Uvod u programiranje kroz programski jezik Python

Skripta za kurs: “Python”

Lazar Premović
mart 2021.

Uvod

Ova skripta je sačinjena iz dva glavne celine: objašnjenja i podsetnici za teme obrađene tokom kursa i zbirka zadataka sa rešenjma koja pokriva sve teme obrađene na kursu i može se koristiti kako za rad na casu tako i za samostalan rad.

Uvod u Python kurs

Kurs “Python” je namenjen učenicima starijih razreda osnovnih škola koji do sada nisu imali prilike da se upoznaju sa procesom programiranja i kao takav ne zahteva nikakvo predznanje iako je poznavanje Engleskog jezika i dobro znanje matematike poželjno.

Kroz ovaj kurs moći ćete da se upoznate sa osnovama procesa programiranja na primerima u programskom jeziku Python, za početak ćemo raditi na konzolnim aplikacijama dok se ne upoznamo sa konceptima grananja i petlji a onda ćemo se pozabaviti i aplikacijama sa grafičkim interfejsom.

Programski jezik Python

Python je interpretirani programski jezik opšte primene koji je razvio holandski programer Guide van Rossum 80-tih godina 20. veka. Python je interpretirani jezik visokog nivoa, baziran na imperativnoj i proceduralnoj paradigmi a podržava i funkcionalno i objektno orijentisano programiranje.

Python je izabran kao jezik koji ćemo koristiti na ovom kursu zato što je jednostavan za učenje, lako proširiv i zabavan za upotrebu a ipak je dovojno moćan da se koristi i za ozbiljnije projekte

Preuzimanje IDLE razvojno okruženje

Da bi programirali u programskom jeziku Python potrebno je da koristimo neko od razvojnih okruženja koja ga podržavaju, na kursu ćemo koristiti razvojno okruženje IDLE koje se instalira na sledeći način:
Na sajtu https://www.python.org/ možete u sekciji “Downloads” preuzeti Python sa IDLE okruženjem za vaš operativni sistem (vodite računa da preuzmete bar verziju 3), ili koristite direktni link za najnoviju verziju (3.9.2 u vreme pisanja) za Windows operativne sisteme: https://www.python.org/ftp/python/3.9.2/python-3.9.2-amd64.exe.
Nakon instalacije u listi programa ćete moći da pronađete IDLE razvojno okruženje.

Kako se koristi IDLE razvojno okruženje će biti objašnjeno u osnovnim crtama na kursu i nećemo se time baviti u ovoj skripti.
Pored razvojnih okruženja koja se instaliraju i pokreću kao desktop programi, postoji i nekoliko online okruženja koja se mogu koristiti direktno iz browsera, jedno takvo okruženje je https://www.onlinegdb.com/online_python_compiler.

0. Uvod u proces programiranja i izvršavanja koda

Kada programiramo bilo koju aplikaciju postoje dve najbitnije faze u njenoj izradi, to su: faza razade algoritma i faza kodiranja.

Algoritmi i faza razrade algoritama

Algoritam je drugi naziv za konačan niz jednostavnih koraka kojima se opisuje postupak rešavanja određenog realnog problema. A to znači da u fazi razrade algoritma mi uzimamo postavku problema (U našem slučaju tekst zadatka) i predstavljamo proces za njegovo rešavanje kao niz jednostavnih koraka koje računar može da razume (tj. Algoritam).
Ova faza će od vas zahtevati da pokažete sposobnosti logičkog razmišljanja.

Neki primeri algoritama u svakodnevnom životu:

Kuvanje jajeta:

1. Uzmemo lonče i napunimo ga vodom,
2. Uzmemo jaje i stavimo ga u lonče,
3. Stavimo lonče na ringlu i uključimo tu ringlu,
4. Čekamo da voda proključa,
5. Čekamo 5 minuta da se jaje skuva,
6. Sklonimo lonče sa ringle i isključimo tu ringlu,
7. Sačekamo da se jaje ohladi ili ga ohladimo hladnom vodom, kraj postupka.

Telefoniranje:

1. Podižemo slušalicu,
2. Ako nemamo odgovarajući signal, kraj postupka (nesupešno telefoniranje),
3. Biramo broj,
4. Ako se pozvani korisnik javi, razgovaramo
5. Prekidamo vezu, kraj postupka.

Programski kod i faza kodiranja

Kada imamo razrađen algoritam koji smo napisali na normalnom jeziku, potrebno ga je prevesti na niz naredbi programskog jezika koji smo odabrali, to se naziva fazom kodiranja zato što se niz naredbi na programskom jeziku naziva kod. Programski kod koji pišemo na nekom programskom jeziku ima svoja pravila isto kao što i jezici imaju svoju gramatiku.

Kodiranje u programskom jeziku Python

Python je jezik u kome se naredbe i programski kod pišu u vidu teksta (mala i velika slova, cifre i specijalni znaci), pritom razlikujući mala i velika slova `Ime` i `ime` **ne znače** isto. Svaka naredba u Python-u se piše u jednom redu. Naredbe u Python-u se izvršavaju odozgo na dole redosledom kojim su navedene u kodu. Kada pokrenemo program on se izvršava od prve linije koda redom naniže do poslednje linije koda, kada izvrši poslednju liniju koda program prestaje sa radom i gasi se. U skoro svakom programskom jeziku postoje komentari, oni se u Python-u se označavaju znakom taraba `#`. Računar će ignorisati tekst napisan nakon znaka za komentar, što nam omogućava da u njima zapišemo beleške.

```
a = 5
b = 3
c = a + b
#komentar
```

Primer nekoliko linija koda

1. Osnovni programi linijske strukture

Režimi izvršavanja

Python, kao i većina interpretiranih programskih jezika, pored standardnog podžava i takozvani interaktivni mod izvršavanja programa. U interaktivnom modu unosimo i izvršavamo jednu po jednu naredbu programa i interpreter nam nakon svake izvršene naredbe ispisuje njen rezultat.
U standardnom režimu izvršavanja Python izvršava sve naredbe skript fajla (skript fajl je tekstualni fajl sa ekstenzijom .py koji sadrži celokupan kod nekog programa) jednu za drugom i ispisuje rezultate samo ako to od njega zatražimo.

```
>>> print("Hello World!")
Hello World!
>>> 2 + 3
5
```

Primer izvršavanja nekoliko naredbi u interaktivnom režimu

Ugrađene funkcije za pomoć

U interaktivnom modu možemo koristiti funkciju `help()` gde ćemo između zagrada upisati ime funkcije čija nas upotreba zanima i Python će ispisati kratak opis te funkcije.

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream) defaults to the current
sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Primer traženja pomoći za funkciju print

Promenljive, naredba dodele i osnovni tipovi podataka

Jedan od osnovnih koncepata u programiranju su promenljive, promenljive su način na koji možemo da obrađujemo i čuvamo podatke unutar programa. Promenljive možemo najlakše zamisliti kao kutije koje na sebi imaju obeleženo ime u koje možemo staviti neku vrednost i posle je uzeti iz kutije i koristiti negde drugde.

Osnovna naredba u Python-u je naredba dodele, ona nam ovogućava da neku vrednost (dobijenu kao rezultat nekog izraza) zabeležimo u nekoj promenljivoj.

U Python-u nije potrebno posebno napomenuti da želimo da koristimo neku promenljivu, već joj odmah možemo dodeliti vrednost.

```
<Ime promenljive> = <Vrednost>

Broj = 5
tekst = "Zdravo!"
josJedanBroj = 3.6
drugibroj = 3 + Broj
```

Primeri dodele vrednosti promenljivama

Python podržava sledeće osnovne tipove podataka koje možemo dodeliti promenjljivama:

int	Celobrojna vrednost
float	Decimalni broj
str	Tekst (sačinjen od proizvoljnog broja karaktera)
bool	Logička vrednost (tačno **True** ili netačno **False**)

Osnovni tipovi podataka

Pravila za imenovanje promenljivih:

- Mogu da se sastoje samo od slova (malih i velikih), cifara i donjih crta (_)
- Ne smeju počinjati cifrom
- Python razlikuje mala i velika slova, `GodisnjeDoba` i `godisnjeDoba` nisu ista promenljiva
- Promenljive se ne smeju zvati po ključnim rečima Python jezika (neke od klučnih reči su: `or,for,if,class`)

Osnovni numerički operatori

Da bi promenljivama mogli da dodelimo neke smislene vrednosti moramo da znamo da pišemo izraze, izrazi u programiranju su isto što i izrazi u matematici, niz vrednosti i operacija koje vršimo nad njima. Kao vrednosti u izrazu možemo koristiti konstante, vrednosti nekih promenljivih ili rezultate nekih izraza ili operacija.

Tip vrednosti	Izraz	Vrednost
Celobrojna konstanta	`5`	`5`
Decimalna konstanta	`3.75`	`3.75`
Tekstualna konstanta (**Napomena**: tekstualne konstante pišemo između jednostrukih `' '` ili dvostrukih navodnika `" "` da bi ih računar razlikovao od imena promenljivih)	`"Zdravo svete!"`	`Zdravo svete!`
Vrednost promenljive	`broj`	Koja god vrednost se nalazi u promenljivoj broj u tom trenutku
Vrednost izraza	`3+5`	`8`

Primeri izraza

`+`	Sabiranje
`-`	Oduzimanje ili negacija
`*`	Množenje
`/`	Deljenje
`%`	Ostatak pri celobrojnom deljenju
`**`	Stepenovanje
`//`	Celobrojno deljenje

`()`	Grupisanje operacija

Primeri operatora

Unos i ispis podataka u konzolu

Nakon što smo naučili da promenljivama dodeljujemo vrednosti koje izračunavamo uz pomoć operatora, bilo bi zgodno da te vrednosti možemo nekako da prikažemo korisniku i da iskoristimo vrednosti koje korisnik unese za neka izračunavanja. Za to ćemo koristiti funkcije `print()` i `input()` (o funkcijama ćemo detaljnije pričati kasnije).

```
print("Unesite broj:")
broj = input()
print("Uneli ste:",broj)
```

Primer unosa i ispisa u konzolu

Funkcija `print()` ispisuje vrednosti koje joj se zadaju između zagrada i nakon toga prelazi u novi red. Moguće je navesti i više vrednosti odvojenih zagradama, u tom slučaju one će biti odvojene razmacima.
Funkcija `input()` učitava jedan red koji je korisnik uneo kao tekst i omogućava nam da ga sačuvamo u promenljivu.

Konverzije

Kada bi probali da broj koji je korisnik uneo saberemo sa nekim drugim brojem dobili bi grešku je je taj broj zapravo tekst tipa str a da bi mogli da ga koristimo za sabiranje moramo da ga konvertujemo u broj. U Python-u razlikujemo dve vrste konverzija: implicitne i eksplicitne. Implcitne konverzije računar radi umesto nas i o njima nema potrebe da brinemo, eksplicitne konverzije su kada mi eksplicitno kažemo računaru da pretvori neku vrednost iz jednog tipa u drugi i to je ono što je neophodno da uradimo u ovom slučaju.

```
<Ime tipa>(<Izraz>) #konvertuje izraz u dati tip

tekst="3.14"
realni=float(tekst) # realni = 3.14 kao realni broj
ceo=int(realni) # ceo = 3
```

Primer eksplicitne konverzije

1. Primeri

1. Unose se dva broja i ispisuje se njihov zbir.

```
ta=input()
tb=input()
a=int(ta)
b=int(tb)
c=a+b
print(c)
```

2. Unosi se vremenski interval kao broj sata i minuta kada je počeo i broj sata i minuta kada se završio. Ispisati koliko minuta traje interval.

```
ss=int(input())
sm=int(input())
es=int(input())
em=int(input())
s=ss*60+se
e=es*60+em
print(e-s)
```

1. Zadatci

1. Napraviti program koji učitava dva broja i prikazuje njihov proizvod na ekranu.
2. Napraviti program koji učitava tri broja i prikazuje njihov zbir na ekranu.
3. Napišite program za pretvaranje centimetara u metre. Zadaje se merni broj u centimetrima, a treba odrediti koliko je to metara.
4. Napraviti program koji računa prosek 4 uneta broja.
5. Korisnik unosi broj minuta koliko traje film, ispisati koliko je to sati i minuta.
6. Korisnik unosi u koliko sati i minuta je krenuo na autobus, u koliko sati i minuta autobus kreće i koliko minuta mu treba da stigne do stanice. Ispisati koliko minuta će on poraniti na autobus.
7. Korisnik unosi veličinu jajeta koje želi da skuva, jaje moze biti velicine od 1 do 10, jaje veličine jedan se kuva 5 minuta a za svaku jedinicu veličine preko 1 dodajemo još 2 minuta.
8. Unosi se temperatura u celzijusima, ispisati koliko je to farenhajta.

   $$T_{(°F)} = T_{(°C)} \times 1.8 + 32$$

9. Biciklista kreće da vozi bicikl u 8 ujutru, unosi se kada je završio sa treningom (kao broj sati i broj minuta) i brzina u kilometrima na čas. Ispisati koliko kilometara je prešao
10. Unosi se pravougaonik preko koordinata dve tačke, izračunati njegovu površinu i obim.
11. Unosi se trocifreni broj, zameniti njegovu cifru jedinica i stotina.
12. Osmisliti algoritam koji zamenjuje vrednosti dve promenljive.

1. Rešenja

1.

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

print(a*b)
```

2.

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

d = a + b + c

print(d)
```

3.

```
cm=int(input())

m = cm / 100

print(m)
```

4.

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

p = (a+b+c+d) / 4

print(p)
```

5.

```
UkupnoMinuta=int(input())

Sati = UkupnoMinuta // 60
Minuti = UkupnoMinuta % 60

print(Sati,":",Minuti)
```

6.

```
KrenuoS = int(input())
KrenuoMin = int(input())
Vreme = int(input())
```

```
BusS = int(input())
BusMin = int(input())

KrenuoU = KrenuoS * 60 + KrenuoMin
BusU = BusS * 60 + BusMin
Poranio = BusU - (KrenuoU + Vreme)

print(Poranio)
```

7.

```
Velicina = int(input())
Vreme = Velicina * 2 + 3

print(Vreme)
```

8.

```
TC = int(input())
TF = TC * 1.8 + 32

print(TF)
```

9.

```
S = int(input())
M = int(input())
B = int(input())

UkupnoVreme = (S * 60 + M) - 8 * 60
PP = UkupnoVreme * (B / 60)

print(PP)
```

10.

```
X1 = int(input())
Y1 = int(input())
X2 = int(input())
Y2 = int(input())
A = X2 - X1
B = Y2 - Y1
O = 2 * A + 2 * B
P = A * B

print("Obim:",O)
print("Povrsina:",P)
```

11.

```
B = int(input())
J = B % 10
```

```
D = (B // 10) % 10
S = B // 100
NB = J * 100 + D * 10 + S
print(NB)
```

12.

```
A = int(input())
B = int(input())

c = A
A = B
B = c

print(A,B)
```

3. Naredbe grananja

Do sada su se programi koje smo pisali izvršavali inijski odozgo na dole i svaka komanda je bila izvršena. Uvođenje naredbi grananja nam odmogućava da imamo delove koda koji se izvršavaju samo ako je neki logički uslov ispunjen.

Logički izrazi

Da bi mogli da formiramo uslov koji određuje da li se deo koda izvršava potrebno je da znamo da pišemo logičke izraze. Logički izrazi nisu mnogo drugačiji od izraza sa kojima smo se do sada sretali, jedina razlika je to da logički izraz na kraju vraća logičku vrednost (`True` ili `False`) i da postoje posebni operatori za rad sa logičkim vrednostima.

Logički operatori:

==	Jednakost, radi na dva operanda bilo kog tipa
!=	Nije jednako, radi na dva operanda bilo kog tipa
<	Manje, radi na dva operanda brojevnog tipa
<=	Manje jednako, radi na dva operanda brojevnog tipa
>	Veće, radi na dva operanda brojevnog tipa
>=	Veće jednako, radi na dva operanda brojevnog tipa
not	logička negacija, radi na jednom operandu logičkog tipa (bool)
and	logičko I, radi na dva operanda logičkog tipa
or	logičko ILI, radi na dva operanda logičkog tipa

A	B	not A	A and B	A or B
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Tablica logičkih operacija

Blok koda

Blok koda se na python-u formira po broju razmaka ili tabova pre početka naredbe, tj. Jedan blok koda na pythonu čine sve uzastopne naredbe koje su podjednako uvučene (indentovane). Svrha bloka koda je da se nekoliko naredbi grupišu za potrebe nekih naredbi.

Naredba grananja if

Naredba grananja if je osnovna naredba grananja i ona u svom osnovnom obliku izvršava naredbe unutar nje samo ako je njen logički uslov tačan.

```
if <Logički uslov>:
      #naredbe
```

Primer if naredbe

Klauza else

Else klauza nam omogućava da dopunimo našu if naredbu naredbama koje trebaju da se izvrše ako logički uslov nije tačan.

```
if <Logički uslov>:
      #naredbe A
else:
      #naredbe B

#if <Logički uslov>:
#     naredbe A
#if not<Logički uslov>:
#     naredbe B
```

Primer else klauze i kod koji ona zamenjuje

Klauza elif

Elif klauza nam omogućava da else klauzu proširimo dodatnim uslovom, samo ako nijedan od else if uslova nije tačan else klauza će se izvršiti.

```
if <Logički uslov 1>:
      #naredbe A
elif <Logički uslov 2>:
      #naredbe B
else:
      #naredbe C

#if <Logički uslov 1>:
#     naredbe A
#else:
#     if <Logički uslov 2>:
#           naredbe B
#     else:
#           naredbe C
```

Primer else if klauze i kod koji ona zamenjuje

Naredba try/except

try/except naredba nam omogućava da obradimo greške koje se dešavaju prilikom pokretanja programa (na primer: ako pokušamo da pretvorimo “Tekst” u broj desiće se greška prilikom izvršavanja). Try except će izvršiti komande try bloka i ako se u try bloku desi greška umesto njih će izvršiti except blok.

```
try:
      #komande za try blok
except:
      #komande koje se izvrše ako se desi greška u try blok-u
```

Primer try/except naredbe

3. Primeri

1. Maksimum dva broja.

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

2. Absolutna vrednost.

```
a = int(input())
if a < 0:
      a = -a
print(a)
```

3. Kalkulator sa dve operacije.

```
a = int(input())
b = int(input())
o = input()
if o == "+":
      print(a+b)
elif o == "-":
      print(a-b)
else:
      print("Nije zadata tacna operacija")
```

4. Program koji ispisuje puno ime strane sveta na osnovu prvog slova.

```
unos = input()
if unos == 'I' or unos == 'i':
```

```
    print("Istok")
elif unos == 'Z' or unos == 'z':
    print("Zapad")
elif unos == 'S' or unos == 's':
    print("Sever")
elif unos == 'J' or unos == 'j':
    print("Jug")
else:
    print("Ne postoji to pocetno slovo")
```

3. Zadatci

1. Biciklista Zdravko planira da kupi novi bicikl kako bi celo leto vozio krugove po Adi, dobio je ponudu od Mileta da povoljno kupi bicikl, prijatelj mu nudi dva bicikla po različitim cenama a Zdravko planira da kupi jeftiniji pošto su oba odgovarajućeg kvaliteta. Napišite program koji će Zdravku da pomogne u izboru.
2. Zdravko je kupio novi bicikl ali i dalje nije u dobroj formi za vožnju, Mile mu je predložio da idu da se provozaju za vikend i ponudio je tri različite putanje. Pošto Zdravko nije u formi napišite program koji će mu pomoći da odabere turu sa najmanje kilometara.
3. Posle par treninga Zdravko je počeo redovno da vozi ali zbog obaveza ne može svaki dan da trenira. Ako mu dnevna vožnja iznosi 20km i nikad više od toga, napišite program koji računa koliko dana u nedelji Zdravko mora da vozi bicikl kako bi prešao bar X kilometara, gde se X unosi. Ako je nemoguće da se za nedelju dana pređe X kilometara ispisati odgovarajuću poruku.
4. Tokom jedne od vožnji sa Raletom naišli su na raskrsnicu i tri moguća puta do kuće (A,B i C). Pošto obojica hoće što više da treniraju taj dan, napisati program koji će odabrati za njih najduži put od tri uneta.
5. Zdravko prilikom planiranja vožnje za sledeći vikend ima dilemu, ako je vožnja kraća od 50km hteo bi da vozi po brdu a ako je duža voziće po ravnici. Zatim kada odabere put treba da odluči koliko vode da ponese pošto nigde usput nema nijedna česma. Ako ide po brdu trebaće mu 1 litar za svakih 5km, a ako ide po ravnom treba mu 1 litar za svakih 10 km. Takođe ako ide u brdsku vožnju poneće fotoaparat da slika veverice u šumi a ako ide po ravnom treba da ponese mrvice da nahrani patke na reci. Napišite program koji za unetu kilometražu ispisuje putanju, količinu vode i stvari koje Zdravko nosi sa sobom.
6. Napisati program koji proverava da li je uneta godina prestupna (deljiva sa 4 i ne sa 100 ili deljiva sa 400).
7. Napisati program koji za uneta tri ugla trougla proverava da li taj trougao postoji.
8. Napisati program koji za unetu tačku (unose se X i Y koordinate) pronalazi u kom kvadrantu se nalazi ta tačka.
9. Unosi se zaključna ocena (int), ispisati tekstualni opis ocene (odlican,vrlo dobar,...).
10. Kalkulator sa sve 4 operacije (opciono ubaciti try except).

3. Rešenja

1.

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

2.

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

3.

```
a = int(input())
if a < 140:
    print((a + 19) / 20)
else:
    print("Nije moguce da postigne zadatu kilometrazu")
```

4.

```
a = int(input())
b = int(input())
c = int(input())
if a >= b and a >= c:
    print(a)
elif b >= a and b >= c:
    print(b)
else:
    print(c)
```

5.

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

```
if a < 50:
    print("Planina")
    print((a + 4) / 5)
    print("Fotoaparat")
else:
    print("Ravnica")
    print((a + 9) / 10)
    print("Mrvice")
```

6.

```
a = int(input())
if (a % 4 == 0 and a % 100 != 0) or a % 400 == 0):
    print("Prestupna")
else:
    print("Nije prestupna")
```

7.

```
a = int(input())
b = int(input())
c = int(input())
if a + b + c == 180:
    print("Postoji")
else:
    print("Ne postoji")
```

8.

```
x = int(input())
y = int(input())
if x >= 0:
    if y >= 0:
        print(1)
    else:
        print(4)
else:
    if y >= 0:
        print(2)
    else:
        print(3)
```

9.

```
a = int(input())
if a==1:
    print("Nedovojan")
elif a==2:
    print("Dovojan")
elif a==3:
    print("Dobar")
```

```
elif a==4:
    print("Vrlo Dobar")
elif a==5:
    print("Odlican")
else:
    print("Ocena mora biti <= 5")
```

10.

```
try:
    a = int(input())
    b = int(input())
    op = input()
    if op == '+':
        print(a+b)
    elif op == '-':
        print(a-b)
    elif op == '/':
        print(a/b)
    elif op == '*':
        print(a*b)
except:
    print("Pogresan unos")
```

4. Naredbe Ponavljanja

Do sada su se sve naredbe koje smo pisali izvršavale tačno jednom ili nijednom (korišćenjem naredbi grananja), ali nekada je potrebno da se neke naredbe izvrše više puta ili da se izvršavaju dokle god je neki uslov ispunjen.

Naredba ponavljanja while

While je osnovna naredba ponavljanja, sastoji se od uslova i bloka koda. While naredba izvršava blok koda dokle god je uslov ispunjen.

```
while <Logički uslov>:
      #naredbe
```

Primer while naredbe

Naredba ponavljanja for

For je naredba koja određen blok naredbi ponavlja neki broj puta ili za svaki broj/element iz sekvence/kolekcije. Sekvence brojeva za for petlju se najlakše stvaraju funkcijom `range()`, funkcija range ima tri moguća potpisa:

- `range(<start>,<stop>,<step>)`
- `range(<start>,<stop>)` podrazumeva <step>=1
- `range(<stop>)` podrazumeva <step>=1 i <start>=0

Funkcija range() će generisati sekvencu u opsegu [start,stop) (start je uključen a stop ne), opciono step definiše korak kojim će se generisati vrednosti (mora biti različit od 0 a može biti negativan za opadajuće sekvence)

```
for <iterator> in <sekvenca>:
      #kod
```

Primer for naredbe

Pri izvršavanju for petlje, promenljiva <iterator> u svakoj iteraciji redom uzima vrednost jednog elementa u sekvenci.

Naredbe pass, break, continue i složeni operatori

Naredba **`pass`** ne radi ništa i efektivno predstavlja prazan blok koda. Koristi se kada neka druga naredba zahteva bar jedan blok koda a želimo da testiramo program pre pisanja konkretnih naredbi u tom bloku koda.

Kada program naiđe na naredbu **`break`** svoje izvršavanje će nastaviti nakon petlje unutar koje se naredba break nalazi, time možemo da prevremeno prekinemo izvršavanje neke petlje. Slično tome pri nailasku na naredbu **`continue`** program nastavlja izvršavanje od

sledeće iteracije petlje, time možemo da preskočimo određene naredbe u trenutnoj iteraciji petlje.

```
s=0
while True:
    n=int(input())
    if n<0:
        continue
    elif n==0:
        break
    s+=n
print("Suma je:",s)
```

Primer naredbi break i continue

U programiranju ćemo često sretati naredbe formata:
`<Promenljiva A> = <Promenljiva A> <Operacija> <Promenljiva B>`
i zbog toga python uvodi kraći način da se takve naredbe pišu:
`<Promenljiva A> <Operacija>=<Promenljiva B>`

4. Primeri

1. Ispisati prvih 10 prirodnih brojeva.
   a. While

```
i = 0
while i < 10:
    print(i+1)
    i+=1
```

   b. For

```
for i in range(10):
    print(i+1)
```

2. Ispisati sumu prvih 10 prirodnih brojeva.
   a. While

```
i = 0
sum=0
while i < 10:
    sum += i + 1
    i+=1
print(sum)
```

   b. For

```
sum = 0
for i in range(10):
    sum += i + 1
```

```
print(sum)
```

4. Zadatci

1. Ispisati program koji ispisuje prvih n prirodnih brojeva.
2. Ispisati program koji ispisuje sumu prvih n prirodnih brojeva.
3. Ispisati program koji ispisuje sumu i prosek n unetih brojeva.
4. Ispisati program koji ispisuje kubove prirodnih brojeva do zadatog broja.
   Primer: 125 1 8 27 64 125
5. Ispisati program koji ispisuje tablicu množenja do 10 unetog broja.
   Primer: 7 7 14 21 28 35 42 49 56 63 70
6. Ispisati program koji ispisuje prvih n neparnih brojeva i njihovu sumu.
7. Ispisati program koji ispisuje tablicu množenja do n.
   Primer: 3 1 2 3
   2 4 6
   3 6 9
8. Ispisati program koji iscrtava dati šablon.
   a. *
   **
   ***
   ****
   b. 1
   12
   123
   1234
   c. 1
   22
   333
   4444
   d. 1
   2 3
   4 5 6
   7 8 9 10
   e.
   ```
      *
     * *
    * * *
   * * * *
   ```
9. Ispisati program koji ispisuje faktorijal unetog broja.
10. Ispisati program koji iscrtava šahovsku tablu.
11. Korisnik unosi brojeve sve dok ne unese nulu, nakon toga treba ispisati koliko je parnih a koliko neparnih brojeva uneo

4. Rešenja

1.

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

2.

```
n = int(input())
sum = 0
for i in range(n):
    sum += i + 1
print(sum)
```

3.

```
n = int(input())
sum = 0
for i in range(n)
    sum += i + 1
print(sum)
print(sum/n)
```

4.

```
n = int(input())
broj=1
while broj**3<=n:
    print(broj**3)
    broj+=1
```

5.

```
n=int(input())
for i in range(n,11*n,n):
    print(i)
```

6.

```
n=int(input())
for i in range(1,2*n,2):
    print(i)
```

7.

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

8.

a.

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

b.

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

c.

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

d.

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

e.

```
n=int(input())
for i in range(n):
    for j in range(n-1-i):
        print(' ',end='')
    for k in range(i+1):
        print('* ',end='')
    print()
```

9.

```
n = int(input())
fact = 1
for i in range(2,n+1):
    fact *= i
print(fact)
```

10.

```
for i in range(8):
    if i%2==0:
        par='O'
        nep='X'
    else:
        par='X'
        nep='O'
    for j in range(8):
        if j%2==0:
            print(par,end='')
        else:
            print(nep,end='')
    print()
```

11.

```
brojParnih=0
brojNeparnih=0
while True:
    n=int(input())

    if n==0:
        break

    if n%2==0:
        brojParnih+=1
    else:
        brojNeparnih+=1

print("Parnih:",brojParnih)
print("Neparnih:",brojNeparnih)
```

5. Liste

Do sada smo videli osnovne tipove podataka koje Python poseduje i verovatno smo zaključili da postoje situacije u kojima nam oni nisu dovojni. Jedna od takvih situacija su programi kod kojih je potrebno pamtiti veliki broj vrednosti, na primer program koji učitava 100 brojeva, deli ih na parne i neparne i potom ih ispisuje. Kod rešavanja takvog programa nekako bi morali da sačuvamo svih 100 brojeva (bilo pre ili posle razvrstavanja) što bi sa dosadašnjim znanjem zahtevalo ručno pravljenje 100 promenljivih što složićemo se optimalno.
Kod takvih i sličnih zadataka na scenu stupaju liste, liste su tip podataka koji predstavlja kolekciju nekih drugih podataka, tj. Jedna lista može da sadrži više podataka koji čak mogu biti i različitih tipova (*type mixing*). Ako smo promenljive zamišljali kao kutije koje imaju neko ime i unutar sebe pamte neku vrednost, onda liste možemo zamisliti kao iste te kutije samo što one sada unutar sebe imaju numerisane pregrade od kojih svaka može da čuva neku vrednost.

Kreiranje lista

Slično kao što smo kreirali promenljive dodeljivanjem neke vrednosti određenom imenu po prvi put, liste kreiramo dodeljivanjem neke liste proizvoljnom imenu. To možemo izvršiti na dva načina: dodeljivanjem prazne liste ili dodeljivanjem liste popunjene nekim vrednostima.

```
#prazna lista
lista=[]
#primer liste
lista1=[5,3,7,2,9]
```

Primeri kreiranja liste

Pristup elementima liste

Za pristup vrednostima sačuvanim u listi (u daljem tekstu elementima) potrebno je da pored imena liste specifikujemo i njihov indeks i to direkno nakon navođenja imena liste u uglastim zagradama `[ ]`. Takođe je moguće pristupiti delu liste definisanom parametrima `start`, `stop` i `step` slično kao kod naredbe `range`, takav pristup listi se zove *slicing*.

Indeksiranje elemenata u listi

Već je rečeno da je za pristup elementu u listi pored imena liste potrebno poznavati i njegov poziciju u listi tj. indeks, a sada ćemo i videti kako Python numeriše elemente u listi. Kao i većina programskih jezika Python koristi indeksiranje bazirano od nule što znači da će prvi element liste imati indeks 0 a svaki sledeći element će imati indeks veći za jedan od prethodnog. Na primer ako imamo listu od tri elementa njihovi indeksi će biti 0,1 i 2 (zgodno je uočiti da je indeks poslednjeg elementa uvek za jedan manji od broja elemenata u listi). Pored ovog načina indeksiranja Python omogućava i indeksiranje od kraja liste koristeći negativne indekse, ovog puta indeksiranje počinje od -1 (jer bi -0 bilo isto što 0) pa indeks -1

odgovara poslednjem elementu liste. Na primer ako opet imamo listu od tri elementa i želimo da je indeksiramo od kraja indeksi elemenata će biti -3,-2 i -1 (opet je zgodno uočiti da je u ovom slučaju indeks prvog elementa dužina liste pomnožena sa -1).

```
# 0  1  2  3  4
[ 5, 3, 7, 2, 9]
#-5 -4 -3 -2 -1
```

Primeri indeksiranja od početka i kraja liste

```
#pristup listi
lista1[3] # 2
lista1[-2] # 2
#pristup delu liste "slicing"
lista1[1:-1:1] #[3,7,2]
#high nije ukljucen
```

Primeri pristupa elementima liste

Korisne operacije nad listama

<vrednost> in <lista>	Provera da li se vrednost nalazi u listi
<vrednost> not in <lista> Ili not <vrednost> in <lista>	Provera da li se vrednost ne nalazi u listi
<lista>.index(<vrednost>)	Pronalaženje indeksa prvog pojavljivanja vrednosti
len(<lista>)	Dužina liste
<lista>*<vrednost>	Ponavljanje liste vrednost puta
<lista>+<lista>	Nadovezivanje dve liste
<lista>+=<lista>	Nadovezivanje druge liste na prvu
<lista>.append(<vrednost>)	Dodavanje vrednosti na kraj liste
<lista>.insert(<indeks>,<vrednost>)	Dodavanje vrednosti na zadati indeks uz pomeranje ostalih elemenata
del <lista>[<indeks>]	Brisanje elementa na datom indeksu uz pomeranje ostalih elemenata
<lista>.remove(<vrednost>)	Brisanje prvog pojavljivanja date vrednosti uz pomeranje ostalih elemenata
sum(<lista>)	Suma elemenata liste
min(<lista>)	Minimalni element liste

max(<lista>)	Maksimalni element liste
<lista>.reverse()	Obrtanje redosleda elemenata liste
<lista>.sort()	Sortiranje liste rastuće
<lista>.count(<vrednost>)	Broj pojavljivanja vrednosti u listi

```
lista=[5,3,7,2,9]
#Provera da li se vrednost nalazi u listi
5 in lista # True
12 in lista # False
#Provera da li se vrednost ne nalazi u listi
5 not in lista # False
not 12 in lista # True
#Pronalaženje indeksa prvog pojavljivanja vrednosti
lista.index(7) # 2
#Dužina liste
len(lista) # 5
#Ponavljanje liste vrednost puta
[0]*3 #[0,0,0]
[1,2]*2 #[1,2,1,2]
#Nadovezivanje dve liste
[1,2]+[3] #[1,2,3]
#Nadovezivanje druge liste na prvu
lista+[3] # lista=[5,3,7,2,9,3]
#Dodavanje vrednosti na kraj liste
lista.append(7) # [5,3,7,2,9,3,7]
[1,2,3].append([4]) # [1,2,3,[4]]
#Dodavanje vrednosti na zadati indeks uz pomeranje ostalih elemenata
lista.insert(1,1) # [5,1,3,7,2,9,3,7]
#Brisanje elementa na datom indeksu uz pomeranje ostalih elemenata
del lista[2] # [5,1,7,2,9,3,7]
#Brisanje prvog pojavljivanja date vrednosti uz pomeranje ostalih
elemenata
lista.remove(5) # [1,7,2,9,3,7]
#Suma elemenata liste
sum(lista) # 29
#Minimalni element liste
min(lista) # 1
#Maksimalni element liste
max(lista) # 9
#Obrtanje redosleda elemenata liste
lista.reverse() # [7,3,9,2,7,1]
#Sortiranje liste rastuće
lista.sort() # [1,2,3,7,7,9]
```

```
#Broj pojavljivanja vrednosti u listi
lista.count(7) # 2
lista.count(12) # 0
```

Primeri korisnih operacija nad listama

Prolazak kroz elemente liste

Kada radimo sa listama često je potrebno da prođemo kroz sve elemente liste i uradimo nešto sa njima, za to nam se kao logično rešenje nude petlje a pogotovo for petlja. Ukoliko želimo da prođemo kroz listu koristeći for petlju, to možemo da izvedemo na dva načina koji se razlikuju po tome da li nam je dozvoljeno da menjamo elemente liste. Ukoliko želimo da menjamo elemente liste koristimo for petlju kao što smo je i do sada koristili sa sekvencom koja će proći kroz sve indekse u nizu. A ukoliko samo želimo da prođemo kroz elemente liste bez menjanja njihovih vrednosti onda možemo iskoristiti samu listu kao sekvencu za for petlju, u tom slučaju naš iterator će redom uzimati vrednosti elemenata liste.

```
lista=[5,3,7,2,9]
#prolazak kroz listu sa menjanjem elemenata liste
for i in range(len(lista)):
      print(lista[i])
      lista[i]+=1

#prolazak kroz listu bez menjanja elemenata liste
for i in lista:
      print(i)
```

Primer prolazka (iteracije) kroz elemente liste

5. Primeri

1. Učitava se lista proizvoljne dužine koju određuje korisnik, nakon toga se unose dva broja x i y, potrebno je sva pojavljivanja broja x u listi zameniti brojem y a potom ispisati listu

```
#korisnik unese prvo broj elemenata liste pa zatim i elemente liste
n=int(input())
lista=[]
for i in range(n):
    broj=int(input())
    lista.append(broj)
x=int(input())
y=int(input())
while x in lista:
    i=lista.index(x)
    lista[i]=y
print(lista)
```

2. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je ispisati sve podnizove dužine 3 koji sadrže minimum te liste

```
n=int(input())
lista=[]
for i in range(n):
    broj=int(input())
    lista.append(broj)

minimum=min(lista)
for i in range(len(lista)-2):
    podniz=lista[i:i+3]
    if minimum in podniz:
        print(podniz)
```

5. Zadatci

1. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je izračunati sumu elemenata liste bez korišćenja funkcije sum
    a. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je izračunati proizvod elemenata liste
2. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je izračunati minimalni element liste bez korišćenja funkcije min
3. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je izračunati indeks prvog pojavljivanja traženog elementa bez korišćenja funkcije index
    a. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je izračunati broj pojavljivanja traženog elementa bez korišćenja funkcije count
4. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je dodati traženi element na određenu poziciju bez korišćenja funkcije insert
5. Učitava se lista proizvoljne dužine koju određuje korisnik, potrebno je obrnuti listu bez korišćenja funkcije reverse
6. Učitava se lista tekstova proizvoljne dužine koju određuje korisnik, potrebno je izračunati broj tekstova dužine veće od 2 kojima su prvo i poslednje slovo isto.
7. Napisati program koji uklanja duplikate iz liste tako što uklanja svako sem prvog ponavljanja određenog broja
8. Napisati program koji proverava da li je lista prazna
9. Napisati program koji kopira listu
10. Napisati program koji iz liste reči ispisuje reči duže od x (x se unosi)
11. Napisati program koji proverava da li dve unete liste imaju zajedničke elemente

5. Rešenja