Увод у рачунарство 1

Александар Картељ

kartelj@matf.bg.ac.rs

Функције

Увод, декларације, пренос аргумената, ...

Декларација и позив функције

```
# definicija funkcije
# def naziv_funkcije(lista_argumenata):
# naredbe...
# komanda return u slucaju da postoji povratna vrednost
def kvadrat(x):
      kv = x*x
      return kv

# poziv funkcije za konkretne argumente
y = kvadrat(5)
print(y)
```

Функције - мотивација

```
def kvadrat(x):
    kv = x*x
    return kv
```

- Зашто користимо функције?
  - Елиминација вишеструког писања истог кода
  - Када утврдимо да се нешто учестало користи, као нпр. неке математичке функције, има смисла те функционалности описати функцијом

Аргументи функција и локалне променљиве

```
def kvadrat(x):
    kv = x*x
    return kv
```

Променљиве **x** и **kv** имају само локални досег, тј. не виде се ван тела функције **kvadrat**.

```
# poziv funkcije za konkretne argumente
y = kvadrat(5)
print(y)
```

Пренос аргумената у функцију

```
def kvadrat_pogresan(x, kv):
        kv = x*x

y = 0
kvadrat_pogresan(4,y)
# y se prosledjuje po vrednosti, sto znaci da se pravi
# nova lokalna promenljiva u okviru tela funkcije kvadrat_pogresan
# tako da save sto se desava unutar nje, nema efekta na y
print(y)
```

Пример

- Написати функцију која проверава да ли је прослеђени број прост

```
def prost(b):
    for i in range(2,b):
        if b%i==0:
            return False
    return True
```

Пример (2)

- Написати функцију која враћа листу свих простих бројева до броја N при чему се N задаје као аргумент те функције

```
def prostiN(N):
        # l je lokalna promenljiva koju cemo vratiti sa return
        l = []
        for i in range(2,N+1):
                if prost(i):
                        l.append(i)
        return l
x = int(input())
lista = prostiN(x)
print(lista)
```

Пример (3)

- Написати функцију која за задата два броја рачуна најмањи заједнички делилац

```
def nzd(a, b):
      while a!=b:
            if a>b:
                  a = a-b
            else:
                  b = b-a
      return a
n = nzd(48, 36)
print(n)
```

Задаци

- Написати функцију која проверава да ли је година преступна (година је преступна ако је дељива са 4, при чему свака која је дељива са 100 није преступна осим оних које су дељиве са 400)

- Написати функцију која за задати датум у форми (дан у месецу, редни број месеца и година) проверава да ли је могућ

- Написати функцију проверава да ли се задата текст налази унутар другог задатог текста (у случају да се налази, враћа индекс почетка, а у супротном враћа -1)

Рекурзија

Рекурзија

- Као што смо видели у примеру са листом простих бројева, функција може да позива другу функцију
- Испоставља се да функција може да позива и **саму себе!**
- Тај процес се назива **рекурзија**

Пример

- Написати функцију која исписује бројеве почев од **n** до 1

```
def izbrojiUnazadRek(n):
        # izlaz iz rekurzije obavezno na pocetak
        if n==0:
                return # ovo znaci samo izlazak iz funkcije, ne vracamo nista
        print(n)
        # rekurzivni korak (pozivanje rekurzije za drugi argumenat)
        izbrojiUnazadRek(n-1)

izbrojiUnazadRek(17)
```

Пример (2)

- Написати функцију која сабира све бројеве од 1 до N рекурзивно при чему се N задаје као аргумент функције

```
def sumiranjeRek(n):
    if n==1:
        return 1
    return n+sumiranjeRek(n-1)

n = int(input())
print(sumiranjeRek(n))
```

Главни елементи рекурзије

- Како можемо описати претка неке особе.
  Један могући начин је коришћењем следеће две дефиниције:
  1. Родитељ те особе је њен предак (базни случај)
  2. Родитељ било ког претка неке особе је такође предак те особе (рекурзивни корак)
- На пример, отац се може добити директно применом прве дефиниције
- Деда се може добити најпре применом правила два па потом и правила 1: мој деда је родитељ мог оца, а мој отац је мој родитељ

Главни елементи рекурзије (2)

- Два главна елемента рекурзије су:
  1. База рекурзије – односно дефиниција којом се излази из рекурзије
  2. Рекурзивни корак – дефиниција којом се проблем своди на нови рекурзивни позив
- Код проблема сумирања бројева од 1 до n:
  1. База рекурзије: када је аргумент функције 1
     тада се само враћа вредност 1 и излази из рекурзије
  2. Рекурзивни корак: када је аргумент функције n>1
     тада се позива рекурзија за n-1 и на добијени резултат се додаје број n

Пример (3)

- Написати рекурзивну функцију која прихвата два броја **x** и **n** (реалан и цео број) и рачуна вредност $\mathbf{x^n}$

```
def stepen(x, n):
        if n == 0:
                return 1
        return x*stepen(x,n-1)

def stepen_brzi(x, n):
        if n==0:
                return 1
        if n%2==0:
                pom = stepen_brzi(x,n/2)
                return pom*pom
        else:
                # celobrojno deljenje
                pom = stepen_brzi(x, n//2)
                return pom*pom*x
```

Пример (4)

- Написати функцију која рачуна најмањи заједнички делилац рекурзивно

Задаци

- Написати рекурзивну функцију која рачуна факторијел броја
- Написати рекурзивну функцију која рачуна N-ти фибоначијев број
- Написати рекурзивну функцију која обрће листу вредности
- Написати рекурзивну функцију која из задате листе брише све бројеве који су непарни
- Написати рекурзивну функцију која генерише паскалов троугао

Материјал за читање

- За боље разумевање рекурзије прочитати Главу 5 из http://poincare.matf.bg.ac.rs/~janicic//courses/p2.pdf