3670

Pokušaj sada da koristiš promjenljive i da brojeve sačuvaš u promjenljivima. Na primjer:

```
pocetna_ustedjevina = 20
dnevna_ustedjevina = 10
broj_dana_u_godini = 365
```

Ove komande prave promjenljive koje se zovu *pocetna_ustedjevina, dnevna_ustedjevina, broj_dana_u_godini*. Sada, možemo ponovo unijeti izraz, ali na sljedeći način:

```
pocetna_ustedjevina + dnevna_ustedjevina *
broj_dana_u_godini
```

izlaz:
3670

Da li smo dobili isti odgovor? Da, jesmo!

Zašto bi onda koristili promjenljive kad se sve fino može izračunati i bez njih?
Veoma često ćemo imati potrebu da promijenimo vrijednost npr. samo jednoj promjenljivoj i za takve vrijednosti izračunati vrijednost izraza. Npr. dnevna ušteđevina se promijenila i sada umjesto 10, ima vrijednost 1. Izmijenićemo vrijednost promjenljivoj `dnevna_ustedjevina`:

```
dnevna_ustedjevina = 1
```

Sada možemo kopirati isti izraz od ranije i vidjećemo da se njegova vrijednost promijenila:

```
pocetna_ustedjevina + dnevna_ustedjevina
* broj_dana_u_godini
```

izlaz:
385

Ako promijenimo vrijednosti za naše tri promjenljive iz primjera, promijeniće se i vrijednost izraza:

```
pocetna_ustedjevina + dnevna_ustedjevina
* broj_dana_u_godini
```

Na primjer, program bi mogao da izgleda ovako:

```
pocetna_ustedjevina = 100
dnevna_ustedjevina = 100
broj_dana_u_godini =366
pocetna_ustedjevina + dnevna_ustedjevina
* broj_dana_u_godini
```

izlaz:
36700

Naravno, korišćenje promjenljivih u jednostavnim primjerima poput ovog možda i ne izgleda mnogo korisno i neophodno. Biće prilike da se uvjeriš na složenijim primjerima. Za sada upamti da promjenljive služe za označavanje stvari, odnosno vrijednosti, tako da ih lako možeš kasnije koristiti.

Da malo zakomplikujemo igru? Napiši program u kojem unosiš dva cijela broja. Program dalje treba da sračuna njihov zbir i da ga ispiše na ekranu.

ULAZ/IZLAZ ODNOSNO INPUT/OUTPUT

Ulaz (Input)

Programeri u svojim programima često imaju potrebu za interakcijom sa korisnicima, bilo sa ciljem da preuzmu neke podatke od korisnika ili da mu omoguće uvid u neke rezultate. Većina programa danas koristi dijalog prozorčiće (dialog boxes) kako bi se od korisnika zatražili i preuzeli potrebni ulazni podaci. U Pajtonu imamo na raspolaganju ugrađenu funkciju za preuzimanje ulaznih vrijednosti. Pogledaćemo primjere unošenja vrijednosti sa standardnog ulaza, odnosno sa tastature:

input ( ) : Ova funkcija uzima ulaznu vrijednost od korisnika

Na primjer:

```
v = input("Unesi vrijednost: ")
print(v)
```

Kako funkcija input() radi u Pajtonu:

- Kada se input() funkcija izvršava, tok programa se **zaustavlja** sve dok korisnik ne unese vrijednost.
- Tekst, odnosno poruka koja se na izlaznom ekranu može prikazati korisniku (poput poruke "Unesi vrijednost: ") je **opcionalna**.
- Koju god vrijednost korisnik da unese kao ulaznu vrijednost, input funkcija je konvertuje u **string**. Čak i kada korisnik unese neki cijeli broj, input() funkcija ga konvertuje u string. Ukoliko u programu želite da ulazna vrijednost bude tretirana kao cijeli broj, potrebno je da je u programu eksplicitno konvertujete u integer koristeći typecasting.

Na primjer:

```
b=int(input())
#promjenljiva b čuva unijetu cjelobrojnu
#vrijednost
```

Izlaz (Output)

Najjednostavniji način da generišemo izlaz, odnosno output iz programa je korišćenjem print() funkcije kojoj možemo proslijediti nula ili više izraza, razdvojenim zarezima. Ova funkcija, sve izraze koje joj proslijedite, prvo izračuna a zatim ih, prije nego što ih ispiše na ekranu konvertuje u string.

Pojednostavljena sintaksa naredbe:

```
print(value(s), sep= ' ', end = '\n')
```

Parametri koje ćemo koristiti:

value(s) : *Bilo koje vrijednosti. Biće konvertovane u string prije štampanja*

sep='separator' : *(opciono) Specificira način na koji se razdvajaju vrijednosti prilikom ispisivanja, ukoliko ih ima više od jedne.Default :' '*

end='end': *(opciono) Specificira šta da se ispiše na kraju. Default : '\n' (novi red)*

```
# Jedna vrijednost proslijeđena:
print("TinkerIgramiranje")

x = 2020
# Dvije vrijednosti proslijeđene:
print("x =", x)

# Štampa bez blankova između vrijednosti:
print('M', 'N', 'E', sep ='')

# korišćenje end argumenta:
print("Python", end = '@')
print("Tinker")
```

izlaz:
TinkerIgramiranje
x = 2020
MNE
Python@Tinker

Ispisivanje bez novog reda

Sigurno će ponekad u programima biti potrebno odštampati vrijednosti dvije ili više promjenljivih bez prelaska u novi red.

Pajton naredba print() je definisana tako da se, ukoliko ne navedemo drugačije, završava sa prelaskom u novi red, odnosno **prelazak u novi red se automatski izvrši.** Ukoliko to želimo promijeniti, možemo koristiti **end** parametar funkcije **print().**

end parametar u print() funkciji

Funkcija print() ima parameter **end** koji nam omogućava da podesimo karakter ili string koji želimo ispisivati **na kraju izvršavanja svake print() naredbe.** Kao što smo ranije rekli default vrijednost parametra **end** je '\n' (novi red)!

Primjer:

```
# završava se blankom <space>
print("Dobrodošli na" , end = ' ')
print("Tinker radionice!", end = ' ')
```

izlaz:
Dobrodošli na Tinker radionice!

sep parametar u print() funkciji

Komandu print() možemo koristiti za ispisivanje više vrijednosti koje nazivamo argumentima. Čime će se te vrijednosti, odnosno argumenti, razdvojiti prilikom ispisivanja, definišemo **sep** parametrom naredbe print(). Default vrijednost ovog parametra je blanko (" "), ali može biti izmijenjena u bilo koji drugi **karakter, string ili cijeli broj.**

Primjeri:

```
#separator  je "prazan string"
print('M','N','E', sep='')
```

izlaz:
`ispisuje MNE`

```
#primjer formatiranja datuma
print('01','06','2020', sep='-')
```

izlaz:
`ispisuje 01-06-2020`

```
#another example
print('Vjezbamo','Tinker    radionicama',
sep='@')
```

izlaz:
Vjezbamo@Tinker radionicama

KOJE TIPOVE PODATAKA POZNAJE PAJTON?

Izdvojili smo osnovne tipove podataka, odnosno vrijednosti, koje ćemo koristiti:

TIP PODATKA	PAJTON TIP
Cjelobrojni broj	**integer**
Realni broj	**float**
Logička vrijednost	**bool**
Niz znakova, niska	**string**

Brojevi

Cijeli brojevi

Programski jezik Pajton podržava rad sa cijelim brojevima. **Cijeli brojevi** mogu biti negativni i njih zapisujemo na isti način kao u matematici (npr. -100, 0, 101). Na engleskom se cijeli broj naziva *integer* i zato za cijelobrojne tipove podataka u Pajtonu kažemo da su tipa **int.**

Realni brojevi

Realne brojeve koristimo na isti način kao u matematici. Razlika je u tome što umjesto decimalnog zareza koristimo decimalnu tačku (npr. 28,22 u Pajtonu zapisujemo kao 28.22). Za realne brojeve u računaru se kaže da su zapisani u obliku pokretnog zareza (na engleskom: floating point), pa se ovaj tip naziva **float.**

Logičke vrijednosti

Promjenljiva može da sadrži i istinitosne, odnosno logičke, vrijednosti:

- tačno (True) i
- netačno (False)

Da li primjećuješ da su logičke vrijednosti napisane **velikim slovom**? Odlično!

Ovakve vrijednosti nazivamo logičkim ili bulovskim vrijednostima, odnosno kažemo da su one tipa **bool**.

Ime su dobile u čast velikog matematičara Džordža Bula koji je prvi u matematički račun uveo istinitosne vrijednosti. Iako nismo još pomenuli *grananje* i naredbu *if...else*, zbog bliskosti Pajtona sa engleskim jezikom ćemo biti slobodni da, kao nagovještaj načina korišćenja logičkih vrijednosti, damo sljedeći primjer:

```
pada_kisa = True
if pada_kisa:
   print("Ne zaboravi kišobran!")
else:
   print("Ne treba ti kišobran.")
```

Tekstualne vrijednosti, niska, string

Pored cijelobrojnih i realnih brojeva, **tekstualni tip** je jedan od osnovnih tipova podataka u programiranju.

*Sve znakove jednim imenom nazivamo **karakteri.***

Stringom, odnosno **niskom**, nazivamo tekstualni tip podataka kao i svaki izraz čija je vrijednost tekstualni tip.

Tekst ili string je niz karaktera (znakova) koji mogu biti:
- velika i mala slova,
- cifre,
- znaci matematičkih operacija,
- interpunkcijski znaci,
- zagrade,
- specijalni karakteri poput @, #, $, %, ^, &, ...

Stringovi, odnosno tekstualne vrijednosti se pišu između:
- navodnika ili
- apostrofa ili
- dvostrukih apostrofa ili
- trostrukih apostrofa.

Primjer:

```
naziv = "Gorski vijenac"
pisac = 'Petar II Petrović Njegoš'
planeta=''Jupiter''
satelit='''Mjesec'''
```

Šta ako naš string sadrži navodnike ili neke druge specijalne karaktere? Tada je potrebno te karaktere posebno označiti kosom crtom: \ .

Primjer: Napisati program koji ispisuje tekst sljedeće sadržine:
Rekao je: "Bez alata nema ni zanata."

```
print("Rekao je:\"Bez alata nema zanata\"")
```

ARITMETIČKE OPERACIJE U PAJTONU

Riječ aritmetika potiče od grčke riječi ἀριθμός tj. aritmos koja znači broj, brojanje, računanje.

Računar (ili kompjuter) je sprava koja računa. Ta sprava je napravljena tako da može **veoma brzo** da izvodi **računske operacije** (u matematici kažemo aritmetičke operacije) nad **brojevima**.

Osnovne operacije u Pajtonu su:
- sabiranje,
- oduzimanje,
- množenje,
- dijeljenje,
- stepenovanje,
- grupisanje.

MATEMATIČKA OPERACIJA	PAJTON OPERATOR	PRIMJER	REZULTAT
sabiranje	+	3+2	5
oduzimanje	-	3-2	1
množenje	*	3*2	6
dijeljenje	/	6/2	3
stepenovanje	**	4 ** 2	16
cijelobrojno dijeljenje	//	9//4	2 (9=2*4+1)
ostatak pri cijelobrojnom dijeljenju	%	9%4	1 (9=2*4+1)

Kao što ste već mogli primijetiti, u programskom jeziku Python, kao i u matematici, zagrade koristimo za grupisanje u aritmetičkim izrazima. Na primjer:

```
(23 - 12) * (329 +1)
```

Ukoliko nema zagrada, pravila računanja vrijednosti izraza su ista kao u matematici:

- operacija stepenovanja se izvršava prije svih ostalih operacija. Ukoliko imamo više operacija stepenovanja, jedna za drugom, onda se izvršavaju sa desna nalijevo.
- množenje, dijeljenje, cijelobrojno dijeljenje i ostatak se primjenjuju prije sabiranja i oduzimanja. Kada ih ima više u nizu, izvršavaju se sa lijeve prema desnoj strani, redom.

Da li Pajton stvarno zna da računa? Da li Pajton zna da sabere brojeve 2 i 3 i da ispiše rezultat na ekranu? Pokušaj da pogodiš pravu komandu u Pythonu koja radi upravo to.

U programskom jeziku Python, program bi mogao da izgleda jednostavno poput sljedećeg:

```
2+3
```

izlaz:
5

ili

```
print(2+3)
```

izlaz:
5

Provjerimo da li ispravno računa i druge izraze. Na primjer, pokušajmo izračunati koliku vrijednost dobijemo kada tri puta uvećamo razliku proizvoda i količnika brojeva 6 i 3?

```
((6*3)-(6/3))*3
```

izlaz:
48.0

Računari razlikuju cijele i realne brojeve i različito ih zapisuju u svojoj memoriji i na različit način interno računaju sa njima. Na primjer, u programiranju 5.0 i 5 nije sasvim isto i pored toga što je 5.0==5 tačno, odnosno vrijednosti 5.0 i 5 su jednake.

```
5.0 == 5
```

izlaz:
True

Da bi provjerili kojeg su tipa vrijednosti 5.0 i 5, u Pythonu možemo koristiti ugrađenu funkciju **type**. Pogledajmo primjere:

```
type(5.0)
```

izlaz:
<class 'float'>

```
type(5)
```

izlaz:
<class 'int'>

Vidimo da je vrijednost 5.0 tipa float, a vrijednost 5 tipa integer, odnosno da je prvi broj realan, a drugi cijeli broj. Treba obratiti pažnju na to da tip vrijednosti rezultata neke operacije može zavisiti i od same operacije. Na primjer, ukoliko u Pajtonu dijelimo dva broja, rezultat će uvijek biti realan broj, čak i kada dijelimo cijele brojeve koji pri dijeljenju daju ostatak nula. Sigurno će postojati situacije kada će nam biti potrebno da rezultat dijeljenja cijelih brojeva bude cio broj i tada možemo, na primjer, koristiti operaciju cijelobrojnog dijeljenja: // .

Sa druge strane, kada u Pajtonu sabiramo, oduzimamo ili množimo dva cijela broja, rezultat koji dobijemo je, takođe, cijeli broj.

RELACIJSKI OPERATORI (POREĐENJE)

Da bi u programu mogli da uporedimo veličine, koristimo **operatore**: `<,>,>=,<=,==,!=.`

Primjećuje se da su relacijski operatori u programskom jeziku Pajton slični matematičkim.
Rezultat relacijskih operatora su logičke vrijednosti tačno ili netačno, odnosno True ili False.

a < b provjerava da li je a manje od b
a > b provjerava da li je a veće od b
a >= b provjerava da li je a veće ili jednako od b
a <= b provjerava da li je a manje ili jednako od
a == b provjerava da li je a jednako b
a != b provjerava da li je a različito od b

LOGIČKI OPERATORI (KOMBINOVANJE LOGIČKIH USLOVA)

Uslovi se mogu kombinovati. Obično to činimo koristeći riječi:

- i (na engleskom: AND)
- ili (na engleskom: OR)
- ne (na engleskom: NOT).

Na primjer: Zamislimo da želimo opisati sljedeće: Ako ne pada kiša i prijavi se barem 20 učenika iz naše škole ili na ekskurziju idu bar dvije škole zajedno, ići ćemo na dvodnevnu ekskurziju.

Možemo izdvojiti tri uslova:
- uslov1: pada kiša
- uslov2: broj prijavljenih učenika iz naše škole je veći ili jednak 20
- uslov3: na ekskurziju idu bar dvije škole zajedno

Uslov iz primjera možemo zapisati na sljedeći način:

ne uslov1 **i** (uslov2 **ili** uslov3).

U programskom jeziku Python, to bi smo zapisali kao:

not uslov1 and (uslov2 **or** uslov3).

Logički operatori u Pythonu su:

- AND
- OR
- NOT.

Pogledajmo rezultate izvršavanja logičkih operatora AND, OR i NOT u zavisnosti od operanada A i B:

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

RAD SA TEKSTUALNIM VRIJEDNOSTIMA (NISMAKA, STRINGOVIMA)

Nadovezivanje stringova

Kako bismo stringove spajali, odnosno nadovezivali jedan na drugi, koristimo operaciju konkatenacije stringova. Da, **konkatenacija** je naziv operacije nadovezivanja. Ova operacija se označava znakom **+**, kao i operacija sabiranja brojeva.

```
a = 'infor' + matika
print(a)
```

Možemo "sabirati" dva brojevna izraza ili dva stringa, ali ukoliko pokušamo "sabrati" brojevni izraz sa stringom dobićemo grešku.

Množenje stringova

String možemo "pomnožiti" sa cijelim brojem i rezultat te operacije je novi string koji se dobija ponavljanjem polaznog stringa onoliko puta kolikim smo ga brojem pomnožili. Redosljed operanada nije bitan, odnosno string možemo "pomnožiti" cijelim brojem slijeva ili zdesna.

```
print(10 * '#')      # štampa se '##########'
print('#' * 10)      # štampa se '##########'
```

U prethodnim primjerima, string '#' množimo 10 puta i dobijamo novi string '##########' koji ispisujemo.

Dužina stringa: len()

Ukoliko nam je potreban podatak o dužini stringa tj. o broju karaktera od kojih se string sastoji, možemo koristiti len() funkciju.

```
s = 'Pozdrav svima!'
print(len(s))   ## dužna stringa iz primjera je 14
```

Kako doći do karaktera iz string-a: []

Da bi pristupili svakom karakteru iz stringa pojedinačno ili pak podskupu uzastopnih karaktera stringa, koristimo uglaste zagrade: []. U stringu su karakteri sa lijeve prema desnoj strani stringa obrojani, odnosno indeksirani, brojevima od 0 do len(s)-1.

```
        ## Karakteri su obrojani počevši sa nulom:
s[0]    ## 'P'
s[1]    ## 'o'
s[4]    ## 'd'
s[8]    ## 's'
s[13]   ## '!' – posljednji karakter na poziciji len(s)-1
s[14]   ## Greška: ERROR, index out of bounds
```

Dio(parče) stringa: [:]

"Parče" je u Pajtonu moćan način da se dobije dio stringa, odnosno podstring. Sintaksa **s[i:j]** označava da podstring počinje od pozicije sa indeksom ***i***, uključujući i tu poziciju i proteže se do pozicije sa indeksom ***j***, ne uključujući nju.

```
s = 'Tinker'
#  012345  ## indeksi pozicija karaktera u
#  stringu 'Tinker'
s[0:4]  ## 'Tink' -počevši od pozicije 0, do
#  pozicije sa indeksom 4, ne uključujući i nju
s[1:3]  ## 'in'
```

Ukoliko izostavimo prvi broj u uglastim zagradama, time označavamo da želimo da naš dio stringa počinje od samog početka stringa čiji dio uzimamo. Slično, ukoliko izostavimo drugi broj, odnosno parametar, u uglastim zagradama, time označavamo da želimo da se naš dio stringa proteže do samog kraja stringa čiji dio uzimamo.

Kombinovanje: sabiranje dijelova stringa

Možemo kombinovati sintaksu za dobijanje odgovarajućeg dijela stringa sa operacijom sabiranja „+" :

```
a = 'Zdravo!'
b = 'Tink Tinker'

# Generišemo c od:
# prvih 6 karaktera stringa a, praćenih sa
# dijelom stringa b.
# Ako dodamo još na kraju string "!" dobićemo
# string: "Zdravo Tinker!"
c = a[:6] + b[len(b) - 7:] + "!"
```

Detaljnije o indeksima

Korišćenje **nepostojećeg indeksa** u Pajtonu, npr. s[100], za s="Tinker", prouzrokovaće *runtime error*. Sa druge strane, definisanje dijela stringa radi drugačije.

Ukoliko je indeks "out of bounds", on se ignoriše i uzima se dio od početka ili do kraja stringa.

```
s[0:20] # rezultat je "Tinker"
```

Pajton podržava korišćenje **negativnog broja za indeksiranje string-a:**

indeks -1 odnosi se na posljednji karakter u stringu,
indeks-2 odnosi se na pretposljednji karakter u stringu, itd.

Drugim riječima:

indeks -1 nam daje isti rezultat kao len(s)-1,
indeks -2 nam daje isti rezultat kao len(s)-2,
itd.

Na primjer:

```
s = 'Tinker'
#   -54321    ## negativni brojevi kao indeksi
s[-2:]  ## 'er', dio počinje sa drugim
#  karakterom od kraja
s[:-3]  ## 'Tin', dio do trećeg karaktera od
#  kraja, ne uključujući i njega
```

Petlje po stringovima

Jedan način da koristimo petlju nad stringom je korišćenjem funkcije **range(n)**, koja za dato *n* vraća niz *0, 1, 2, 3 ... n-1*. Ove vrijednosti su potpuno odgovarajuće indeksima karaktera u stringu, tako da ćemo u petlji (o petljama će biti više riječi kasnije):

```
for i in range(len(s)):
```

"provrtjeti" promjenljivu i kroz vrijednosti *0, 1, 2, ... len(s)-1*, i omogućiti nam da po jedan put pristupimo svakom indeksu stringa.

Svakom karakteru stringa, redom, od prvog do posljednjeg, sa lijeva na desno, možemo pristupiti koristeći u prethodnoj petlji s[i].

```
s = 'Hello'
rezultat = ''
for i in range(len(s)):
    # Dodajemo svaki karakter s[i]
    # promjenljivoj rezultat
    rezultat = rezultat + s[i] + '.'
print(rezultat)
```

Prethodni kod će nam ispisati string 'H.e.l.l.o.'

Ako imamo indeks odgovarajućeg karaktera u stringu, tokom kretanja kroz petlju, veoma jednostavno možemo pristupiti i drugim karakterima u blizini *i*-te pozicije, ili slično. Na primjer, karakter koji je lijevo od i-te pozicije se nalazi na poziciji *i-1* i možemo mu pristupiti sa s[i-1].

Drugi način da pristupimo svakom karakteru u stringu je sa regularnom for petljom:

```
rijec = 'Tinker'
rezultat = ''
## slovo će u narednoj for petlji imati
## vrijednosti: 'T', zatim 'i', ...
for slovo in riječ:
    rezultat = rezultat + slovo
```

Iako na ovaj način, takođe možemo pristupiti svakom karakteru u stringu, imamo manje fleksibilnosti nego u prethodnom primjeru.

KONTROLA TOKA

Uslovi i grananje: šta ako? (if-elif-else)

Često se susrijećemo sa tim da je nešto moguće raditi samo ukoliko je ispunjen neki unaprijed definisani uslov. Na primjer:

- Ako je visina djeteta veća od 130cm, onda ono može ući u akva-park.
- Ako učenik dobije više od 80 poena, onda će dobiti ocjenu 5.
- Ako je osoba starija od 18 godina i ima važeću vozačku dozvolu, onda može da upravlja motornim vozilom.

U programiranju je slično. *Neke naredbe (određeni dio programa) se mogu izvršavati samo ako je ispunjen neki uslov (ili više uslova).*

U pomenutim slučajevima koristimo **if** naredbu. *Riječ **if** na engleskom znači **"ako"**.*

Najjednostavniji oblik naredbe if

```
if uslov:        # ako je uslov ispunjen:
  naredba_1      #   izvrši naredbu 1
  ...            #   ...
  naredba_n      #   izvrši naredbu n
```

Ne zaboravi:
- *Nakon uslova napisati dvije tačke ":"*
- *Naredbe koje se izvršavaju ukoliko je uslov ispunjen trebaju se uvući (tab)!*

A šta ako uslov nije ispunjen? Korišćenje konstrukcije if-else

Na primjer: ukoliko je **a** djeljivo sa **b**, treba ispisati "DJELJIVO", u suprotnom ispisati "NIJE DJELJIVO".

Tada koristimo naredbu oblika **if-else**. *Riječ **else** na engleskom znači **"u suprotnom"**.*

```
if uslov:      # ako je uslov ispunjen:
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_n    #   izvrši naredbu n
else:          # u suprotnom:
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_m    #   izvrši naredbu m
```

Ne zaboravi:
- *Nakon **else** napisati dvije tačke ":"!*
- *Naredbe koje se izvršavaju ukoliko je uslov nije ispunjen trebaju se uvući (tab)!*

Sa naredbom if-else, tok programa se razdvaja na dva dijela:
- mogu se izvršiti naredbe ispod if (ukoliko je uslov ispunjen) ili se
- mogu izvršiti naredbe ispod else (ukoliko uslov nije ispunjen).

Iz tog razloga kažemo da se **tok programa na tom mjestu grana**, a naredbu **if-else** nazivamo **naredbom grananja**.

A šta ako imamo više pitanja koje želimo postaviti?

```
if uslov:      # ako je uslov ispunjen:
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_n    #   izvrši naredbu n
elif (uslov):  # inače, ako je uslov ispunjen:
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_m    #   izvrši naredbu m
...

elif (uslov):  # inače, ako je uslov ispunjen:
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_p    #   izvrši naredbu p

else:          # u suprotnom (inače):
  naredba_1    #   izvrši naredbu 1
  ...          #   ...
  naredba_m    #   izvrši naredbu m
```

Sa naredbom if-elif-else, tok programa se razdvaja na više dijelova:
- mogu se izvršiti naredbe ispod **if** (ukoliko je **uslov ispunjen**) ili se
- mogu izvršiti naredbe ispod jednog **elif** (ukoliko **svi uslovi iznad njega nisu ispunjeni**, a **uslov iz tekućeg elif je ispunjen**),
- mogu izvršiti naredbe ispod **else** (ukoliko **svi navedeni uslovi nisu ispunjeni**).

*Obratite pažnju kada koristite if-elif-else kontrukciju: uvijek se izvršava **najviše jedan blok naredbi** u kodu!*

Ponavljanje. Petlje su zabavne (for, while)

Možeš li to da ponoviš?

Kada koristimo "for" petlju?

"For" petlje obično koristimo kada izvršavanje nekog bloka komandi, odnosno grupe komandi, želimo da ponovimo tačno određeni broj puta, i pri tome brojač, odnosno brojačka promjenljiva, prolazi
- *kroz pravilnu seriju brojeva iz zadatog intervala i sa*

zadatim "korakom" ili
- *kroz sve karaktere stringa ili*
- *kroz sve elemente liste (a, vidjećemo u narednim serijalima radionica, i kroz elemente niske ili riječnika).*

Dakle, ukoliko želimo da se nešto ponovi *n* puta, u Pajtonu to možemo uraditi koristeći "for" petlju na sljedeći način:

```
for i in range(n):
    #naredba 1
    #naredba 2
    #...
    #naredba m
```

Tokom izvršavanja ove "for" petlje, promjenljiva *i* će uzimati redom sljedeće vrijednosti: *0,1,2* itd., sve do vrijednosti *n-1*. U prvom izvršavanju bloka naredbi, promjenljiva *i* će imati vrijednost *0*, u drugom *1*, u trećem *2*, itd. U posljednjoj iteraciji, odnosno u posljednjem izvršavanju bloka naredbi, promjenljiva *i* će imati vrijednost n-1.

Pozivajući funkciju range(n) kreiramo listu od brojeva 0,1,2,...,n-1, iz koje promjenljiva *i uzima vrijednosti, redom, jednu po jednu. Funkciju range() možemo pozvati prosljeđujući joj i dvije ili tri vrijednosti (argumenta, parametra). Ukoliko je pozovemo navodeći dva argumenta range(x,y), vrijednosti koje će uzimati brojačka promjenljiva su svi cijeli brojevi iz intervala [x,y). Ukoliko funkciju pozovemo navodeći tri argumenta, range(x,y,k) brojačka promjenljiva će uzimati svaki k-ti element iz intervala cijelih brojeva [x,y), počevši od x.*

Ugnježdene petlje

Kada imamo blok naredbi koje želimo da izvršimo x puta, a u okviru njega blok naredbi koje želimo da izvršimo y puta, to nazivamo „ugnježdenim petljama" ili „petljom u petlji". Na primjer:

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

Break

U Pajtonu možemo iz „for" petlje izaći prije nego što se realizuju sve iteracije i to možemo učiniti korišćenjem naredbe ***break*** koja automatski prekida izvršavanje petlje, tako da se izvršavanje programa nastavlja od prve naredbe nakon "for" petlje.

```
for x in range(3):
  if x == 1:
        break
  print(x)
```

For..Else

Takođe i u "for" petlji možemo koristiti *else* ukoliko želimo da se grupa naredbi izvrši nakon regularnog izlaska iz "for" petlje, bez prijevremenog prekidanja njenog izvršavanja korišćenjem naredbe *break*.

```
for x in range(3):
    print(x)
else:
    print(Na kraju je x = ',x)
```

Primjeri „for" petlje

Petlja po stringovima

```
string = "Tinker Igramranje"
for karakter in string:
    print(karakter)
```

Petlja po listama

```
lista = ['Tinker', 2020, 'Igramiranje']
for element in lista:
    print(element)
```

Petlja po listi lista

```
lista_lista = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for l in lista_lista:
    for e in l:
        print(e)
```

Napomena u vezi sa range() funkcijom

Iako se range funkcija često koristi u "for" petlji, ona nije dio sintakse "for" petlje, već je riječ o ugrađenoj (built-in) Pajton funkciji koja vraća listu koja odgovara definisanom pravilu.

"For" petlju možemo izvršavati direktno nad listom vrijednosti. Brojačka promjenljiva može uzimati vrijednosti iz liste koja je rezultat funkcije range(len(lista)):

```
lista = ['a', 'b', 'c', 'd']
for i in range(len(lista)):
    print(lista[i])
```

ili, npr. brojačka promjenljiva može uzimati redom vrijednosti iz same liste.

```
lista = ['a', 'b', 'c', 'd']
for v in lista:
    print(v)
```

U slučajevima kada nam je, pored opisanog, potrebno i da bolje kontrolišemo ponavljanja, odnosno u slučajevima kada je potrebno da provjeravamo ispunjenost nekog uslova, koristićemo takozvanu uslovnu petlju ili "while" petlju.

While petlja

Slično kao kod "for" petlje, "while" petlju koristimo kada želimo da ponavljamo određeni blok naredbi ali, za razliku od "for" petlje, "while" petlja se neće "provrtjeti" n puta, već će se "vrtjeti", sve dok je definisani uslov ispunjen. Ukoliko je pomenuti uslov u samom startu False (netačan, odnosno nije ispunjen), blok naredbi u „while" petlji se neće izvršiti ni jednom.

Kako sa „for" petljom u Pajtonu možemo napisati dosta toga, "while" petlja se možda rijeđe koristi.

„While" petlja, na primjer, može biti korisna ukoliko želimo da provjeravamo vrijednosti koje korisnik programa unosi. Na primjer:

```
n = input("Unesi riječ 'Tinker':")
while n!= Tinker:
    n = input("Nije unijeta tražena riječ.
Unesi riječ 'Tinker':")
```

Prethodni kod možemo izmijeniti, tako što ćemo kompletnu logiku smjestiti u "while" petlju:

```
while True:
    n = input("Unesi riječ 'Tinker':")
    if n == 'Tinker':
        break
```

Kako smo kao uslov naveli *True, kod će se izvršavati sve dok se, u slučaju kada korisnik unese string "Tinker", ne prekine izvršavanje „while" petlje sa naredbom break.*

Za razliku od "for" petlje, "while" petlju koristimo kada ispunjenost nekog uslova trebamo provjeravati na početku svake nove iteracije ili u slučaju kada nam je potrebna "beskonačna petlja", tj. kada je želimo da blok koda ponavljamo "zauvijek".

Primjeri

„for" petlja od 0 do 2, ponavlja se 3 puta

```
for x in range(0, 3):
    print("Ovo je ", x , ". put")
```

"while" petlja od 1 do beskonačno. Izvršava se "zauvijek"

```
x = 1
while True:
    print("Do beskraja i nazad! Sve bliže
bliže. x , ". put")
    x += 1
```

Možemo primijetiti da se "for" petlja izvršava konačan broj puta, dok se "while" petlja izvršava dok se uslov while petlje ne promijeni. U navedenim primjerima "for" petlja se izvršava 3 puta, a logički uslov je logička vrijednost (boolean), konstanta: True, koja se nikada neće promijeniti, tako da bi se teorijski "while" petlja trebala izvršavati beskonačno dugo.

FUNKCIJE, RECIKLAŽA: ISKORISTI KOD!

Vjerovatno ste se u matematici susreli sa pojmom **funkcije**. One predstavljaju preslikavanje ulaznih vrijednosti u jednu ili više izlaznih vrijednosti. Ulazne vrijednosti funkcije nazivamo **parametrima** ili **argumentima** funkcije.

Na primjer, funkcija obim_kvadrata(a), može kao ulazni parametar da dobije dužinu stranice kvadrata i da kao rezultat, odnosno izlaznu vrijednost vrati obim tog kvadrata: 4*a. U matematici bi ovakvu zavisnost između dužine stranica i obima kvadrata opisali kao funkciju f(a)=4*a.

Slično tome, izračunavanje obima pravougaonika na osnovu dužina njegovih stranica *a* i *b*, u matematici bi opisali sa funkcijom *p*, takvom da je: p(a,b)=2*a+2*b. Dakle, funkcija koju smo nazvali *p* ima dva ulazna parametra *a* i *b* i vraća rezultat vrijednost izraza 2*a+2*b.

Funkcije u Pajtonu

Funkcije u programskom jeziku Pajton su grupe naredbi koje izvršavaju neki zadatak. Ta grupa naredbi mora da se imenuje, kako bi funkcija bila prepoznatljiva u ostalom dijelu programskog koda. Funkcije omogućavaju višestruku upotrebu programskog koda, skraćivanje dužine programa, bolju organizaciju i lakšu čitljivost programa njegovom podjelom na manje cjeline.

Funkcija se u Pajtonu definiše posebnom konstrukcijom sljedećeg oblika:

```
def <naziv_funkcije> (<lista parametara>):
<blok naredbi>
<return vrijednost>
```

Definicija funkcije sastoji se od:

- *zaglavlja funkcije* koje čine
  - ključna riječ *def*, zatim
  - naziv funkcije nakon koga slijedi
  - lista parametara navedenih u zagradama i koji su, ukoliko ih je više od jednog, razdvojeni zarezom i
- *tijela funkcije,* koje čini blok naredbi koje izvršavaju neki zadatak.

Funkcije imaju *parametre* odnosno *listu parametara* čije vrijednosti se koriste prilikom njenog izvršavanja. Promjenljive u listi parametara koje preuzimaju aktuelne vrijednosti iz poziva funkcije su *parametri*, a same vrijednosti koje se prenose funkciji prilikom njene upotrebe su *argumenti* funkcije.

Funkcija se završava posljednjom naredbom ili naredbom *return*, a rad programa se nastavlja od prve sljedeće naredbe iza poziva funkcije.

Funkcije mogu, ali i ne moraju da kao rezultat vrate neku vrijednost.

- Funkcija može da vrati neku vrijednost kao rezultat pomoću posebne naredbe u tijelu funkcije koja ima oblik:
  return *vrijednost* .
- Ukoliko funkcija treba da vrati više vrijednosti onda ova naredba ima oblik **return vrijednost1, vrijednost_2, vrijednost_3,..., vrijednost_n.**
- Ukoliko funkcija nema povratnu vrijednost ona implicitno vraća *None.*

Funkcija se u ostatku programskog koda koristi pomoću naziva funkcije i liste aktuelnih argumenta funkcije.

Primjeri

Paran ili neparan?

Napisati funkciju koja provjerava da li je unešeni broj paran ili neparan:

```
def DaLiJeParan(n):
 if n%2==0:
    print("Uneseni broj je paran.", n)
 else:
    print("Uneseni broj je neparan.", n)
```

Ako pozovemo ovu funkciju: **paran(3)** dobićemo rezultat funkcije, odnosno odgovor: **Uneseni broj je paran, 3.**

Koji je veći?

Napisati funkciju koja vraća vrijednost većeg od dva zadana broja:

```
def max(n1,n2):
  if n1>n2:
    rezultat=n1
  else:
    rezultat=n2
return rezultat
```

Ako pozovemo ovu funkciju **max(5,6)** dobićemo rezultat funkcije, odnosno odgovor: **6.**

Napomene

*U Pajtonu se iz jedne funkcije može pozvati druga funkcija, tako da su dozvoljeni tzv. **ugnježđeni pozivi**. Takođe je dozvoljeno da funkcija sadrži poziv same sebe, odnosno dozvoljeni su i **rekurzivni pozivi**.*

Argumenti se funkciji mogu prenositi:

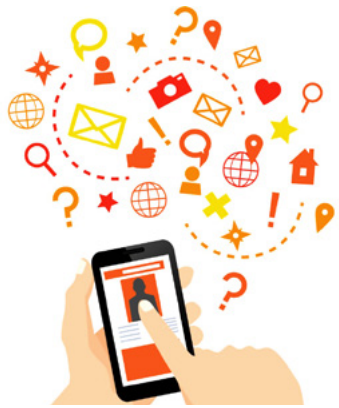
- **po poziciji**(redosledu) ili
- **prema njihovom nazivu**, u obliku <naziv>=<vrijednost> .

Na primjer, funkcija def f(n1,n2,n3), može se pozvati sa vrijednostima argumenata:

- po poziciji:
  f(1,23,456) ili
- po nazivima argumenta koji se mogu navoditi proizvoljnim redosljedom
  f(n3=456, n1=1, n2=23).

Ugrađene funkcije

Ove funkcije su već "ugrađene" u sam Pajton jezik, odnosno već postoje u samom jeziku i možemo ih po potrebi koristiti, odnosno pozivati u programima koje pišemo.



Pogledajmo par primjera ugrađenih funkcija:

PAJTON FUNKCIJA	OPIS FUNKCIJE
int()	pretvaranje vrijednosti u cijeli broj, odbacujući decimalu
float()	pretvaranje vrijednost u realni broj
bool()	za rad sa logičkim tipovima podataka (true, false)
string()	pretvaranje u znakovni tip podataka - niz znakova
math. sqrt()	računa i vraća korijen zadatog broja
abs()	računa i vraća apsolutnu vrijednost zadatog broja
min()	vraća (naj)manji od proslijeđenih brojeva npr. min(22,23) je manji od brojeva 22 i 23, odnosno broj 22
max()	vraća (naj)veći od proslijeđenih brojeva npr. max(22,23) je veći od brojeva 22 i 23, odnosno broj 23

LISTE

Lista u Pajtonu može se sastojati od proizvoljnog broja "stvari", odnosno elemenata. Svaki element ima svoje mjesto u listi. Redni broj pojavljivanja elementa u listi nazivamo njegovim indeksom, s tim što brojanje počinjemo od 0, pa tako prvi element liste ima indeks 0, drugi element ima indeks 1, treći 2 itd.

Šta možemo raditi sa listom?

Dužina liste

Kao što smo to ranije radili sa stringovima, i kod lista možemo koristiti **len**() funkciju kako bismo dobili podatak o dužini liste, tj. o broju elemenata koje ta lista sadrži.

Pristupanje elementima liste

Na sličan način kao kod stringova, i u radu sa listama možemo korisiti uglaste zagrade [ ] da bi pristupili elementima liste, ili kako bismo im promijenili vrijednost:

```
a = [‘Pozdrav’, ‘svima’, ‘!’]  # lista sa 3 elementa
len(a)                ## 3
a[0]                  ## ‘Pozdrav’
a[2] = ‘!!!’          ## Možemo promijeniti vrijednost
                      ## elementa liste
```

Dodavanje elementa u listu

Za dodavanje elementa na kraj liste, koristimo metodu **append**(). Pogledajmo primjer:

```
a = [‘Zdravo’, ‘svima’]
a.append(‘!!!’)  ## dodavanje elemenata na kraj
                 ## liste. rezultat: [‘Zdravo’, ‘svima’, ‘!!!’]
```

Sortiranje liste

Da bi elemente liste poređali po alfabetskom redosljedu, možemo koristiti funkciju sorted (lista) koja za proslijeđenu listu vraća novu listu koja sadrži iste te elemente, ali poređane u rastućem redosljedu:

```
## ako je a = [‘hvala’, ‘tinker’, ‘algoritam’, ‘bold’]
b = sorted(a)
# lista b će biti [‘algoritam’, ‘bold’, ‘hvala’,
# ‘tinker’],
# lista a ostaje nepromijenjena
```

Petlje i liste

Najjednostavniji način da pristupimo elementima liste je koristeći petlju. Na primjer, da bismo pristupili svakom elementu liste a, počevši od prvog do posljednjeg elementa liste, možemo koristiti naredbu:

```
for element in a:
```

Pogledajmo konkretan primjer:

```
a = [1, 2, 3]
zbir = 0
for element in a:  ## proći ćemo kroz vrijednosti
                   ## 1, 2, 3
    zbir = zbir + element
```

Drugi način da koristimo petlju za “prolazak kroz listu“ je korišćenjem **range**(n) funkcije koja vraća vrijednosti 0, 1, 2, ... n-1, pa se tako u naredbi

```
for i in range(len(list)):
```

vrijednosti indeksa “provrte” na odgovarajući način: od nula do (dužine liste -1):

```
a = [‘Zdravo’, ‘svima’, ‘!!!’]
rezultat  = ‘’
for i in range(len(a)):
  # i će uzimati redom vrijednosti
  # 0, 1, 2 ... len(a)-1
  # Koristimo a[i] da pristupimo elementu
  # liste na I-toj poziciji.
  # Sabiramo vrijednosti elementa liste,
  # povezujemo ih u string rezultat.
  rezultat = rezultat + a[i]
```

Iako imamo fleksibilnost u pristupanju elementima koji su lijevo (a[i-1]) ili desno (a[i+1]) od i-tog elementa, trebamo voditi računa da je maksimalna vrijednost indeksa len(a)-1.

Dijelovi liste

U Pajtonu možemo dobiti i dio liste, na sličan način kao što smo to radili sa stringovima.

- **Lista – uređena kolekcija koja se sastoji od jedne ili više vrijednosti koje ne moraju biti istog tipa i koje su, unutar uglastih zagrada, međusobno razdvojene zarezom.**

N1

OUTPUT, KOMANDA PRINT()

Zdravo svijete
Napiši program koji ispisuje tekst "Zdravo svijete!".

RJEŠENJE

```
print("Zdravo svijete!")
```

Četiri imena
Napiši program koji ispisuje četiri imena, svako u novom redu, sa praznim redom između svaka dva uzastopna imena.

RJEŠENJE

```
print("Danica")
print("")
print("Jovan")
print("")
print("Milica")
print("")
print("Marko")
```

Poslovica
Napisati program koji ispisuje tekst sljedeće sadržine: Rekao je: "Bez alata nema ni zanata."

RJEŠENJE

```
print("Rekao je:\"Bez alata nema ni zanata.\"")
```

Zbir dva broja
Napiši program koji računa i ispisuje zbir brojeva 2020 i 222.

RJEŠENJE

```
print(2020+222)
ili samo:
2020+222
```

PROMJENLJIVE

Proizvod i razlika brojeva
Ako je a=2000, b=222 napiši program koji računa i ispisuje njihov proizvod i razliku.

RJEŠENJE

```
a=2000
b=222
print(a*b)
print(a-b)
```

Tri puta veći broj
Ako je a=2020 napiši program koji ispisuje tri puta veći broj od a.

RJEŠENJE

```
a=2020
#print("Broj koji je 3 puta veći je: ")
print(a*3)
```

Dvostruko manji broj
Ako je a=20202020 napiši program koji ispisuje dvostruko manji broj od a.

RJEŠENJE

```
a=20202020
#print("Dvostruko manji broj je: ")
print(a/2)
```

Proizvod tri broja

Ako je a=2, b=20, c=2020 napiši program koji računa proizvod vrijednosti a,b i c.

RJEŠENJE

```
a=2
b=20
c=2020
#print("Proizvod brojeva je:", a*b*c)
print(a*b*c)
```

Tri drugarice i kolači

Tri drugarice, tri Milice, su pravile čokoladne mafine. Jedna je napravila 24 mafina, druga je napravila isto toliko, ali je pojela osminu, a treća je napravila tri puta više od šestine broja mafina koje je napravila prva Milica. Koliko ukupno mafina njih tri sada imaju?

RJEŠENJE

```
print(24+(24/8*7)+(24/6*3))
# ili:
m1=24
m2=24/8*7
m3=24/6*3
print(m1+m2+m3)
```

Vrijednost izraza

Izračunaj vrijednost izraza (22+222)*222+22*22 koristeći Python:

a) Bez korišćenja promjenljivih,

b) Koristeći promjenljive.

RJEŠENJE

a)
```
print((22+222)*222+22*22)
```

b)
```
x=22
y=222
print((x+y)*y+x*x)
```

Kad smo bili dvostruko mlađi

Promjenljivoj x pridruži svoje ime, promjenljivoj *g* broj svojih godina pa na ekranu ispiši poruku `"Ja se zovem <ime>. Kad sam bila/bio dvostruko mlađa/mlađi imala/imalo sam <broj godina> godina!"

RJEŠENJE

```
x="Ana"
g=20
print("Ja se zovem", x, ". Kad sam bila/bio dvostruko mlađa/mlađi imala sam bar", g//2, "godina!")
```

INPUT()

Kroz 10 godina

Promjenljivoj x pridruži svoje ime, promjenljivoj g broj svojih godina pa na ekranu ispiši poruku " Ja se zovem <ime> i za 10 godina ću imati <broj godina> godina."

Ime i broj godina unose se putem tastature.

RJEŠENJE

```
x=input()
g=int(input())
print ('Ja se zovem', x , 'i za 10 godina cu imati' ,g+10, 'godina.')
```

Svaki uvećaj za 5

Napiši program koji za svaki unijeti broj ispisuje vrijednost tog broja uvećanu za pet.

RJEŠENJE

```
a=int(input())
print(a+5)
```

Razlika bilo koja dva broja

Napiši program koji upisuje dva broja a,b i ispisuje njihovu razliku (a-b).

RJEŠENJE

```
a=int(input())
b=int(input())
print(a-b)
```

Fudbaler

Kada Stefan Savić, poznati crnogorski fudbaler, igra utakmice za reprezentaciju Crne Gore, svako jutro trenira u Podgorici, a popodne se vraća u svoj rodni grad Mojkovac. Temperatura u Podgorici je t1, a u Mojkovcu t2. Ukoliko znamo da je u Mojkovcu **uvijek hladnije**, napiši program koji će za Stefana izračunati kolika je razlika između temperatura u Podgorici i Mojkovcu.

Temperature t1 i t2 unesi putem tastature, svaku u zasebnom redu.

	PRIMJER
Ulaz	22 10
Izlaz	12

RJEŠENJE

```
tp=int(input())
tm=int(input())
print(tp-tm)
# ili:
t1=int(input('Unesi temperaturu vazduha u Podgorici:'))
t2=int(input("Unesi temperaturu vazduha u Mojkovcu:"))
print("'Razlika temperatura je:'",t1-t2)
```

Takmičenje mladih pijanista

Milica je učestvovala na takmičenju mladih pijanista i zanima je prosječna ocjena koju je osvojila. Na osnovu unijetih ocjena koje je dodijelio svaki član žirija, izračunaj i ispiši prosječnu ocjenu na takmičenju. Žiri se sastojao od četiri člana, koji su dodijelili redom sljedeće ocjene: a,b,c i d.

RJEŠENJE

```
a=int(input())
b=int(input())
c=int(input())
d=int(input())
print((a+b+c+d)/4)
```

Bajadere

Lazar je od mame i tate dobio n bajadera. Jednu je dao mami, jednu tati i dvije sestri. Ostale je sam pojeo. Napiši program koji će, za unijeto n, izračunati i ispisati koliko je Lazar pojeo bajadera.

RJEŠENJE

```
n=int(input())
print(n-4)
#n=int(input("Koliko je Lazar dobio bajadera?"))
#print('Pojeo je:',n-1-1-2,' bajadera', sep=' ')
#separator u prethodnoj print() naredbi je blanko!
```

Sekunde

Napiši program koji za unijeti broj sati, minuta i sekundi i ispisuje ukupan broj sekundi.

RJEŠENJE

```
sat=int(input())
min=int(input())
sek=int(input())
print(sat*3600+min*60+sek)
```

N1 DOMAĆI

Naredni paran broj

Napiši program koji za unijeti paran broj x ispisuje naredni paran broj (najmanji paran broj strogo veći od x).

RJEŠENJE

```
x=int(input("Unesi paran broj:"))
print("Naredni paran broj je ", x+2)
```

Metri u centimetre

Napiši program koji pretvara metre u centimetre, pri čemu se broj metara unosi kao cijeli broj.

RJEŠENJE

```
#Ovo je program koji pretvara metre u centimetre
m=int(input('Unesi broj metara(m):'))
print('To je',m*100, 'cm', sep=' ')
```

Prosječna ocjena na eksternom testiranju

Napiši program koji za tvoje unijete ocjene iz matematike, CSBH-a i izbornog predmeta računa i ispisuje prosječnu ocjenu.

RJEŠENJE

```
m=int(input("Matematika:"))
c=int(input('CSBH:'))
i=int(input('"Izborni predmet:""))
print("Tvoja prosječna ocjena je:", (m+c+i)/3,
sep=' ')
```

Riješi jednačinu

Napiši program koji za unijete vrijednosti a, b i c rješava sljedeću jednačinu: *a*x+b*c = 0, po nepoznatoj x.*

RJEŠENJE

```
a=int(input())
b=int(input())
c=int(input())
print('Rješenje jednačine je',-c*b/a)
```

Ime i broj godina

Promjenljivoj *x* pridruži svoje ime, a promjenljivoj *g* broj svojih godina te ispiši poruku: "Zovem se *x* i imam *g* godina." .
Napomena: ime i broj godina unose se sa tastature.

RJEŠENJE

```
x=input()
y=input()
print("Zdravo! Zovem se", x, "i imam", y, "godina.")
```

I1 IZAZOVI (INPUT, OUTPUT, PROMJENLJIVE, TIPOVI PODATAKA)

Zamjena mjesta cifara dvocifrenog broja

Dat je dvocifreni prirodan broj. Napisati program koji će zamijeniti mjesta ciframa tog broja i ispisati dobijeni broj.

	PRIMJER 1	PRIMJER 2
Ulaz	76	29
Izlaz	67	92

RJEŠENJE

```
n=int(input())
print((n%10)*10+(n//10))
```

Zbir cifara četvorocifrenog broja

Dat je četvorocifreni prirodan broj. Napisati kod koji štampa zbir cifara tog broja.

	PRIMJER 1	PRIMJER 2
Ulaz	1234	6453
Izlaz	10	18

RJEŠENJE

```
n=int(input())
s=(n%10)+((n%100)//10)+((n%1000)//100)+(n//1000)
print(s)
```

Broj u ogledalu (M)

Dat je četvorocifreni prirodan broj. Napisati kod koji dati broj štampa unazad. Npr. ako je dat broj 1234 izlaz treba da bude 4321.

	PRIMJER 1	PRIMJER 2	PRIMJER 3
Ulaz	1234	6475	4580
Izlaz	4321	5746	854

RJEŠENJE

```
n=int(input())
a=n%10
b=(n%100)//10
c=(n%1000)//100
d=n//1000
s=a*1000+b*100+c*10+d
print(s)
```

Zbir dvocifrenih brojeva dobijenih od četvorocifrenog broja (M)

Četvorocifreni prirodan broj m podijelimo na dva dvocifrena broja: prvi broj čine cifra hiljada i cifra desetica (tim redom), a drugi broj čine cifra jedinica i cifra stotina (tim redom). Napisati program koji učitava cio broj m i štampa zbir dobijenih dvocifrenih brojeva. Npr. ako je m=1532, tada se sabiraju brojevi 13 i 25 tj. rezultat je 38.

	PRIMJER 1	PRIMJER 2
Ulaz	1532	7586
Izlaz	38	143

RJEŠENJE

```
n=int(input())
a=(n//1000)*10+((n%100)//10)
b=(n%10)*10+((n%1000)//100)
print(a+b)
```

N2

GRANANJE

Ispiši manji broj

Napiši program koji unosi dva broja i ispisuje poruku koji je od dva učitana broja manji.

RJEŠENJE

```
a=int(input())
b=int(input())
if a<b:
  print(a)
else:
  print(b)
```

Paran ili neparan

Napiši program koji za broj unijet sa tastature ispisuje “PARAN” ukoliko je on paran i “NEPARAN” ukoliko je neparan.

RJEŠENJE

```
a=int(input())
if a%2==0:
  print(“PARAN”)
else:
   print(“NEPARAN”)
```

Kvadrat ili pravougaonik

Napiši program koji unosi dužine stranice a i b i provjerava je li zadata figura kvadrat ili pravougaonik.

RJEŠENJE

```
a=int(input())
b=int(input())
```

```
if a==b:
  print("Kvadrat!")
else:
  print("Pravougaonik!")
```

Vaterpolo

Napiši program koji unosi broj golova ekipa na vaterpolo utakmici Mađarska-Crna Gora te ispisuje tekst 'Mađarska je pobjednik!' ukoliko je Mađarska pobijedila, tekst 'Crna Gora je pobjednik!' ukoliko je Crna Gora pobijedila ili 'Neriješeno!' u slučaju da su obije ekipe dale jednak broj golova.

RJEŠENJE

```
a = int(input())
b = int(input())
if a>b:
  print("Mađarska je pobjednik!")
elif a<b:
  print("Crna Gora je pobjednik!")
else:
  print("Neriješeno!")
```

Djeljiv sa 5?

Napiši program koji za broj n, unijet sa tastature, ispisuje da li je n djeljiv sa 5 ili ne. Napomena: koristi operator za računanje ostatka pri cijelobrojnom dijeljenju (%)

RJEŠENJE

```
n=int(input())
if n%5==0:
  print("Broj je djeljiv sa 5.")
else:
  print("Broj nije djeljiv sa 5.")
```

Naredni neparan broj

Napiši program koji učitava broj n i ispisuje prvi sljedeći, njemu najbliži, neparan broj (najmanji neparan broj koji je strogo veći od n).

RJEŠENJE

```
a = int(input())

if a%2==0:
  print(a+1)
else:
  print(a+2)
```

Plan čitanja knjige

Tamara je dobila novu knjigu od bake koja ima s stranica. Odlučila je svaki dan pročitati maksimalno x stranica. Napiši program koji će ispisati poruku `DA' ili `NE' u zavisnosti od toga može li Tamara pročitati svoju knjigu za nedjelju dana (7 dana). Ukoliko je odgovor `NE' u narednom redu ispisati koliko bi joj još stranica ostalo za čitanje za narednu nedjelju.

RJEŠENJE

```
s=int(input())
x=int(input())
n=s/x
if n<=7:
  print("DA")
else:
  print("NE", s-7*x, sep='\n')
```

Ko je pobijedio?

Ana i Todor su učestvovali u kvizu znanja. Svako od njih odgovarao je na tri pitanja i za svako pitanje osvojili su 1, 2 ili 3 poena. Napiši program koji će izračunati koliko poena su ukupno osvojili i ispisuje ko je pobijedio u kvizu i koliko poena je osvojio/

osvojila(svaki podatak u zasebnom redu). Ukoliko su imali jednak broj poena, ispisati "Neriješeno." i broj osvojenih bodova.

Osvojene bodove unesi pomoću tastature. U prva tri reda unose se brojevi poena koje je osvojila Ana, a u naredna tri reda brojevi poena koje je osvojio Todor.

RJEŠENJE

```
A1=int(input())
A2=int(input())
A3=int(input())
A=(A1+A2+A3)
T1=int(input())
T2=int(input())
T3=int(input())
T=(T1+T2+T3)
if A>T:
  print(" Pobijedila je Ana.")
elif T>A:
  print("Pobijedio je Todor.")

else:
  print("Neriješeno.")
```

Kakav je trougao?

Napiši program koji prvo provjerava da li postoji trougao sa stranicama dužina a, b i c unijetim sa tastature. Ukoliko takav trougao postoji program ispisuje da li je on jednakostraničan, jednakokraki ili nejednakostraničan. U suprotnom ispisati tekst: "Ne postoji trougao sa unijetim dužinama stranica!".

RJEŠENJE

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

if a==b and b==c:
  print("Trougao je jednakostraničan.")
elif a==b or a==c or b==c:
  print("Trougao je jednakokraki.")
else:
  print("Trougao je nejednakostraničan."
```

Farbanje ograde

Đed Ivo je odlučio ofarbati svoju ogradu. Farba je tako da svaki stubić ograde boji redom u tri boje: zelena, crvena, plava, zelena, crvena, plava, itd. Napiši program koji unosi redni broj stubića i ispisuje boju kojom on treba biti ofarban.

Npr.
1 - zelena,
4 - zelena,
6- plava
itd.
Napomena: Pokušaj koristiti operaciju **mod (%).**

RJEŠENJE

```
a=int(input())
if a%3==0:
  print("plava")
elif a%3==1:
  print("zelena")
elif a%3==2:
   print("crvena")
```

N2 DOMAĆI

Broj poena u igrici veći od 100?

Broj poena u igrici veći od 100? Napiši program u kojem se unosi broj osvojenih poena u igrici, a koji ispisuje da li je taj broj manji od, veći od ili jednak 100:

- manji
- veći ili
- jednak.

Na primjer:
ulaz:
90
izlaz:
manji

RJEŠENJE

```
a=int(input())
if a==100:
  print("Jednak")
elif a<100:
  print("Manji")
else:
  print("Veci")
```

Jednaki ili različiti?

Napiši program koji za unijeta dva broja provjerava da li su jednaki ili različiti i ispisuje odgovarajuću poruku:

- jednaki ili
- različiti.

Na primjer:
ulaz:
2
3
izlaz:
različiti

RJEŠENJE

```
a=int(input())
b=int(input())
if a==b:
  print("jednaki")
else:
  print("različiti")
```

Pozitivan, negativan ili nula?

Koristeći if, elif, else konstrukciju, napiši program koji će za unešeni broj ispisati da li je on pozitivan, negativan ili jednak nuli:

- pozitivan
- negativan
- nula.

U programu omogući unos decimalnog broja koristeći **float**: a = float(input())

RJEŠENJE

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

Imaš, nemaš, taman!

Želiš da kupiš dvije stvari. Napiši program u kojem se unose cijene tih proizvoda (tip vrijednosti:float) i koji provjerava da li imaš dovoljno novca da kupiš obije stvari. Kod sebe imaš 20 eura. U zavisnosti od rezultata provjere, program treba ispisati:

- imaš
- nemaš ili
- taman (ukoliko zajedno koštaju tačno 20 eur).

RJEŠENJE

```
a=float(input())
b=float(input())
c=a+b
if c<20:
        print("imaš")
elif c>20:
        print("nemaš")
else:
        print("taman")
```

Ocjenjivanje po bodovnoj skali

Napiši program koji korisniku omogućava unos broja bodova od 1 do 100, a zatim na osnovu bodovne skale

<=50 Nedovoljan
<=63 Dovoljan
<=76 Dobar
<=89 Vrlo dobar
<=100 Odličan

ispisuje odgovarajuću ocjenu.

RJEŠENJE

```
a = int(input())
if a<=50:
  print("Nedovoljan")
elif a<=63:
  print("Dovoljan")
elif a<=76:
  print("Dobar")
elif a<=89:
  print("Vrlo dobar")
else:
  print("Odlican")
```

Ostaci pri dijeljenju i boje

Napišite program koji korisniku omogućava unos cijelog broja i ukoliko je ostatak dijeljenja tog broja sa brojem 3:
- jednak nuli, ispisuje „plava“,
- jednak 1, ispisuje „crvena“,
- jednak 2, ispisuje „zelena“,

Uputstvo: Koristiti **elif** i **mod**.
Mod (u oznaci %) nam daje ostatak pri dijeljenju.
15%3 je jednak 0. (15 pri cijelobrojnom dijeljenju sa 3 daje ostatak 0)

Podsjetimo se:

18=6*3+0, 18%6=0 -> „plava“
19=6*3+1, 19%3=1 -> „crvena“
20=6*3+2, 20%3=2 -> „zelena“
21=7*3+0, 21%3=0 -> „plava“
22=7*3+1, 22%3=1 -> „crvena“
23=7*3+2, 23%3=2 -> „zelena“
24=8*3+0, 24%3=2 -> „plava“
... ...

RJEŠENJE

```
a=int(input())
if a%3==0:
  print("plava")
elif a%3==1:
  print("crvena")
elif a%3==2:
  print("zelena")
```

I2 IZAZOVI (IF)

Da li su brojevi istog znaka?

Napisati program kojim se provjerava da li su dva cijela broja istog znaka. Za potrebe ovog zadatka, nulu ćemo posmatrati kao pozitivan broj.

RJEŠENJE

```
a=int(input())
b=int(input())
if a>0 and b>0:
  print("da")
elif a<0 and b<0:
  print("da")
else:
  print("nisu")
```

	PRIMJERI	
Ulaz	14	134
	-144	12
Izlaz	NE	DA

RJEŠENJE

```
a = int(input())
b = int(input())

if (a>=0 and b>=0) or (a<0 and b<0):
  print("DA")
else:
  print("NE")
```

Preskoči 13 i brojeve desno od njega

Za date vrijednosti **a, b** i **c** izračunaj njihov zbir. Ipak, ako je neka od unijetih vrijednosti 13 tada traženi zbir ne treba sadržati tu vrijednost, kao ni vrijednosti promjenljivih koje su "desno od" nje (**b** je "desno od" **a, b** i **c** su "desno od" **a**). Zadatak riješi korišćenjem **if** pitalice!

Na primjer:
a, b, c tako da je: a=1, b=13, c = 3 .... tada zbir treba da bude 1.
(ne računa se b i broj desno od njega, dakle ne računa se ni c)

Ulaz		Izlaz
a, b, c	→	Zbir
1, 2, 3	→	6
1, 2, 13	→	3
1, 13, 3	→	1

RJEŠENJE

```
a= int(input())
b= int(input())
c= int(input())
if a==13:
  print(0)
elif b==13:
  print(a)
elif c==13:
  print(a+b)
else:
  print(a+b+c)
```

Prestupna godina (M)

Napisati program koji učitava prirodan broj g i štampa poruku "GODINA ____ NIJE/JE PRESTUPNA".

	PRIMJER 1	PRIMJER 2
Ulaz	2020	2018
Izlaz	GODINA 2020 JE PRESTUPNA	GODINA 2018 NIJE PRESTUPNA

RJEŠENJE

```
g=int(input())
if(g%400==0 or ((g%4==0) and (g%100!=0))):
  print("GODINA", g, "JE PRESTUPNA")
else:
  print("GODINA", g, "NIJE PRESTUPNA")
```

N3

"FOR" PETLJA

Jedan, dva tri,...deset

Napiši program koji pomoću for petlje ispisuje brojeve od 1 do 10, jedan pored drugog sa razmakom(blankom) između brojeva.

Izlaz
1 2 3 4 5 6 7 8 9 10

RJEŠENJE

```
for i in range (1,11):
  print(i, end=" ")
```

Samoglasnici u riječi

Napiši program koji za unijetu riječ ispisuje sve samoglasnike koji se nalaze u toj riječi (čitajući riječ sa lijeva na desno). Svaki samoglasnik ispisati u novom redu.

	PRIMJER
Ulaz	Informatika
Izlaz	I o a i a

RJEŠENJE

```
riječ=input()
for i in rijec:
   if i in 'aeiouAEIOU':
        print(i)
   #if i.lower() in 'aeiou'
```

Parni brojevi iz intervala [240,260]

Napiši program koji pomoću petlje for ispisuje sve parne brojeve od 240 do 260 jedan pored drugoga.

RJEŠENJE

```
#1. primjer rješenja:
for i in range(240,261):
  if i%2==0:
    print(i, end = ' ')


#2. primjer rješenja:
'''for i in range(240,261,2):
    print(i, end=' ') '''
```

Djeljivi sa 5 iz intervala [100, 150]

Napiši program koji pomoću petlje for ispisuje, jedan ispod drugoga, brojeve iz intervala [100, 150] koji su djeljivi sa 5.

RJEŠENJE

```
for i in range(100,151):
  if i%5==0:
    print(i)
```

"Speluj" mi riječ "sunce"

Koristeći for petlji, napiši program koji ispisuje slovo po slovo riječi "sunce", jedno ispod drugoga.

RJEŠENJE

```
for slovo in "sunce":
  print(slovo)

'''
a='sunce'
for i in a:
  print(i)
'''
```

Prebroj pojavljivanja slova u riječi

Napiši program koji za unijetu riječ provjerava koliko puta se slovo **a** (veliko ili malo) nalazi u njoj i ispisuje dobijeni broj pojavljivanja.

RJEŠENJE

```
n = input()
p = 0
for i in n:
  if i in 'aA':
    p = p + 1
print(p)
```

Ispiši u opadajućem redosljedu

Napiši program koji prvih šest prirodnih brojeva ispisuje jedan ispod drugog u opadajućem redosljedu (od najvećeg do najmanjeg).

RJEŠENJE

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

Svi manji od mene

Napiši program koji za unijeti prirodni broj **n** ispisuje sve prirodne brojeve koji se nalaze u intervalu [1, n] i to u rastućem redosljedu, jedan pored drugog, razdvojene blankom.

RJEŠENJE

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

Pravougaonik ispunjen zvjezdicama

Koristeći dvije ugnježdene for petlje (petlja u petlji) napisati program koji ispisuje zvijezdice na sljedeći način:

```
*****
*****
*****
*****
```

RJEŠENJE

```
#1. primjer korišćenjem dvije for petlje:
for i in range(4):
  for j in range(5):
    print ('*', end='')
  print ('')

#2. primjer korišćenjem dvije for petlje:
for i in range(1,5):
  for j in range (1, 6):
    print('*', end='')
  print('')

#koristeći jednu for petlju :) , što nije u skladu sa
#postavkom zadatka
for i in range(1,5):
  print('*****')
```

Trougao ispunjen zvjezdicama

Koristeći dvije ugnježdene for petlje (petlja u petlji) napisati program koji ispisuje zvijezdice na sljedeći način:

```
*
**
***
****
*****
```

RJEŠENJE

```
for i in range(1,6):
  for j in range(i):
    print('*', end='')
  print('')
```

Trougao ispunjen brojevima

Koristeći dvije ugnježdene for petlje (petlja u petlji) napisati program koji ispisuje brojeve na sljedeći način:

0
11
222
3333
44444
555555

RJEŠENJE

```
for i in range (1,7):
  for j in range (0,i):
    print (i-1,end=" ")
  print ("")

#i=1, j=0 ,          print 0
#i=2, j=0,1 ,        print 11
#i=3, j=0,1,2        print 222
#...
#i=6, j=0,1,2,3,4,5  print 555555
```

Zbir n prirodnih brojeva

Napisati program koji za unijet prirodan broj n, računa i ispisuje zbir prvih n uzastopnih prirodnih brojeva: 1+2+3+...+n.

RJEŠENJE

```
#1. način:
n = int(input())
p = 0
for i in range(1,n+1):
  p = p + i
print(p)

#2. nacin:
print("Unesi prirodan broj: ")
n=int(input())
z=0
for i in range (1,n+1,1):
  z=z+i
  print (i,z)
print ("Zbir prvih n uzastopnih prirodnih brojeva je", z)
#print ("Zbir prvih n uzastopnih prirodnih brojeva
#je", (n*(n+1))/2)"
```

Za koliko brojeva želiš provjeriti djeljivost?

Napiši program koji prvo učitava prirodan broj **b**, a zatim, omogućava korisniku da **b** puta unese novi prirodan broj **n**. Nakon svakog unosa broja **n**, program provjerava da li je unijeto **n** djeljivo sa 3 i ispisuje odgovarajuću poruku:

- Broj <n> je djeljiv sa 3 ili
- Broj <n> nije djeljiv sa 3.

Primjer:

```
Koliko brojeva želiš da ispitaš: 4
 25
Broj 25 nije djeljiv sa 3
 26
Broj 26 nije djeljiv sa 3
 27
Broj 27 je djeljiv sa 3
 253
Broj 253 nije djeljiv sa 3
```

RJEŠENJE

```
b=int(input('Koliko brojeva želiš da ispitaš:'))
for i in range(b):
  n=int(input())
  if n%3==0:
    print("Broj", n, "je djeljiv sa 3")
  else:
    print("Broj", n, "nije djeljiv sa 3")
```

N3 DOMAĆI

Štampa neparnih brojeva iz intervala

Napisati program koji ispisuje sve neparne brojeve iz intervala [100, 150], jedan pored drugog.

RJEŠENJE

```
for i in range(100,151):
        if i%2!=0:
                print(i,end=' ')
```

Prebroj suglasnike

Koristeći for petlju, napisati program koji za unijetu riječ ispisuje broj suglasnika koje ona sadrži (izbrojati i ponovljena pojavljivanja istog suglasnika).

	PRIMJER
Ulaz	Programiranje
Izlaz	8

Kratko podsjećanje u vezi sa ovim zadatkom:

Ukoliko u programu želimo da neka promjenljiva uzima vrijednosti svih karaktera od koji se sastoji neki string, možemo iskoristiti for petlju poput sljedeće:

for i in 'Tinker'

za koju će promjenljiva i u for petlji imati vrijednosti redom:

1. 'T'
2. 'i'
3. 'n'
4. 'k'
5. 'e'
6. 'r'

odnosno, naredbe unutar for petlje će se izvršiti 6 puta, i pri tom će svaki put promjenljiva i dobiti novu vrijednost.

Ukoliko želimo da provjerimo da li je neka promjenljiva čuva vrijednost(karakter) koja je samoglasnik, možemo napisati pitalicu:

```
if i in 'aeiouAEIOU':
```

Na sličan način, ukoliko želimo da provjerimo da li neka promjenljiva čuva vrijednost (karakter) koja je samoglasnik, možemo napisati pitalicu:

```
if i not in 'aeiouAEIOU':
```

Takođe, treba obratiti pažnju: karakter 'a' razlikuje se od karaktera 'A'!

RJEŠENJE

```
a=input()
p=0
for i in a:
  if i in "aeiouAEIOU":
    p=p+1
print(p)
```

Svi koji pri dijeljenju sa 3 daju ostatak 1

Napiši program koji za unijeti prirodan ispisuje jedan ispod drugog sve brojeve od 1 do tog broja koji pri dijeljenju sa 3 daju ostatak 1.

RJEŠENJE

```
n=int(input())
for i in range(1,n+1):
  if i%3==1:
    print(i)
```

Obrnuti trougao sa ciframa

Koristeći for petlju, napisati program koji prikazuje sljedeći izlaz:

7 7 7 7 7 7 7
6 6 6 6 6 6
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1

RJEŠENJE

```
#1.način:
      for i in range(7,0,-1):
            for j in range(1,i+1):
                  print(i, end=' ')
            print(' ')
#2. način:
for i in range (7,0,-1):
  for j in range(i,0,-1):
    print(i,end=' ')
  print(' ')
```

Saberi brojeve od 1 do n i natrag

Koristeći naredbu ponavljanja napisati program koji za unijeti prirodan broj **n** računa vrijednost sljedećeg izraza 1 + 2 + 3 +... n + ... + 3 + 2 + 1 .

Na primjer, ako unesemo broj n = 5, dati izraz izgleda ovako: 1+2+3+4+5+4+3+2+1

RJEŠENJE

```
n=int(input())
p=0
for i in range (1,n+1):
  p=p+i
for j in range(n-1,0,-1):
  p=p+j
print(p)
```

Kvadrati 3 broja

Koristeći for petlju, napiši program koji omogućava da se unese neki broj, a zatim štampa vrijednost kvadrata toga broja, i to ponoviti tri puta. Na kraju programa treba jednom odštampati poruku:"THE END".

RJEŠENJE

```
for i in range(3):
        broj = int(input())
        print ('Kvadrat vašeg broja je',
broj*broj)
print('THE END')

#primjer rješenja koje nije u skladu sa zahtjevom
#zadatka(tražena je primjena for petlje)
a=int (input ())
print ("Kvadrat vašeg broja je", a**2)
b=int (input ())
print ("Kvadrat vašeg broja je", b**2)
c=int (input ())
print ("Kvadrat vašeg broja je", c**2)
print ("THE END")
```

Koliko puta ispisuje Hello?

Napisati program (koristeći for petlju) koji ispisuje sljedeće:

1 -- Hello
2 -- Hello
3 -- Hello
4 – Hello

RJEŠENJE

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

Čudna riječ

Napišite program koji koristi četiri for petlje štampa dolje prikazani niz slova:
AAAAAAAAAABBBBBBBCDCDCDCDEFFFFFFG.

RJEŠENJE

```
for i in range(1,11):
  print("A",end='')
for i in range(1,8):
  print("B",end='')
for i in range(1,5):
  print("CD",end='')
print("E",end='')
for i  in range(1,7):
  print("F",end='')
print("G")

#bez for petlje, rješenje bi moglo izgledati ovako:
print(10*"A",7*"B",4*"CD","E",6*"F","G", sep="")
```

I3 IZAZOVI (IF, FOR)

Izračunaj i ispiši vrijednost izraza

Napiši program koji će korisniku omogućiti unos prirodnog broja n i realnog broja x (za n unijeti vrijednost tipa int, a za x vrijednost tipa float). Program, zatim, za unijeto n i x treba izračunati i ispisati vrijednost sljedećeg izraza:

$1 - x + x^2 - x^3 + x^4 \ldots \pm x^n.$

	PRIMJER
Ulaz	10 2.0
Izlaz	683.0

Pomoć:
Operacija stepenovanja u Pajtonu je: **, tako da bi gore navedeni izraz, u Pythonu, za n = 6 mogli zapisati na sljedeći način:

1 - x + x**2 - x**3 + x**4 - x**5 + x**6 .

RJEŠENJE

```
n = int(input())
x = float(input())
p=0
q=0
for i in range(0,n+1):
  if i%2==0:
    p = p + x**i
  else:
    q = q + x**i
print(p-q)
```

Ram od zvjezdica

Napisati program koji, koristeći dvije „for“ petlje, ispisuje zvjezdice na sljedeći način:

```
* * * * * * * * *
*               *
*               *
*               *
*               *
*               *
*               *
*               *
* * * * * * * * *
```

Napomena: Voditi računa da kod ne sadrži ovakvu ili sličnu liniju koda: print('* *') !

RJEŠENJE

```
#za svaki red:
for i in range(1,9):
  print(' ')
  #za svaku kolonu
  for j in range(1,9):
    #ukoliko crtamo gornju i donju ivicu rama:
    if i==1 or i==8:
       print('*', end= ' ')
    else:
      #ukoliko crtamo lijevu li desnu ivicu rama:
      if j==1 or j==8:
        print('*', end=' ')
      else:
        print(' ', end=' ')
```

Strelica od zvjezdica

Koristeći „for“ petlju napisati program koji ispisuje zvjezdice na način prikazan na slici:

```
*
***
*****
*******
*****
***
*
```

RJEŠENJE

```
for i in range(1,8,2):
  for j in range(1,i+1):
    print('*',end='')
  print('')
for i in range(5,0,-2):
  for j in range(1,i+1):
    print('*',end='')
  print('')
```

Proizvod prvih n prirodnih brojeva

Napisati program koji za unijet prirodan broj *n* računa proizvod prvih *n* prirodnih brojeva.

RJEŠENJE

```
n=int(input())
p=1
for i in range(1,n+1):
  p=p*i
  i=i+1
print(p)
```

N4

"WHILE" PETLJA

Sa while petljom od 1 do 10

Koristeći "while" naredbu ponavljanja napisati program koji ispisuje brojeve od 1 do 10, jedan ispod drugog.

RJEŠENJE

```
i=1
while i<11:
  print(i)
  i=i+1
```

Parni brojevi razdvojeni zarezom

Koristeći „while" petlju ispisati jedan do drugog, u rastućem redosljedu, sve parne brojeve iz intervala [1, 50] tako da međusobno budu razdvojeni zarezom.

OUTPUT
2,4,6,8,10,12,...46,48,50

RJEŠENJE

```
i=2
while i<=50:
        if i < 50:
                print(i, end=",")
        else:
                print(i)
                i=i+2
```

U opadajućem redosljedu [500,520]

Koristeći while petlju, ispisati u opadajućem redosljedu, sve brojeve od 500 do 520, jedan ispod drugog.

RJEŠENJE

```
i=520
while 500<=i<=520:
  print(i)
  i=i-1
```

Preskoči broj

Koristeći „while" petlju napiši program koji ispisuje sve brojeve od 1 do 100, isključujući broj 55.

RJEŠENJE

```
i=0
while i<=100:
  if i==55:
    continue
  i=1+i
  print(i, end=")
```

Pogodi broj (osnova)

Koristeći while petlju, napraviti jednostavnu igru u kojoj računar na slučajan način generiše zamišljen broj iz intervala [1,10] na slučajan način, a korisnik pogađa zamišljeni broj sve dok ga ne pogodi.

RJEŠENJE

```
from random import randint
tajni_broj=randint(1,10)

pokusaj=int(input())

while tajni_broj != pokusaj:
  print ("Ne, nije.")
  pokusaj=int(input("Unesi neki drugi broj:"))
else:
  print ("Bravo! Pogodak!")
```

N4 DOMAĆI

Nisu djeljivi sa 7

Napisati program koji, koristeći while petlju, ispisuje jedan pored drugog sve prirodne brojeve od 23 do 77 koji nisu djeljivi sa 7.

RJEŠENJE

```
i=23
#while i>=23 and i <=77:
while i <=77:
  if i%7!=0:
    print(i, end=' ')
  i=i+1
```

Koliko brojeva je djeljivo sa 8?

Napiši program koji, koristeći while petlju, za unijeti broj n ispisuje koliko ukupno ima brojeva u intervalu [1,n] koji su djeljivi sa 8.

RJEŠENJE

```
n = int(input())
i=1
br=0
while i<n:
  if i%8==0:
    br=br+1
  i=i+1
```

Trougao od zvjezdica i upitnika

Napisati program koji štampa zvjezdice i upitnike na sljedeći način:

izlaz:

```
*
??
***
????
```

RJEŠENJE

```
#rješenje korišćenjem whil petlje
i=1
while i<6:
  j=1
  while j<i+1:
    if i%2!=0:
      print('*',end='')
    else:
      print('?',end='')
    j=j+1
  print('')
  i=i+1

#rješenje sa for petljom bi moglo izgledati ovako:
for i in range(1,6):
  print('')
  for j in range(1,i+1):
    if i%2!=0:
      print('*',end='')
    else:
      print('?',end='')
```

Razlika parnih i neparnih

Napiši program koji za unijeto n (n je prirodan broj, n >5) računa dva zbira:

- zbir svih parnih,
- zbir svih neparnih brojeva,

iz intervala [5, n], i ispisuje ekranu te zbirove, a zatim i razliku zbira svih parnih i zbira svih neparnih brojeva iz intervala.

Primjer:

n =10
zbir_parnih_brojeva: 6+8+10 = 24
zbir_neparnih_brojeva: 5+7+9 = 21
Razlika zbira parnih i zbira neparnih brojeva je:
24 - 21 = 3

	PRIMJER
Ulaz	10
Izlaz	24 21 3

RJEŠENJE

```
n = int(input())
i = 5
parni=0
neparni=0

while i<=n:
  if i%2==0:
    #i je paran
    parni=parni+i
  else:
    #i je neparan
    neparni=neparni+i
  i=i+1

print(parni,neparni)
print(parni-neparni)
print(parni+neparni)
```

I4 IZAZOVI (IF, FOR, WHILE)

Zbir cifara, korišćenjem while petlje

Koristeći while petlju, napisati program koji za unijeti broj n (n>0) ispisuje zbir njegovih cifara.

	PRIMJER 1	PRIMJER 2
Ulaz	123	5403
Izlaz	6	12

Pomoć:
Koristiti operacije cijelobrojno dijeljenje (//) i ostatak pri cijelobrojnom dijeljenju (%) .

RJEŠENJE

```
n = int(input())
zbir=0
while n>0:
  cifra = n%10
  zbir = zbir + cifra
  n = n//10
  #prethodnu liniju koda možemo napisati i ovako:
  #n = int(n/10)
print(zbir)
```

Cijena n proizvoda

Proizvod košta a eura i b centi (a i b nenegativni cijeli brojevi, b < 100). Koliko košta n takvih proizvoda? Napisati program koji učitava brojeve a, b i n i štampati dvije vrijednosti: koliko eura i koliko centi treba platiti za n proizvoda.

	PRIMJER
Ulaz	Eura: 2 Centi: 30 Unesi broj proizvoda: 4
Izlaz	9 20

RJEŠENJE

```
a = int(input("Eura:"))
while(a<0):
  a = int(input("Neispravno. Uneti novi broj:"))

b = int(input("Centi:"))
while(b<0 or b>=100):
  b = int(input("Neispravno. Uneti novi broj:"))

n = int(input("Unesi broj proizvoda:"))
while(n<0):
  n = int(input("Neispravno. Uneti novi broj:"))

ukupno_centi = n*(a*100+b)
print(ukupno_centi//100, ukupno_centi%100)
```

IGRA - Pogodi broj

Isprogramiraj igricu Pogodi broj poštujući sljedeća pravila:

- Program generiše zamišljeni broj na slučajan način.
- Igrač ima samo pet pokušaja da pogodi broj (od 1 do 100).
- Poslije svakog pokušaja program saopštava da li je unijeti broj manji ili veći od traženog broja.
- Program štampa odgovarajuću poruku kada igrač pobijedi ili izgubi igru.

Ovdje je prikazan primjer kako treba da izgleda tok igre:

```
Pogodi broj [1-100]:  50
Zamišljeni broj je VEĆI. Imaš još 4 pokušaja

Pogodi broj [1-100]:  75
Zamišljeni broj je MANJI. Imaš još 3 pokušaja

Pogodi broj [1-100]:  60
Zamišljeni broj je VEĆI. Imaš još 2 pokušaja

Pogodi broj [1-100]:  68
Zamišljeni broj je VEĆI. Imaš još 1 pokušaja

Pogodi broj [1-100]:  70
Da, to je taj broj!
```

RJEŠENJE

```
from random import randint

tajni_broj = randint(1,100)

for brojac_pokusaja in range(1,6):
  pokusaj = int(input('Pogodi broj [1-100]: '))
  if pokusaj < tajni_broj:
        print("Zamišljeni broj je VEĆI.","Imaš još ",
5-brojac_pokusaja,'pokusaja\n')
  elif pokusaj > tajni_broj:
      print("Zamišljeni broj je MANJI.", "Imaš još ",
5-brojac_pokusaja,'pokusaja\n')
  else:
    print('Da, to je taj broj!')
    break
else:
    print('Kraj igre. Zamišljen broj je ', tajni_broj, '.')
```

Prosječan broj bodova na testiranju(M)

Napisati program koji učitava broj bodova studenata osvojenih na testiranju (cijeli brojevi iz intervala[0, 100]) i štampa **broj studenata i prosječan broj bodova na testiranju.** Učitavanje se prekida kada se unese broj bodova koji ne pripada intervalu [0, 100].

	PRIMJER 1	PRIMJER 2	PRIMJER 3
Ulaz	12 23 123	123	10 90 80 75 -1
Izlaz	2 17.5	0 0	4 63.75

RJEŠENJE

```
n = int(input())
z = 0
b = 0
while(n<=100 and n>=0):
  z = z+n
  b = b + 1
  n = int(input())
print(b)
if b==0:
  print(0)
else:
  print(z/b)
```

Cijena n proizvoda(M)

Proizvod košta a eura i b centi (a i b nenegativni cijeli brojevi, b < 100). Koliko košta n takvih proizvoda? Napisati program koji učitava brojeve a, b i n i štampa dvije vrijednosti: koliko eura i koliko centi treba platiti za n proizvoda.

	PRIMJER
Ulaz	2 30 4
Izlaz	9 20

RJEŠENJE

```
a = int(input("Eura:"))
while(a<0):
  a = int(input("Neispravno. Uneti novi broj:"))

b = int(input("Centi:"))
while(b<0 or b>=100):
  b = int(input("Neispravno. Uneti novi broj:"))

n = int(input("Unesi broj proizvoda:"))
while(n<0):
  n = int(input("Neispravno. Uneti novi broj:"))

ukupno_centi = n*(a*100+b)
print(ukupno_centi//100, ukupno_centi%100)
```

N5

STRINGOVI (NISKE)

Na kojim pozicijama se nalazi slovo

Napiši program koji za unijeti string štampa jedan ispod drugog sve brojeve pozicije na kojima se slovo 'a' (malo slovo 'a') nalazi u tom stringu.

Napomena:
Pozicija prvog slova u riječi, a lijeve strane riječi je 0, drugog 1, itd.

	PRIMJER
Ulaz	Matematika
Izlaz	1 5 9

RJEŠENJE

```
s = input()
for i in range(len(s)):
  if s[i] == 'a':
    print (i)
```

Vidim duplo

Napiši program koji na osnovu unijetog stringa, formira i ispisuje novi string u kojem je duplirano svako slovo iz unijetog stringa.

	PRIMJER
Ulaz	Hello
Izlaz	HHeellllloo

RJEŠENJE

```
s = input()
for i in range(len(s)):
   print(s[i]* 2, end = '')
```

Igra sa slovima

Napiši program koji za unijetu riječ štampa tu riječ na način opisan u primjeru:

	PRIMJER
Ulaz	Tinker
Izlaz	T Ti Tin Tink Tinke Tinker

RJEŠENJE

```
s = input()
for i in range(len(s)):
  print(s[0:i+1])
```

Sredi tekst

Napiši program koji u unijetom tekstu sva velika slova zamjenjuje malim slovima, uklanja sljedeće karaktere: ,.;:-?!\() i na kraju štampa dobijeni tekst.

	PRIMJER
Ulaz	(Ti:nk;)2\0,Er.2-0!
Izlaz	tink20er20

RJEŠENJE

```
s = input()
s = s.lower()
p = ''
for i in range(len(s)):
  if s[i] not in ",.;:-?!\()":
    p = p + s[i]
print(p)
```

Iza decimalne tačke

Napiši program koji za unijeti string koji predstavlja broj sa decimalnom tačkom, štampa cifre toga broja desno od decimalne tačke.

	PRIMJER
Ulaz	3.14159
Izlaz	14159

RJEŠENJE

```
s = input()
for i in range(len(s)):
  if s[i] == '.':
    print(s[i+1:])
```

Prebroj riječi

Jedan jednostavan, nesavršen, način da se procijeni broj riječi u tekstu je da se prebroji koliko taj tekst ima blankova („ "). Napiši program koji za unijeti tekst, ispisuje procijenjeni broj riječi u tekstu prebrojavanjem blankova.

	PRIMJER
Ulaz	„Why Software Is Eating the World by Marc Andreessen"
Izlaz	9

RJEŠENJE

```
s = input()
#obrati pažnju: krećemo od 1!
p = 1
for i in range(len(s)):
  if s[i] == ' ':
    p = p + 1
print (p)
```

Provjeri zagrade

U aritmetičkim izrazima, u kojima koristimo zagrade za grupisanje, ponekad se dogodi da nenamjerno ostavimo poneku neotvorenu ili nezatvorenu zagradu. Napiši program koji za unijeti aritmetički izraz provjerava da li ima jednak broj lijevih i desnih zagrada i štampa odgovarajuću poruku: „Jednak je broj zagrada“ ili „Nije jednak broj zagrada“.

	PRIMJER 1	PRIMJER 2
Ulaz	(x-3)*2+21+(-11+8)	(x-3)*2+21+(-11+8)+1)
Izlaz	Jednak je broj zagrada	Nije jednak broj zagrada

RJEŠENJE

```
s = input()
#na početku, broj otvorenih i broj zatvorenih
#zagrada postavljamo na 0
br_o = 0
br_z = 0
for i in range(len(s)):
  if s[i] == '(':
    br_o = br_o + 1
  if s[i] == ')':
    br_z = br_z + 1
if br_o == br_z:
  print('Jednak je broj zagrada')
else:
  print('Nije jednak broj zagrada')
```

Korekcija teksta: mala slova, ukloni tačke i zareze

Napiši program koji unijeti tekst koriguje na sljedeći način:
- konvertuje sva velika u mala slova i
- uklanja sve tačke i zareze,

a zatim štampa izmijenjeni tekst.

	PRIMJER
Ulaz	Programira,.3nje!?
Izlaz	programira3nje!?

RJEŠENJE

```
s = input()
s = s.lower()
p = ''
for i in range(len(s)):
  if s[i] not in ",.":
    p = p + s[i]
print(p)
```

N5 DOMAĆI

Korekcija teksta: drugi karakter zvjezdica, na kraju teksta dodaj uzvičnike!

Napiši program koji unijeti tekst koriguje na sljedeći način:
- karakter na drugom mjestu sa lijeva mijenja sa zvjezdicom: „*“
- na kraju teksta obavezno doda tri znaka uzvika „!!!“.

a zatim štampa izmijenjeni tekst.

	PRIMJER
Ulaz	Tinker
Izlaz	T*nker!!!

RJEŠENJE

```
s = input()
novi_string = s[0:1]+'*'+s[2:] + '!!!'
print(novi_string)
```

Da li su sve e-mail adrese studentske?

U nekoj školi e-mail adrese studenata se završavaju sa @student.college.edu, dok se adrese profesora završavaju sa @prof.college.edu. Napiši program koji najprije traži

od korisnika da unese podatak o broju email adresa koje želi unijeti, a zatim, nakon unosa navedenog broja konkretnih e-mail adresa program treba da odštampa odgovarajuću poruku:
- „Sve adrese su studentske.“ ili
- „Nisu sve adrese studentske.“,
u zavisnosti od toga da li su sve adrese bile studentske ili je bilo i nekih profesorskih adresa.

	PRIMJER 1	PRIMJER 2
Ulaz	3 katarina@student.college.edu marko@student.college.edu milica@prof.college.edu	3 katarina@student.college.edu marko@student.college.edu milica@student.college.edu
Izlaz	Nisu sve adrese studentske.	Sve adrese su studentske.

RJEŠENJE

```
n=int(input())
br=0
for i in range(n):
  b=input()
  if b[-19:]!='student.college.edu':
    br+=1
if br>0:
  print('Nisu sve adrese bile studentske.')
else:
  print('Sve adrese su bile studentske.')
```

Da li tekst sadrži tačku?

Napiši program koji za unijeti tekst provjerava da li se u tekstu nalazi tačka “.“. Ukoliko tekst sadrži bar jednu tačku, štampa se dio teksta do njenog prvog pojavljivanja. Ukoliko unijeti tekst ne sadrži tačku, štampa se tekst: “Tekst ne sadrži tačku.”.

	PRIMJER 1	PRIMJER 2
Ulaz	abc.defgh.ijkl	ajfasfm
Izlaz	abc	Tekst ne sadrži tačku.

RJEŠENJE

```
s = input()
for i in range(1, len(s)):
  if s[i] == '.':
    print(s[0:i])
    break
else:
  print('Tekst ne sadrži tačku.')
#provjeri kod
```

Broj pojavljivanja riječi u tekstu

Napiši program koji računa koliko se puta u unijetom tekstu pojavljuje riječ ‘Tinker’.

	PRIMJER 1	PRIMJER 2
Ulaz	Tinkeri programiraju Tinker Robota.	TinkTinkTink
Izlaz	2	0

RJEŠENJE

```
s = input()
b = 0
for i in range(len(s)):
  if s[i:i+6] == 'Tinker':
    b = b + 1
print(b)
```

I5 IZAZOVI (STRINGOVI(NISKE))

Šifriranje poruka

Jednostavan i veoma star metod slanja tajnih poruka je šifrovana zamjena slova. U osnovi, svako slovo se mijenja nekim drugim, unaprijed dogovorenim, slovom. Na primjer, svako „a“ se mijenja u „x“,

svako „b“ u „z“, i tako dalje. Napiši program kojim se implementira ovaj metod koristeći šifriranje slova abecede definisano sljedećim stringovima:
abeceda = ‘abcdefghijklmnopqrstuvwxyz’
kod = ‘xznlwebgjhqdyvtkfuompciasr’ .

(n-to slovo stringa abeceda šifrira se n-tim slovom stringa *kod*)

Napomena:
Prije šifriranja sva velika slova treba zamijeniti sa malim slovima. Za potrebe ovog zadatka, karakteri koji nisu slova ostaju nepromijenjeni.

	PRIMJER 1	PRIMJER 2
Ulaz	more	tinker
Izlaz	ytuw	mjvqwu

RJEŠENJE

```
s = input()
s = s.lower()
s1 = ''
a = ‘abcdefghijklmnopqrstuvwxyz’
k = ‘xznlwebgjhqdyvtkfuompciasr’
for i in range(len(s)):
  for j in range(len(a)):
    if(s[i] == a[j]):
      s1 = s1 + k[j]
print(s1)
```

Poseban anagram

Anagram neke riječi je riječ koja se dobija premještanjem slova polazne riječi. Na primjer, neki od anagrama riječi *idle* su: deli, ledi i lied. Napiši program koji za unijetu riječ štampa jedan poseban anagram, koji se dobija od unijete riječi na sljedeći način:
- ukoliko unijeta riječ ima paran broj slova, konačan anagram dobijamo tako što uzimamo redom
  - prvo slovo sa lijeve strane unijete riječi, pa prvo slovo sa desne strane, zatim
  - drugo slovo sa lijeve strane, pa drugo slovo sa desne strane itd. sve dok ne uzmemo sva slova.
- ukoliko unijeta riječ ima neparan broj slova:
  - formiramo prvo dvije riječi:
    - prvu koja se sastoji od slova sa parnih pozicija unijete riječi i
    - drugu koja se sastoji od slova sa neparnih pozicija unijete riječi
  - konačan anagram dobijamo spajanjem prve sa drugom riječju (prva+druga).

	PRIMJER 1	PRIMJER 2
Ulaz	matematika	Programiranje
Izlaz	maktietma	Pormrnergaiaj

RJEŠENJE

```
s = input()
#prvo sva slova prebacimo u mala
s = s.lower()
p = ''
if len(s) % 2 == 0:
  for i in range(len(s)//2):
    p = p + s[i] + s[len(s)-i-1]
  print (p)
else:
  s1 = ''
  s2 = ''
  for i in range(len(s)):
    if i % 2 == 0:
      s1 = s1 + s[i]
    else:
      s2 = s2 + s[i]
  print (s1+s2)
```

Suma cifara u tekstu

Napiši program koji, za unijeti tekst, ignorišući sve ostale karaktere, računa i ispisuje sumu svih cifara koje se u tom tekstu pojavljuju. Ukoliko unijeti tekst ne sadrži cifre, program treba da ispiše: 0.

Podsjetnik:
String a=’9’ možete pretvoriti u cijeli broj na sljedeći način: a=int(a).

	PRIMJER 1	PRIMJER 2
Ulaz	absdkj123af	absdkjaf
Izlaz	6	0

RJEŠENJE

```
s = input()
suma = 0
for i in range(len(s)):
  if s[i] in '1234567890':
    m = int(s[i])
    suma = suma + m
if suma > 0:
  print(suma)
else:
  print('0')
```

Palindrom

je niz karaktera koji se čita jednako sa lijeva na desno i sa desna na lijevo. Napiši program koji za unijetu riječ provjerava da li je palindrom ili ne i ispisuje:
"DA", ukoliko unijeta riječ jeste palindrom i
"NE" ukoliko nije.
Prije provjere, izvrši promjenu unijete riječi tako da sva slova budu velika ili sva slova budu mala. Primjeri palindroma: „1221" i „Ana" su palindromi.

	PRIMJER 1	PRIMJER 2
Ulaz	Ana	Marko
Izlaz	DA	NE

RJEŠENJE

```
n=input().lower()
duzina=len(n)
jeste_palindrom=True

for i in range(duzina//2):
    if n[i]!=n[duzina-(i+1)]:
      jeste_palindrom=False
      break
if jeste_palindrom:
  print('DA')
else:
  print('NE')
```

N6

FUNKCIJE

Izračunavanje površine kvadrata

Napiši funkciju *pov_kvadrata()* koja na osnovu proslijeđene dužine stranice izračunava površinu kvadrata.

Nakon toga napiši program koji korisniku omogućava da unese dužinu stranice i koji koristeći funkciju *pov_kvadrata()* računa njegovu površinu, a zatim je i ispisuje.

	PRIMJER
Ulaz	3
Izlaz	9

Varijacija:
Da li bi ispisivanje površine mogli vršiti u samoj funkciji pov_kvadrata?

RJEŠENJE

```
def pov_kvadrata(a):
  p=a**2
  return p

x=int(input())
povrsina = pov_kvadrata(x)
print(povrsina)

#print(pov_kvadrata(x))
```

Izračunavanje obima pravougaonika

Napiši funkciju *obim_*pravougaonika() koja na osnovu proslijeđenih dužina stranica pravougaonika računa njegov obim.

Nakon toga napiši program koji korisniku omogućava da unese dužine stranica pravougaonika i koji koristeći

funkciju *obim_pravougaonika()* računa njegov obim, a zatim ga i ispisuje.

	PRIMJER
Ulaz	3 4
Izlaz	14

RJEŠENJE

```
def obim_pravougaonika (a, b):
  return 2*a + 2*b

x=int(input())
y=int(input())

obim = obim_pravougaonika (x,y)
print(obim)

#print(obim_pravougaonika (x,y))
```

Prosječna vrijednost

Napiši funkciju prosjek(a,b,c) koja za proslijeđene tri cijelobrojne vrijednosti računa i vraća prosječnu vrijednost (aritmetičku sredinu) za ta tri broja.

Nakon toga napiši program koji korisniku omogućava da sa tastature unese tri cijelobrojne vrijednosti i koji koristeći funkciju prosjek(a,b,c) računa prosječnu vrijednost za ta tri broja, a zatim je i ispisuje.

	PRIMJER
Ulaz	0 10 20
Izlaz	10.0

RJEŠENJE

```
def prosjek (a, b, c):
  return (a+b+c)/3

a=int(input())
b=int(input())
c=int(input())

print(prosjek (a,b,c))
```

Obim i površina kruga

Napiši dvije funkcije koje na osnovu proslijeđene dužine poluprečnika izračunavaju i vraćaju jedna obim a druga površinu kruga.

Nakon toga napiši program koji korisniku omogućava da sa tastature unese dužinu poluprečnika (float) i koji koristeći napisane funkcije računa obim i površinu kruga, a zatim ih i ispisuje.

Formule za površinu i obim kruga (u Pajtonu) su sljedeće: O=2*r*math.pi i P=r*r*math.pi
Da bi koristili vrijednost *math.pi* potrebno je importovati *math* biblioteku: **import math**.

	PRIMJER
Ulaz	2 4
Izlaz	12.566370614359172 50.26548245743669

RJEŠENJE

```
import math
pi = math.pi

def obim(r):
  return 2*r*pi

def povrsina(r):
  return r**2*pi

r=float(input())
print (obim(r), povrsina(r))
```

Provjeri brojeve

Napiši funkciju **provjeri(a,b)** koja za proslijeđena dva cijela broja provjerava da li je bar jedan od tih brojeva

jednak 10 ili je njihov zbir jednak 10 i ukoliko jeste vrati True. Inače vrati False.

Nakon toga napiši program koji korisniku omogućava da sa tastature unese dvije cijelobrojne vrijednosti i koji štampa rezultat funkcije **provjeri(a,b)**.

	PRIMJER 1	PRIMJER 2	PRIMJER 3
Ulaz	2 10	3 7	2 3
Izlaz	True	True	False

RJEŠENJE

```
def provjeri(a, b):
  if a== 10 or b == 10 or a+b==10:
    return('True')
  else:
    return('False')

x=int(input())
y=int(input())
print(provjeri (x,y))
```

Broj crnih polja na šahovskoj tabli

Na šahovskoj tabli širine n i dužine m, gornje lijevo polje je bijele boje. Napisati funkciju broj_crnih_polja() u kojoj korisnik sa tastature unosi širinu i dužinu šahovske table i koja računa i ispisuje i vraća broj crnih polja na datoj šahovskoj tabli.

	PRIMJER 1	PRIMJER 2
Ulaz	8 8	7 7
Izlaz	32	24

RJEŠENJE

```
def broj_crnih_polja():
  n = int(input())
  m = int(input())
  br_crnih_polja=(m*n)//2
  print(br_crnih_polja)
  return br_crnih_polja

broj_crnih_polja()
#a=broj_crnih_polja()
```

Srednji

Napravi funkciju koja za tri unijeta broja vraća vrijednost drugog najvećeg broja.
Nazovite funkciju max2.

Pozovite funkciju za sljedeće parametre u datom redosljedu:

```
print(max2(1,2,3))
print(max2(2,1,3))
print(max2(3,2,1))
```

Očekivani izlaz je:
2
2
2

RJEŠENJE

```
def max2(a, b, c):
  if(a<b and a>c) or (a>b and a<c):
    return a
  elif(b<a and b>c) or (b>a and b<c):
    return b
  else:
    return c
print(max2(1,2,3))
print(max2(2,1,3))
print(max2(3,2,1))
```

N6 DOMAĆI

Predviđeno je ponavljanje do sada naučenog gradiva i vježbanje po sopstvenom izboru. Evo jednog zadatka koji ima za cilj da podstakne proces obnavaljanja:

Dužina hipotenuze (M)

Napisati program koji učitava dva pozitivna broja koji predstavljaju dužine kateta pravouglog trougla i štampa dužinu hipotenuze tog trougla.

	PRIMJER 1	PRIMJER 2
Ulaz	3 4	6 8
Izlaz	5	10

RJEŠENJE

```
import math
a=int(input())
b=int(input())
c=int(math.sqrt(a*a+b*b))
print(c)
```

I6 IZAZOVI (FUNKCIJE)

Saberi sve do 13

Napiši funkciju koja za unijetu vrijednost **n** računa zbir **n** unešenih brojeva. Ukoliko, tokom unosa tih n brojeva, u nekom trenutku korisnik unese broj 13, program ga ne treba dodati u zbir. I ne samo to: program ne treba dodati u zbir i sve ostale brojeve unijete nakon broja 13.

	PRIMJER 1	PRIMJER 2
Ulaz	3 1 1 1	4 1 9 13 14
Izlaz	3	10

RJEŠENJE

```
def sve_do_13():
  n = int(input())
  suma = 0
  indikator_13 = False
  for i in range(1, n+1):
    m = int(input())
    if m == 13:
      indikator_13 = True
    if m != 13 and not indikator_13:
      suma = suma + m
  return suma
print(sve_do_13 ())
```

Brojevi čiji kvadrati koji imaju cifru jedinice 4

Napiši funkciju kvadrat4() koja vraća podatak o tome koliko se kvadrata brojeva od 1 do 100 završava cifrom 4.

Npr. jedan broj čiji se kvadrat završava brojem 4 je broj 8, jer je 8 na kvadrat 64.

Obrati pažnju: funkciji ne prosljeđujemo ni jednu vrijednost.

Izlaz	20

RJEŠENJE

```
def kvadrat4():
  br = 0
  for i in range(1, 101):
    r = i**2
    cif = r%10
    if cif == 4:
      br = br+1
  return br

print(kvadrat4())
```

Prebroj riječi koje se završavaju sa 'e' ili 'r'

Napravi funkciju prebroj_ER, koja za proslijeđeni tekst broji koliko riječi u tekstu se završava sa slovima **'e' ili 'r'**. Nije bitno da li su slova **'e' i 'r'** velika ili mala.

	PRIMJER
Ulaz	Tinker mikser robot dijete kod
Izlaz	3

RJEŠENJE

```
def prebroj_ER(s):
  s=s.lower()
  n = 0
  for i in range(0, len(s)):
    if(s[i] == " "):
      if(s[i-1] in "er"):
        n = n + 1
#posljednji karakter u tekstu (ne kraju teksta ne
#mora biti blanko)
  if(s[len(s) - 1] in "er"):
    n = n + 1
  return n

tekst=input()
print(prebroj_ER(tekst))
```

Trojka u stringu

Reći ćemo da imamo „trojku u stringu" kada se neki karakter ponavlja tri puta zaredom. Napiši funkciju BrojTrojki(str) koja prebrojava i vraća broj trojki u datom stringu.

	PRIMJER 1	PRIMJER 2	PRIMJER 3
Ulaz	abcXXXabc	xxxabyyyycd	a
Izlaz	1	3	0

RJEŠENJE

```
def BrojTrojki(str):
  str=str.lower()
  br=0
  for i in range(len(str)-2):
    if str[i]==str[i+1]==str[i+2]:
      br=br+1
  return br

x=input()
print(BrojTrojki(x))
```

Izračunaj vrijednost izraza

Napiši funkciju „zbir" koja za proslijeđenu vrijednost n (n je prirodan broj) računa i vraća vrijednost sljedećeg izraza:

$1-2+3-4+...+(-1)^{n+1}\cdot n$

Na primjer, ukoliko je n=1994, funkcija treba da vrati vrijednost sljedećeg izraza:

$1-2+3-4+...+1993-1994.$

Ukoliko je n=1995, ona bi trebala da izračuna i vrati vrijednost narednog izraza:

$1-2+3-4+...+1993-1994+1995.$

Primjeri pozivanja funkcije:
print(zbir(1994))
print(zbir(1995))

izlaz:
-997
998

RJEŠENJE

```
def zbir(n):
  if n%2 != 0:
    return (((n-1)//2)+1)
  else:
    return (-(n//2))
print(zbir(1994))
print(zbir(1995))
```

N7

LISTE

Loto lista (sa mogućim ponavljanjem brojeva)

Napiši program koji generiše listu L koja je sačinjena od 7 slučajnih brojeva iz intervala [1, 39]. Za razliku od loto brojeva, u našoj listi L brojevi se mogu ponavljati.

PRIMJER IZLAZA
[34,21,1,3,16,14,1]

RJEŠENJE

```
from random import randint

L=[]
for i in range(7):
  L.append(randint(1,39))
print(L)
```

Kvadriraj elemente liste

Napiši funkciju *kvadriraj()* koja mijenja svaki element iz liste L sa njegovim kvadratom.

Primjer:

L= [1,2,3,4,5]
kvadriraj(L)
izlaz: [1,4,9,16,25]

RJEŠENJE

```
def kvadriraj(x):
  for i in range(len(x)):
    x[i]=x[i]**2
  return x

L=[1,2,3,4,5]
print(kvadriraj(L))
```

Koliko je elementa liste veće od 50?

Napravite funkciju VeciOd50(L) koja prebrojava koliko elemenata iz liste L je veće od 50. Svi elementi liste L su brojevi.

Primjer:

L= [1,5,50,100,201,1,2,3]
VeciOd50(L)

izlaz: 2

RJEŠENJE

```
def VeciOd50(x):
  br=0
  for i in range(len(x)):
    if x[i]>50:
      br=br+1
  return br

L= [1,5,50,100,201,1,2,3]
VeciOd50(L)
#print(VeciOd50([1,5,50,100,201,1,2,3]))
```

Zamijeni sa većim

Napiši funkciju *ZamijeniVecim()* koja za agrument ima listu cijelih brojeva dužine 3 i koja prvo određuje koji član liste je veći: prvi ili posljednji, a zatim svim ostalim elementima liste dodijeli tu vrijednost i štampa novodobijenu listu.

Primjeri pozivanja funkcije:

```
print(ZamijeniVecim([1,2,3]))
print(ZamijeniVecim([1,2,4]))
print(ZamijeniVecim([5,2,3]))
```

IZLAZ
[3,3,3]
[4,4,4]
[5,5,5]

RJEŠENJE

```
def ZamijeniVecim (x):
  veci=max(x[0],x[2])
  x[0]= veci
  x[1]= veci
  x[2]= veci
  return x
#ili:
'''
def ZamijeniVecim (x):
  veci=max(x[0],x[2])
  for i in range (3):
    x[i]= veci
  return x
'''
print(ZamijeniVecim([1,2,3]))
print(ZamijeniVecim([1,2,4]))
print(ZamijeniVecim([5,2,3]))
```

Prebroj parne brojeve

Napiši funkciju parni(L) koja prebrojava i vraća broj parnih brojeva u datoj listi.

Primjeri pozivanja funkcije:

```
print(parni ([1,2,3,4,5,6]))
print(parni ([2,4,6]))
print(parni ([1,3,5]))
```

IZLAZ

```
3
3
0
```

RJEŠENJE

```
def parni(x):
  br=0
  for i in range(len(x)):
     if x[i]%2==0:
       br=br+1
  return br

print(parni ([1,2,3,4,5,6]))
print(parni ([2,4,6]))
print(parni ([1,3,5]))
```

N7 DOMAĆI

Rotacija liste

Napiši funkciju rotacija(L) koja za datu listu dužine 3 vraća niz u kojem su svi elementi „zarotirani u lijevo“, tj. niz [1,2,3] postaje niz [2,3,1].

Primjeri pozivanja funkcije:

```
print(rotacija([1,2,3]))
print(rotacija([1,11,9]))
print(rotacija([1,3,10]))
```

IZLAZ

```
[2,3,1]
[11,9,1]
[3,10,1]
```

RJEŠENJE

```
def rotacija(x):
  L=[]
  L.append(x[1])
  L.append(x[2])
  L.append(x[0])
  return L

print(rotacija([1,2,3]))
print(rotacija([1,11,9]))
print(rotacija([1,3,10]))
```

Saberi prva dva

Napravi funkciju *zbir(L)* koja sabira prva dva elementa iz proslijeđene liste i vraća dobijeni zbir. Ukoliko je dužina proslijeđene liste 1, funkcija treba vratiti vrijednost prvog elementa liste. Ukoliko je proslijeđena lista prazna, funkcija treba vratiti 0. Prepostavka zadatka je da su svi elementi liste, ukoliko ih ima, brojevi.

Primjeri pozivanja funkcije:

```
print(zbir([1]))
print(zbir([]))
print(zbir([3,4,5,6,7]))
```

IZLAZ

```
1
0
7
```

RJEŠENJE

```
def zbir(x):
  if len(x)==0:
    return 0
  elif len(x)==1:
    return x[0]
  else:
    return x[0]+x[1]

print(zbir([1]))
print(zbir([]))
print(zbir([3,4,5,6,7]))
```

I7 IZAZOVI (LISTE)

Da li su prvi ili posljedni elementi međusobno jednaki?

Napiši funkciju *prvi_ili_poslednji(L1, L2)* koja za proslijeđene dvije liste L1 i L2 vraća True ako date liste imaju iste prve elemente ili ako imaju iste posljednje elemente. U suprotnom, vraća False.

Za potrebe ovog zadatka možeš pretpostaviti da oba niza imaju bar jedan element.

Primjeri pozivanja funkcije:

```
print(prvi_ili_poslednji ([1,2,3], [3,2,1]))
print(prvi_ili_poslednji ([3,2,3], [3,2,1]))
print(prvi_ili_poslednji ([1,2,4], [3,2,4]))
```

IZLAZ

```
False
True
True
```

RJEŠENJE

```
def prvi_ili_poslednji (a,b):
    if a[0]==b[0] or a[-1]==b[-1]:
        return True
    else:
        return False

print(prvi_ili_poslednji ([1,2,3], [3,2,1]))
print(prvi_ili_poslednji ([3,2,3], [3,2,1]))
print(prvi_ili_poslednji ([1,2,4], [3,2,4]))
```

Istraži listu

Napiši funkciju razno() kojoj se proslijeđuje lista. Funkcija treba da:

- štampa **ukupan broj podataka u listi**,
- štampa **posljednji podatak u listi**,
- štampa **listu obrnutim redosljedom**,
- štampa 'DA' **ako lista sadrži karakter '5'**, a 'NE' ukoliko ga ne sadrži, sa podatkom o **broju petica u listi**, a zatim
- izbriše **prvi i posljednji podatak iz liste, sortira preostale podatke i štampa rezultat.**

Primjer:

```
L=[1,2,3,4,5,6]
razno(L)

izlaz:
6
6
[6, 5, 4, 3, 2, 1]
DA, broj petica je: 1
[2, 3, 4, 5]
```

RJEŠENJE

```
def razno(x):
  #štampa ukupan broj podataka u listi
  print(len(x))

  #štampa posljednji podatak u listi
  print(x[-1])

  #štampa listu obrnutim redosledom
  x.reverse()
  print(x)
```

```
#štampa 'DA' ako lista sadrži karakter '5'  i
#štampa 'NE' ukoliko ga ne sadrži,
#sa podatkom o broju petica u listi
p=0
for i in range(len(x)):
  if x[i]==5:
    p=p+1
if p>0:
  print('DA, broj petica je: ', p)
else:
  print('NE, broj petica je: ', 0)

#briše prvi i poslednji podatak iz liste i sortira u
#obrnutom redosledu
x=x[1:len(x)-1]
x.reverse()

return x

print(razno([1,2,3,4,5,6]))
```

N7 VJEŽBANJE

Rotacija liste

Saberi sve brojeve iz intervala [1,30] koji su djeljivi sa 9 koristeći **while** petlju.

RJEŠENJE

```
br=0
i=1
while i<30:
  if i%9==0:
   br=br+i
  i=i+1
  print(br)
```

Lozinka

Napisati program koji će korisniku dati mogućnost da pogađa vašu lozinku koja je tajna i koja ima vrijednost „python“. Korisnik može pogađati lozinku sve dok je ne pogodi. Kada korisnik pogodi lozinku, potrebno je ispisati poruku „To je to! Upisali ste tačnu lozinku!“, u suprotnom obavijestiti korisnika o pogrešno upisanoj lozinci i ponoviti unos.

RJEŠENJE

```
lozinka=("python")
pokusaj=str(input())
while pokusaj!= lozinka:
  print("Niste upisali tačnu lozinku.")
  pokusaj=str(input())
else:
        print("To je to! Upisali ste tačnu lozinku!")
```

Ima li negativnih?

Napisati program u kojem se najprije unosi podatak o broju provjera koje želimo da izvršmo, a zatim za svaki unijeti broj provjerava da li među učitanim brojevima postoji negativan broj i ispisuje odgovarajuća poruka: „Postoji“ ili „Ne postoji“

RJEŠENJE

```
n=int(input())
p=0
for n in range(1,n+1):
  m=int(input())
  if m<0:
    p=p+1
if p!=0:
  print("Postoji")
else:
  print("Ne postoji")
```

Veći od 10

Napiši program koji za 10 brojeva koje unosi korisnik, prebrojava koliko od tih brojeva je veće od 10.

	PRIMJER
Ulaz	1 2 3 4 10 11 12 13 14 15
Izlaz	5

RJEŠENJE

```
brojac=0
for i in range(10):
  a = int(input())
  if a>10:
    brojac = brojac + 1
print (brojac)
```

Prebrojavanje

Koliko ima brojeva, od ukupno 7 unijetih, koji su veći od 11? Takođe odredi koliko je od tih unijetih brojeva negativno, ali i veće od broja -10.

Pomoć:

Da prebrojimo dvije različite stvari potrebna su nam dva različita brojača.

	PRIMJER
Ulaz	-1 -2 -3 -10 1 2 3
Izlaz	0 3

RJEŠENJE

```
brojacn = 0
brojacp = 0
for i in range(7):
  a = int(input())
  if a>11:
    brojacp = brojacp + 1
  if a<0 and a>-10:
    brojacn = brojacn + 1
print (brojacp)
print (brojacn)
```

Rezultati sa testa

Tražite od korisnika da unese 10 rezultata nekog testa. Napišite program koji radi sledeće:

- Štampa najveći i najmanji rezultat.
- Štampa prosječan rezultat.
- Ako je bilo koji rezultat veći od 100, nakon što su svi rezultati učitani, štampa poruku upozorenja da je unijeta vrijednost veća od 100.

	PRIMJER
Ulaz	2 3 100 101 102 25 24 23 22 21
Izlaz	102 2 42.3

Oprez, unijet je broj veći od 100!

RJEŠENJE

```
m = int(input())
max = m
min = m
```

```
zb = m
pr = 0
brojac = 0
for i in range(9):
  a = int(input())
  if a > max:
    max = a
  if a < min:
    min = a
  zb = zb + a
  if a>100:
    brojac = brojac + 1
  if m>100:
    brojac = brojac + 1
pr = zb / 10
print(max, min)
print(pr)
if brojac > 0:
  print("Oprez, unijet je broj veci od 100! ")
```

Zbir cifara unijetog broja

Napisati program koji za unijeti broj n (n > 0) ispisuje zbir njegovih cifara.

	PRIMJER 1	PRIMJER 2
Ulaz	123	5403
Izlaz	6	12

RJEŠENJE

```
n = int(input())
zbir = 0
cif = 0
if n < 0:
  n = int(input())
while n>0:
  if n == 0:
    print(zbir)
  cif = n%10
  n = n//10
  zbir = zbir + cif
print(zbir)
```

Minuti i sekunde

Napišite program koji traži od korisnika da unese broj sekundi a onda štampa koliko je to minuta i sekundi. Na primjer 200 sekundi je 3 min i 20 s, a 120 sekundi je 2 min.

	PRIMJER 1	PRIMJER 2
Ulaz	3600	3620
Izlaz	60 min	60 min i 20 s

RJEŠENJE

```
v = int(input())
if v%60==0:
  print(v//60, 'min')
else:
  print(v//60, 'min', 'i' , v%60, 's')
```

Prost broj

Za uneseni broj p, ispitaj da li je prost. Ukoliko jeste, ispiši "DA", inače ispiši "NE".

Podsjetnik:

Broj je prost ukoliko su jedini njegovi djelioci brojevi 1 i sam taj broj.

	PRIMJER 1	PRIMJER 2
Ulaz	13	32
Izlaz	DA	NE

RJEŠENJE

```
p = int(input())
indikator=-1
for i in range (2, p):
  if p % i == 0:
    indikator = 0
    break
```

```
  i = i + 1
if indikator==0:
  print("NE")
else:
  print ("DA")
```

Čudan zbir

Za date vrijednosti dvije promjenljive, prikaži njihov zbir. Ipak, ako se unesu dvije iste vrijednosti, vrati dva puta veću sumu.

	PRIMJER 1	PRIMJER 2
Ulaz	3 4	3 3
Izlaz	7	12

RJEŠENJE

```
a = int(input())
b = int(input())
if a != b:
  print(a+b)
else:
  print((a+b)*2)
```

Da li pripada skupu?

Unosi se broj n koji pripada skupu A={1, 2, ... , 10} (n je prirodan broj i 0<n<11), i string oblika „DA“ ili „NE“. Ukoliko je i unesen string „DA“, tada:

- štampati tekst 'Pripada skupu A', ako n pripada skupu A,
- štampati tekst 'Ne pripada skupu A', ako n ne pripada skupu A.

A ako je unesen string oblika 'NE', tada:

- štampati tekst 'Ne pripada skupu A', ako n pripada skupu A,
- štampati tekst 'Pripada skupu A', ako n ne pripada skupu A.

Primjeri:

n	s	Očekivani izlaz
8	DA	Pripada skupu A
11	DA	Ne pripada skupu A
11	NE	Pripada skupu A
8	NE	Ne pripada skupu A

RJEŠENJE

```
n = int(input())
s = str(input())
if s.upper()=="DA":
  if n>=1 and n<=10:
    print("Pripada skupu A")
  else:
    print("Ne pripada skupu A")
else:
  if n>=1 and n<=10:
    print("Ne pripada skupu A")
  else:
    print("Pripada skupu A")
```

Koliko ima cifara?

Za unijeti broj n odredi koliko on ima cifara.

	PRIMJER 1	PRIMJER 2
Ulaz	1234	1
Izlaz	4	1

RJEŠENJE

```
n = int(input())
br = 0
while n > 0:
  if n == 0:
    print(br)
  br = br + 1
  n = n // 10
print(br)
```

Igra sa ciframa dva broja

Za unijete brojeve a i b odredi proizvod cifara datih brojeva. Saberi dobijene proizvode i ukoliko je zbir veći od 10, zbir povećaj dva puta i prikaži vrijednost zbira. Ukoliko je zbir manji ili jednak od 10, tada prikazati prvobitni zbir.

	PRIMJER 1	PRIMJER 2
Ulaz	1001, 2002	113, 222
Izlaz	0	22

RJEŠENJE

```
a = int(input())
b = int(input())
pr1 = 1
pr2 = 1
cif1 = 0
cif2 = 0
while a > 0:
  if a == 0:
    print(pr1)
  cif1 = a % 10
  pr1 = pr1 * cif1
  a = a //10
#print(pr1)
while b > 0:
  if b == 0:
    print(pr2)
  cif2 = b % 10
  pr2 = pr2 * cif2
  b = b // 10
#print(pr2)
if pr1 + pr2 > 10:
  print((pr1 + pr2)*2)
else:
  print(pr1 + pr2)
```

Memorisanje cifara broja

Petar i Marko se igraju igre memorisanja cifara brojeva. Petar diktira Marku cifre jednog šestocifrenog broja, jednu po jednu, otpozadi, krenuvši od cifre jedinica. Napiši program koji pomaže Marku da odredi broj čije cifre Petar diktira.

ulaz:
Sa standardnog ulaza unosi se 6 cifara, svaka u posebnom redu.
izlaz:
Na standardni izlaz ispisati jedan cio broj - broj čije cifre su unijete sa standardnog ulaza.

	PRIMJER
Ulaz	1 2 3 4 5 6
Izlaz	654321

RJEŠENJE

```
# c0, c1, c2, ... - cifra jedinica, desetica, stotina, ...
# ucitavamo sve cifre
c0 = int(input())
c1 = int(input())
c2 = int(input())
c3 = int(input())
c4 = int(input())
c5 = int(input())
# odredjujemo vrijednost broja na sljedeci nacin
broj = 100000 * c5 + 10000 * c4 + 1000 * c3 + 100
* c2 + 10 * c1 + 1 * c0
# ispisujemo konačnu vrijednost broja
print(broj)
```

Igra sa ciframa

Napisati program koji učitava šestocifreni prirodan broj n i štampa razliku brojeva x i y, gdje je x broj koji se dobija brisanjem svih cifara broja n čija je pozicija parna (gledajući sa desna u lijevo), a y je broj koji se dobija brisanjem svih cifara broja n koje se nalaze na neparnim pozicijama (gledajući sa desna u lijevo). Pozicije cifara u broju brojimo od 0, počevši sa desne strane broja.

Npr. za unijeti broj 123456, broj x=246, dok je broj y=135.

	PRIMJER
Ulaz	123456
Izlaz	111

RJEŠENJE

```
n=int(input())
m=n
x=0
i=0
while n>0:
  x=x+(n%10)*(10**i)
  n=n//100
  i=i+1

y=0
j=0
m=m//10
while m>0:
  y=y+(m%10)*(10**j)
  m=m//100
  j=j+1
print(x-y)
```

Zamjeni 0 sa 5

Napisati program koji u datom prirodnom broju svako pojavljivanje cifre 0 zamjenjuje cifrom 5.

Napomena:

Zadatak uraditi bez korišćenja stringova.

	PRIMJER
Ulaz	1005
Izlaz	1555

RJEŠENJE

```
n=int(input())
b=0
i=1
if n%10!=0:
  b=n%10
  n=n//10
else:
  b=5
  n=n//10
while n>0:
  if n%10!=0:
    b=b+10**i*(n%10)
    n=n//10
    i=i+1
  else:
    b=b+10**i*5
    n=n//10
    i=i+1
print(b)
```

Armstrongov broj

Napisati program koji za dati prirodan broj n provjerava da li je taj broj Armstrongov. K-tocifren broj je Armstrongov ako je jednak sumi k-tih stepena svojih cifara. Na primjer, 370 je Armstrongov jer je 370=3**3+7**3+0**3, dok recimo broj 12 nije Armstrongov jer 12 nije jednako 1**2 + 2**2.

Podsjetnik:

treći stepen broja a u Pythonu zapisujemo: a**3
k- ti stepen broja a u Pythonu zapisujemo: a**k.

	PRIMJER 1	PRIMJER 2
Ulaz	1002	370
Izlaz	NE	DA

RJEŠENJE

```
n=int(input())
m=n
p=n
br=0
while n>0:
  br=br+1
  n=n//10
b=0
while m>0:
  b=b+(m%10)**br
  m=m//10
if b==p:
  print('DA')
else:
  print('NE')
```

Tenis

Napiši program koji na osnovu broja osvojenih gemova dva igrača određuje ishod seta u tenisu.

Napomena:
Ukoliko ne znate pravila igre koja se pominje, istražite na internetu.

ulaz:
Sa standardnog ulaza se unose dva prirodna broja koji predstavljaju broj osvojenih gemova svakog igrača redom.
izlaz:
Ako je prvi igrač osvojio set na standardni izlaz ispisati poruku pobijedio prvi. Ako je drugi igrač osvojio set ispisati pobijedio drugi. Ako je unijeti rezultat neispravan, ispisati neispravno. Ako set još nije završen ispisati nije završeno.

	PRIMJER 1	PRIMJER 2	PRIMJER 3	PRIMJER 4
Ulaz	7 6	4 6	7 4	3 2
Izlaz	Pobijedio prvi	Pobijedio drugi	Neispravno	Nije završeno

RJEŠENJE

```
a = int(input())
b = int(input())
if a == 7 and (b == 6 or b == 5):
  print (“Pobijedio prvi”)
elif (a == 6 or a == 5) and b == 7:
  print(“Pobijedio drugi”)
elif a>b and a == 6:
  print(“Pobijedio prvi”)
elif b>a and b == 6:
  print(“Pobijedio drugi”)
elif a>=b and a == 7:
  print(“Neispravno”)
elif b>=a and b == 7:
  print(“Neispravno”)
elif a>=b:
  print(“Nije zavrseno”)
elif b>=a:
  print(“Nije zavrseno”)
```

Pogodi broj (5 partija)

Napišite program koji traži od korisnika da pogodi slučajan broj od 1 do 10. Za pogađanje svakog zamišljenog broja korisnik ima samo jedan pokušaj. Kada pogodi broj, korisnik dobija 10 poena i poeni se dodaju na njegov rezultat koji je na početku 0. Korisnik gubi 1 poen ako da pogrešan odgovor. Dajte korisniku priliku da pogađa pet zamišljenih brojeva, i na kraju odštampajte konačan rezultat.

RJEŠENJE

```
from random import randint
s=0
p=0
for i in range(5):
  a=randint(1,10)
  pokusaj=int(input())
  if a==pokusaj:
    print(‘Pogodili ste’)
    p=p+10
  else:
    print(‘Promasili ste’)
    s=s+1
print(‘Ukupno bodova:’, p-s)
```

N8

IGRA VJEŠALA

Opis zadatka

Napiši program koji korisniku omogućava da sa računarom odigra igru vješala:

- Program na slučajan način bira zamišljenu riječ iz liste unaprijed definisanih riječi, na primjer: [‘voda’, ‘vazduh’, ‘rijeka’, ‘ptica’, ‘drvo’, ‘životinje’, ‘biljke’, ‘sunce’, ‘trava’].
- Korisnik pogađa slova iz zamišljene riječi.
- Nakon svakog pokušaja, program provjerava da li je korisnik sa tim slovom već pokušavao. Ukoliko jeste, ispisuje se odgovarajuće obavještenje o tome, a ukoliko nije, ispisuje se informacija o tome da li se unijeto slovo nalazi u riječi ili ne.
- Ukoliko korisnik nije unio veliko ili malo slovo tj. unio je neki drugi karakter ili je unio više karaktera, program mu ispisuje odgovarajuće poruke.
- Nakon svakog pokušaja, prikazuje se odgovarajuća slika vješala koja odgovara broju promašaja. Ispod vješala se ispisuju prazne crtice čiji broj odgovara broju slova u zamišljenoj riječi. Pogođena slova se ispisuju umjesto crtica koje označavaju mjesta na kojima se pogođena slova nalaze.
- Korisniku se ispisuju podaci o listi slova koje je unosio a ne nalaze se u zamišljenoj riječi (lista “promašaja”).
- Korisnik ima ograničen broj pokušaja, tj. najviše 6 pokušaja. Ukoliko utroši sve pokušaje, a ne pogodi zamišljenu riječ, ispisuje se obavještenje o tome.
- Ukoliko korisnik pogodi sva slova iz zamišljene riječi, u okviru dozvoljenog broja pokušaja, dobija odgovarajuću poruku.

Primjer toka igrice:

Nakon pokretanja igrice, korisnik dobija sljedeći prikaz:

```
V J E Š A L A

  +---+
      |
      |
      |
     ===

Riječ ne sadrži slova:

_ _ _ _ _ _ _ _ _
Pogodi slovo.
```

Ukoliko korisnik unese slovo **koje se ne nalazi** u zamišljenoj riječi, vješa se novi dio tijela. npr. ukoliko se slovo „a“ ne nalazi u zamišljenoj riječi, iscrtava se kružić (označava glavu), u listu promašaja se dodaje slovo „a“ i ponovo se ispisuje poruka: „Pogodi slovo.“.

```
  +---+
  O   |
      |
      |
     ===

Riječ ne sadrži slova: a

_ _ _ _ _ _ _ _ _
Pogodi slovo.
```

Ukoliko korisnik unese slovo **koje se nalazi** u zamišljenoj riječi, pogođeno slovo se ispisuje umjesto crtica koje označavaju mjesta na kojima se pogođeno slovo nalazi. Na primjer, ukoliko zamišljena riječ sadrži slovo “i”, korisnik će dobiti sljedeći ispis:

```
  +---+
  O   |
      |
      |
     ===

Riječ ne sadrži slova: a
_ i _ _ _ i _ _ e
Pogodi slovo.
```

Ukoliko korisnik unese slovo koje je već unio, ispisuje mu se poruka kojom se predlaže da pokuša sa nekim novim slovom. Npr. ukoliko je korisnik već unosio slovo “o”, a zatim ga ponovo unese, prikaz će izgledati ovako:

```
  +---+
  O   |
      |
      |
     ===

Riječ ne sadrži slova: a
_ i _ o _ i _ _ e
Pogodi slovo.
 o
Ovo slovo je već bilo. Pokušaj sa nekim novim slovom.
Pogodi slovo.
```

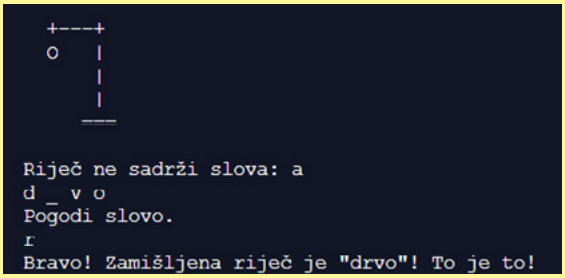
Ukoliko korisnik umjesto slova unese više karaktera, ispisuje se poruka „Molim te, unesi jedno slovo.“:

```
  +---+
  O   |
      |
      |
     ===

Riječ ne sadrži slova: a
```

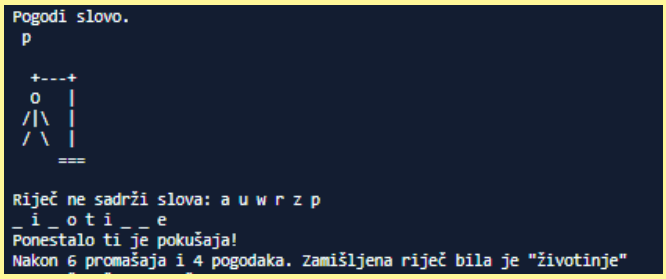
Pogledajmo kako bi mogao da izgleda završetak igrice:

Varijanta 1.

Kraj igrice je nakon što je korisnik pogodio riječ:

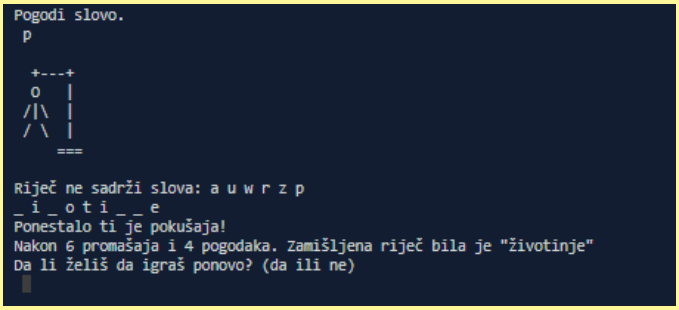


ili nakon što je korisnik utroši sve pokušaje:



Varijanta 2.

Nakon što korisnik završi igricu, možemo ga pitati da li želi da igra ponovo i pokrenuti igru iz početka, ukoliko je odgovor pozitivan:



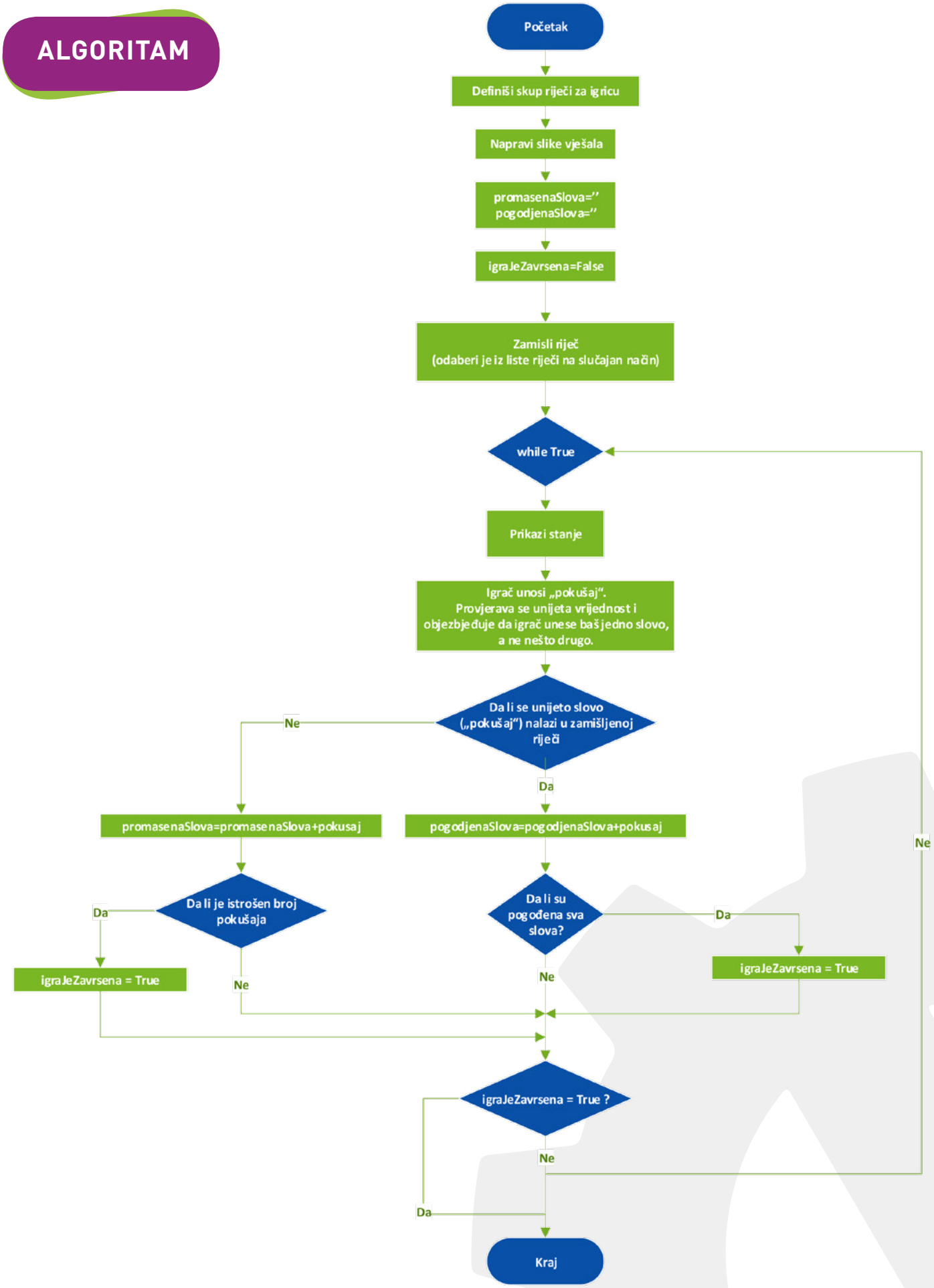
RJEŠENJE

```
import random
VJESALA_SLIKE = ['''
 +---+
   |
   |
   |
  ===''', '''
 +---+
 O |
   |
   |
  ===''', '''
 +---+
 O |
 | |
   |
  ===''', '''
 +---+
 O |
/| |
   |
  ===''', '''
 +---+
 O |
/|\ |
   |
  ===''', '''
 +---+
 O |
/|\ |
/  |
  ===''', '''
 +---+
 O |
/|\ |
/ \ |
  ===''']
rijeci = 'voda vazduh rijeka ptica drvo životinje biljke sunce trava'.split()

def uzmiSlucajnuRijec(listaRijeci):
    # Ova funkcija vraća jednu riječ(jedan string) koja se
    # na slučajan način odabira iz liste stringova.
    indeks = random.randint(0, len(listaRijeci) - 1)
    return listaRijeci[indeks]

def prikaziStanje(promasenaSlova, pogodjenaSlova,
zamisljenaRijec):
```

```
    print(VJESALA_SLIKE[len(promasenaSlova)])
    print()

    print('Riječ ne sadrži slova:', end=' ')
    for slovo in promasenaSlova:
        print(slovo, end=' ')
    print()

    blankovi = '_' * len(zamisljenaRijec)

        for i in range(len(zamisljenaRijec)): # zamjeni
blankove pogodjenim slovima
        if zamisljenaRijec[i] in pogodjenaSlova:
            blankovi = blankovi[:i] + zamisljenaRijec[i] +
blankovi[i+1:]

    for slovo in blankovi: # prikazi tajnu rijec sa blankovi
sa tajnim rijecima
        print(slovo, end=' ')
    print()

def uzmiPokusaj(ranijiPokusaji):
    # Provjerava unijetu vrijednost i obzbjeđuje da igrač
    # unese jedno slovo, a ne nešto drugo i vraća slovo
    # koje je igrač unio.
    while True:
        print('Pogodi slovo.')
        pokusaj = input()
        pokusaj = pokusaj.lower()
        if len(pokusaj) != 1:
            print('Molim te, unesi jedno slovo.')
        elif pokusaj in ranijiPokusaji:
            print('Ovo slovo je već bilo. Pokušaj sa nekim
novim slovom.')
        elif pokusaj not in 'abcdefghijklmnopqrstuvwxyz':
            print('Molim te, unesi SLOVO. :)')
        else:
            return pokusaj

def igrajPonovo():
    # Ova funkcija vraća True ukoliko igrač želi da igra
    # ponovo. Inače vraća False.
    print('Da li želiš da igraš ponovo? (da ili ne)')
    return input().lower().startswith('d')

#-----------------------------------------------------------
print('V J E Š A L A')

#Koristicemo dvije liste:
# - listu promašaja: listu slova, pokušaja korisnika,
# koja se NE NALAZE nalaze u riječi
# - listu pogodaka: listu slova, pokušaja korisnika, koja
# se  NALAZE nalaze u riječi
```

```
promasenaSlova = ''
pogodjenaSlova = ''

zamisljenaRijec = uzmiSlucajnuRijec(rijeci)

igraJeZavrsena = False

while True:
    prikaziStanje(promasenaSlova, pogodjenaSlova,
    zamisljenaRijec)

    # Igrac pogađa slovo
    pokusaj = uzmiPokusaj(promasenaSlova +
    pogodjenaSlova)

    if pokusaj in zamisljenaRijec:
        pogodjenaSlova = pogodjenaSlova + pokusaj

        # Provjeri da li je igrač pogodio sva slova.
        pogodioSvaSlova = True
        for i in range(len(zamisljenaRijec)):
            if zamisljenaRijec[i] not in pogodjenaSlova:
                pogodioSvaSlova = False
                break
        if pogodioSvaSlova:
            print('Bravo! Zamišljena riječ je "' + zamisljenaRijec
+ '"! To je to!')
            igraJeZavrsena = True
    else:
        promasenaSlova = promasenaSlova + pokusaj

        # Provjeri da li je igrač pogađao previše puta i
        # izgubio.
        if len(promasenaSlova) == len(VJESALA_SLIKE) - 1:
            prikaziStanje(promasenaSlova, pogodjenaSlova,
            zamisljenaRijec)
            print('Ponestalo ti je pokušaja!\nNakon ' +
            str(len(promasenaSlova)) + ' promašaja i ' +
            str(len(pogodjenaSlova)) + ' pogodaka. Zamišljena
            riječ bila je "' + zamisljenaRijec + '"')
            igraJeZavrsena = True

    # Ukoliko je igra završena, pitaj igrača da li želi da
    # igra ponovo.
    if igraJeZavrsena:
        if igrajPonovo():
            promasenaSlova = ''
            pogodjenaSlova = ''
            igraJeZavrsena = False
            zamisljenaRijec = uzmiSlucajnuRijec(rijeci)

        else:
            break
```

ALGORITAM


Početak
Definiši skup riječi za igricu
Napravi slike vješala
promasenaSlova=''
pogodjenaSlova=''
igraJeZavrsena=False
Zamisli riječ
(odaberi je iz liste riječi na slučajan način)
while True
Prikazi stanje
Igrač unosi „pokušaj".
Provjerava se unijeta vrijednost i
objezbjeđuje da igrač unese baš jedno slovo,
a ne nešto drugo.
Da li se unijeto slovo
(„pokušaj") nalazi u zamišljenoj
riječi
Ne
Da
promasenaSlova=promasenaSlova+pokusaj
pogodjenaSlova=pogodjenaSlova+pokusaj
Da li je istrošen broj
pokušaja
Da
igraJeZavrsena = True
Ne
Da li su
pogođena sva
slova?
Da
igraJeZavrsena = True
Ne
Ne
igraJeZavrsena = True ?
Ne
Da
Kraj

KORIŠĆENI IZVORI

- „Teach your kids to code: a parent-friendly guide to Python programming“, Bryson Payne, No Starch Press, San Francisco
- „Python for kids: a playfull introduction to programming“, Jason R. Briggs, No Starch Press, San Francisco
- „Invent your own computer games with Python“, Al Sweigart, No Starch Press, San Francisco
- „Pyhon Crash Course: a hands-on, project-based introduction to programming“, Eric Matthes, No Starch Press, San Francisco
- Zvanična Pajton dokumentacija sa stranica: https://docs.python.org, koja obuhvata i zvanični Pajton tutorijal: **https://docs.python.org/3/tutorial/**
- Pajton online kompajler i IDE: **www.repl.it**
- LeetCode: **https://leetcode.com/**
- CodingBet: **https://codingbat.com/python**
- Pinative: **https://pynative.com/**
- RealPython: **https://realpython.com/**
- LearnPython: **https://www.learnpython.org/**
- Python Practice Book, Anand Chitipothu: **https://anandology.com/python-practice-book/index.html**
- GeeksForGeeks: **https://www.geeksforgeeks.org/**

Ilustracije

- **https://www.freepik.com/** *brgfx, macrovector*
- **https://www.pinclipart.com/**
- **https://classroomclipart.com/**
- **https://vectorstock.com/** *lembergvector*
- **https://shutterstock.com/** *dencg*
- **https://www.vecteezy.com/** *Miguel Angel, Graphics RF*