Python Development

Funkcije u Python-u

POWERED BY COMTRADE

Problemi kojima smo se do sada bavili

- Dodavanje / popunjavanje elemenata liste
- Traženje sume elemenata liste
- Prebrojavanje elemenata liste
- Traženje proizvoda elemenata liste
- Računanje proseka
- Traženje minimuma/maksimuma
- Korišćenje “flag”-a
- Izdvajanje cifara broja
- Filtriranje liste

Unos liste sa for petljom

```
duzina_liste = int(input("Unesite duzinu liste:"))
brojevi = []

for i in range(duzina_liste):
        korisnikov_unos = int(input("Unesite {}. element liste.".format(i+1)))
        brojevi.append(korisnikov_unos)
```

Unos liste sa while petljom

```
duzina_liste = int(input("Unesite duzinu liste:"))
brojevi = []
brojac = 0
while brojac<duzina_liste:
        korisnikov_unos = int(input("Unesite {}. element liste.".format(i+1)))
        brojevi.append(korisnikov_unos)
        brojac+=1
```

Sumiranje elemanata liste

Kada sumiramo elemente liste potrebna nam je neka promenljiva u kojoj ćemo čuvati privremeni rezultat koja je na početku jednaka 0.

```
suma = 0

for broj in brojevi:
        #if USLOV: ako broj ispunjava uslov iz zadatka onda na sumu trenutnu dodajemo broj
        suma += broj
print(suma)
```

Prebrojavanje elemenata liste

Kada sumiramo elemente liste potrebna nam je neka promenljiva u kojoj ćemo čuvati privremeni rezultat koja je na početku jednaka 0.

```
brojac = 0

for broj in brojevi:
        #if USLOV: ako broj ispunjava uslov iz zadatka onda na sumu trenutnu dodajemo broj
        brojac += 1 # brojac se uvećava za 1 uvek!
print(suma)
```

Proizvod elemenata liste

Kada tražimo proizvod elemente liste potrebna nam je neka promenljiva u kojoj ćemo čuvati privremeni rezultat koja je na početku jednaka 1

```
proizvod = 1
for broj in brojevi:
        #if USLOV: ako broj ispunjava uslov iz zadatka onda na sumu trenutnu dodajemo broj
        proizvod *= broj
print(proizvod)
```

Traženje proseka

Za prosek su nam potrebni suma i broj elemenata.

```
suma = 0
brojac = 0

for broj in brojevi:
    if USLOV:
        suma += broj
        brojac += 1
prosek = suma/brojac
```

Traženje minimuma i maksimum

```
brojevi=[3,5,6,1,2,3]
trenutni_maksimum = brojevi[0] #postavimo trenutni maksimum ili minimum na prvi ele
ment u listi
for broj in brojevi:#prodjemo kroz list
    if broj >trenutni_maksimum:#pitamo da li je neki element veci od  maksimuma
        trenutni_maksimum=broj#menjamo trenutno maksimum
```

Rad sa flagovima

Na početku flag možemo da postavimo na True,a zatim tražimo makar 1 slučaj koji nam narušava našu teoriju. Npr da li je broj prost?

```
prost = True
broj = int(input("Unesite broj:"))
for i in range(2,broj):
        if broj % i == 0: # ako je deljiv sa makar jos jednim brojem
                prost = False # nije prost
                break # nema potrebe da dalje proveravamo sledece vrednosti
if prost:
        print("{} je prost. ".format(broj))
else:
        print("{} nije prost. ".format(broj))
```

Izdvajanje cifara broja

```
broj =1234
while broj >0:
    poslednja_cifra= broj %10 # izdvajamo poslednju cifru trenutne vrednosti broja
    broj = broj // 10 # "skracujemo" broj za jednu cifru, da bismo bili spremni za izdvajanje
 sledece cifre
    # dodatna manipulacija sa ciframa
```

Filtriranje liste

```
brojevi=[1,2,3,4,5,6]
parni = []
for broj in brojevi:#prolazak kroz listu brojeva
    if broj % 2 == 0: # provera uslova(u ovom slucaju zelimo samo parne brojeve)
        parni.append(broj) # dodavanje tog broja u listu
```

Spisak metoda listi

- naziv_liste.**append**(vrednost) - dodaje vrednost na krajliste
- naziv_liste.**extend**(druga_lista) - dodaje celu listu na kraj
- naziv_liste.**remove**(vrednost) - briše prvo pojavljivanje vrednosti izliste
- naziv_liste.**pop**(pozicija) - briše element iz liste na osnovupozicije
- naziv_liste.**pop**() –briše element sa poslednje pozicije
- naziv_liste.**clear**() - briše sve elemente iz liste

Nastavak

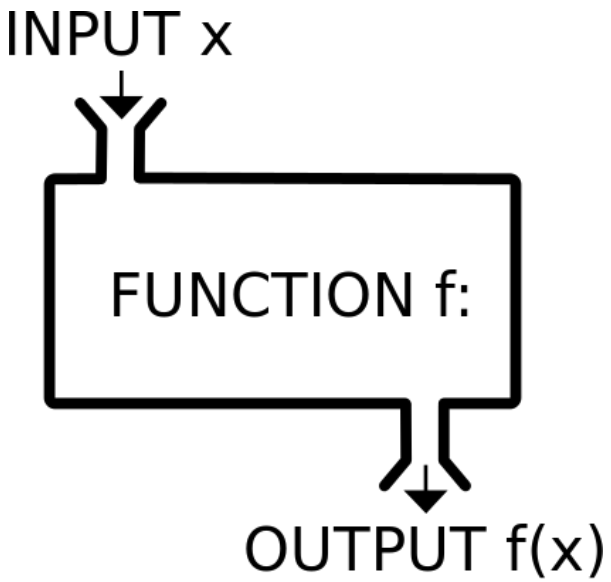
- naziv_liste.**insert**(pozicija,vrednost) - na poziciju u listi postavljavrednost

- naziv_liste.**index**(vrednost) - vraća poziciju na kojoj se pojavljuje po prvi put u listi vrednost
- naziv_liste.**sort**() – sortira samu listu

- naziv_liste.**reverse**() – okreće redosled elemenata u listi unazad

Funkcije

- Veza između inputa i outputa
- Ponovna upotreba koda (*reusabillity*)
- Grupa komandi koja su zadužena za izvršavanje specifičnog zadatka



Funkcije

- Kao i promenljive funkcije moraju prvo biti napravljene i imenovane da bismo mogli da ih koristimo.
- Pravila za imena funkcija u pythonu su ista kao i pravila za promenljive

1. Ne smeju se koristiti rezervisane reči
2. U nazivu ne sme biti razmaka(koristimo ili camelCase ili notaciju sa _)
3. Prvi karakter u nazivu ne sme biti cifra
4. Karakteri koje sadrži mogu biti od a-z A-Z 0-9 _
5. Nazivi funkcija su case sensitive

Deklaracija funkcije i poziv funcije

```
def naziv_funkcije(): # deklaracija funkcije
        # komande unutar funkcije

naziv_funkcije() # poziv funkcije
```

Nakon što se završi izvršavanje funkcije kod se nastavlja od sledećeg reda nakon poziva funkcije.

Argumenti sa ključnim rečima

U slučaju da koristimo samo argumente sa ključnim rečima,redosled argumenata nije bitan.

naziv_funkcije(argument1 = vrednost1, argument2=vrednost2)

naziv_funkcije(argument2 = vrednost2, argument1 = vrednost1)

Poziv sa vrednosnim i ključnim argumentima

U slučaju da pozivamo funkciju sa vrednosnim i ključnim argumentom, potrebno je da prvo navedeno vrednosne argumente, a zatim argumente sa ključnim vrednostima.

naziv_funkcije(vrednost1, argument2 = vrednost2)

naziv_funkcije(argument1 = vrednost1, vrednost2)

Syntax error: Positional argument follows keyword argument

Podrazumevani parametri (default parameters)

def naziv_funkcije(argument1,argument2 = podrazumevana_vrednost):

Funkcije može biti pozvana sa ili bez ovog parametra.
Validni pozivi funkcije

- naziv_funkcije(argument1)
- U slučaju da nije zapisan drugi argument funkcija se izvršava sa podrazumevanom vrednosti

- naziv_funkcije(argument1,argument2)
- Ako dodelimo vrednost drugom argumentu, funkciji će biti prosleđena vrednost i za argument2.

Redosled argumenata

def naziv_funkcije(argument1 = podrazumevana_vrednost,argument2)

SyntaxError: non-default argument follows default argument

U slučaju da funkcija ima podrazumevani argument, nakon njega među argumentima mogu biti samo argumenti koji imaju podrazumevanu vrednost

Funkcije sa nedefinisanim brojem argumenata

U slučaju da se za neku funkciju ne zna unapred sa koliko argumenata će se funkcija izvršiti, tj. ako funkcija može imati proizvoljan broj argumenata,argument počinje sa *

```
def naziv_funkcije(*argument):
        for x in argument:
                print(x)

naziv_funkcije("Dusan","Sijacic","Marko","Jovan")
```

Funkcije sa povratnom vrednosti

U slučaju da funkcija vraća neku vrednost kao rezultat izvršavanja, tada koristimo ključnu reč return

```
def naziv_funkcije(argument1,argument2):
        return argument1 + argument2
```

U slučaju da funkcija ima neku povratnu vrednost, možemo rezultat sačuvati u neku promenljivu.

```
naziv_funkcije(vrednost1, vrednost2)

rezultat = naziv_funkcije(vrednost1, vrednost2)
```

Poziv drugih funkcija unutar funkcije

Unutar jedne funkcije mogu biti pozvane druge funkcije.

```
def foo():
        print("foo")
        bar()

def bar():
        print("bar")

foo()
```

Rekurzivne funkcije

Rekurzivne funkcije su funkcije koje pozivaju same sebe. Za svaku rekurzivnu funkciju nam je potreban kritičan slučaj u kojem se rekurzija prekida, da ne bismo ušli u “beskonačnu petlju”

- Faktorijel
- N-ti element fibonaccijevog niza

Sortiranje

- Često je potrebno sortirati listu
- list.sort() vs sorted()
- Numeričko i leksikografsko sortiranje
- Kodne šeme ASCII, UNICODE, UTF-8 i njihova binarna reprezentacija
- ord() funkcija

for petlja i **else**

- Kod **for** petlje, **else** se izvršava ako se **for** petlja završi, tj. ne naiđemo na **break**

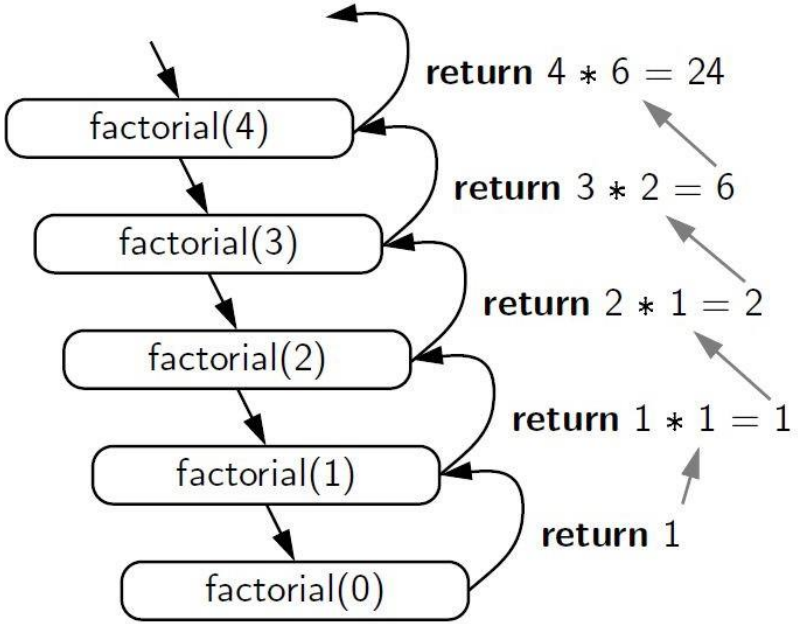
1. *kandidat_broj* = **int(input())**
2. **for** *broj* **in range(2,** *kandidat_broj***):**
3. **if** *kandidat_broj* **%** *broj* **== 0:**

4.print(f'{*kandidat_broj***}** nije prost**')** 05

break

6. **else:**
7. **print(f'{***kandidat_broj***}** je prost**')**

Rekurzivni faktorijal




```
1. def fakt_rek(broj):
2.     if broj == 1:
3.         return broj
4.     else:
5.         return broj * fakt_rek(broj - 1)
```

List comprehension

- Kraći zapis generisanja liste koristeći neku drugu kolekciju/iterator
- Doseg promenljivih u list comperhension-ima

```
01 brojevi = [1, 2, 3, 4, 5]
02
3.  # brojevi_str = []
4.  # for broj in brojevi:
5.  #     brojevi_str.append(str(broj))
06
07 brojevi_str = [str(broj) for broj in brojevi]
```

Zadaci za vežbanje (1):

Napisati rekurzivnu funkciju koja prikazuje sve cifre datog celog pozitivnog broja i to:

- ispis1(x) koja ispisuje s leva na desno
- ispis1_drugi_nacin(x) koji vraća listu cifara koje su poređane s leva na desno
- ispis2(x) koja ispisuje s desna na levo
- ispis2_drugi_nacin(x) koji vraća listu cifara koje su poređane s desna na levo

Primer 1:
a)
34021

3 4 0 2 1

Primer 2:
b)
56219

9 1 2 6 5

Zadaci za vežbanje (2):

Napisati rekurzivnu funkciju broj_parnih(x) koja računa broj parnih cifara datog celog broja x.

Zadaci za vežbanje (3):

Napisati rekurzivnu funkciju najveca_cifra(x) koja racuna najvecu cifru datog celog broja x.

Zadaci za vežbanje (4):

Napisati rekurzivnu funkciju ukloni(x, c) koja uklanja sva pojavljivanja date cifre c iz datog broja x. Zadatak se može uraditi uz pomoć listi.

Zadaci za vežbanje (5):

Napisati rekurzivnu funkciju napravi_niz(broj) koja kreira niz cifara datog celog broja. Napisati rekurzivnu funkciju ispisi_niz(niz, index) koja ispisuje elemente niza dužine n. Testirati obe funkcije pozivom iz glavnog programa.

Zadaci za vežbanje (6):

Napisati rekurzivnu funkciju obrni(x) koja obrće cifre datog celog broja x. Zadatak se može rešiti korišćenjem listi.

Zadaci za vežbanje (7):

Napisati rekurzivnu funkciju obrni_niz(niz, index) koja obrće niz brojeva.

Zadaci za vežbanje (8):

Napisati rekurzivnu funkciju palindrom(niz) koja ispituje da li su elementi nekog niza brojeva poređani palindromski (isto napred i od pozadi).

Zadaci za vežbanje (9):

Napisati rekurzivnu funkciju dodaj_nulu(x) koja posle svake neparne cifre datog broja dodaje 0.