PROGRAMIRANJE 1

PRIRODOSLOVNO-MATEMATIČKI FAKULTET
SVEUČILIŠTE U SPLITU

python™

- Postoje brojni razlozi i prednosti korištenja procedura:
  - Smanjiti ponavljanja (redundancije) istih ili sličnih sekvenci programskog koda.
  - Omogućiti ponovnu uporabu dijelova programskog koda u različitim programima.
  - Dekompozicija složenih problema u manje i jednostavnije cjeline.
  - Poboljšanje "čitljivosti" i preglednosti programa.
  - Repliciranje rješenja za matematičke funkcije.
  - Sakrivanje informacija i nevažnih detalja unutar pojedinih dijelova programa.
  - Smanjenje rizika od grešaka, lakše održavanje i nadogradnja velikih sustava.

- Korištenje procedura omogućuje bolji uvid u tok programa, tako da je pažnju lakše posvetiti problemu „što radi" za razliku od "kako to radi".
- Koristi se određena forma naredbe poziva (eng. call).
- U trenutku izvođenja naredbe poziva procedure, slijed (kontrola) izvođenja naredbi predaje se pozvanoj cjelini, odnosno prelazi se na izvršavanje naredbi procedure.
- Nakon što se izvedu sve naredbe pozvanog procedure, kontrola se vraća "pozivaču" i izvršavaju se dalje njegove instrukcije.
- Procedure mogu biti pozvane iz glavnog programa, ali i same procedure mogu pozivati druge procedure.

- Osnovne karakteristike procedura:
  - imaju jednu ulaznu točku
  - pozivajuća procedura se zaustavlja u toku izvršavanja pozvane procedure,
  - samo jedna procedura se izvršava u danom vremenu
  - kontrola se uvijek vraća pozivajućoj proceduri po završetku izvršavanja određene procedure

- Uobičajena je podjela procedura na dvije osnovne vrste:
  - **Potprograme** (eng. subroutine) koji imaju od nula do više ulaznih parametara i ne vraćaju rezultat
    - Pozivaju se najčešće posebnom naredbom iza koje slijedi ime potprograma i lista ulaznih i izlaznih parametara.
  - **Funkcije** (eng. function) koje imaju od nula do više ulaznih parametara i vraćaju rezultat
    - Funkcije se pozivaju kao dio izraza, npr. A = funkcija(B).
    - Pojam "vraćanja" vrijednosti zapravo je izračunavanje vrijednosti funkcije koja se dalje tretira kao i ostale varijable u izrazu.

- Python: potprogrami i funkcije: def

```
def <ime_funkcije>(<parametri>):
        <tijelo_funkcije>
```

- Dio programa u kojem se kreira funkcija se naziva: definicija funkcije

```
def <ime_funkcije>(<parametri>):
        <tijelo_funkcije>
        return value
```

- Funkcija može vraćati vrijednosti – tada koristimo naredbu **return**
  - naredba **return** može se pojaviti bilo gdje u tijelu funkcije
  - završava se izvršavanje funkcije i vraća rezultat pozivatelju

```
#definicija funkcije
def Pozdrav():
    print("Dobar dan!")

#glavni dio programa
for i in range(5):
    Pozdrav()
```

Ova funkcija ne prima ništa, ne vraća ništa

```
#definicija funkcije
def Pozdrav2(osoba):
    print("Dobar dan",osoba,"!")

#glavni dio programa
ime = input("Kako se zoves? ")
for i in range(5):
    Pozdrav2(ime)
```

Ova funkcija prima parametar, ali ne vraća ništa!

```
def Kvadrat(broj):
    return broj*broj

x = 5
print("Kvadrat broja: ", Kvadrat(x))
```

Ova funkcija prima broj, a vraća kvadrat tog broja!

```
def Kvadrat(broj):
    return broj*broj

for i in range(1,11):
    print("Kvadrat broja: ", Kvadrat(i))
```

```
def Pravokutnik(a,b):
    op = 2*(a+b)
    p = a*b
    return op, p

a = 2
b = 3
opseg,povrsina = Pravokutnik(a,b)
print("opseg: ", opseg)
print("povrsina: ", povrsina)
```

Ova funkcija prima i vraća dvije vrijednosti!

- Procedure su zasebne programske jedinice koje imaju svoje varijable.
- Jedna procedura "ne zna" za varijable druge, odnosno nema direktnog pristupa do adresa memorijskih lokacija na kojima su varijable drugih procedura.
- Stoga pri pozivu procedura pozvanoj cjelini treba "prenijeti" ulazne podatke (parametri), a pri završetku pozvana cjelina treba "vratiti" rezultate programu koji ga je pozvao.

- **Globalne varijable** se definiraju izvan procedure i žive od trenutka svoga postanka pa sve do kraja programskog koda (i u procedurama).
- **Lokalne varijable** su varijable definirane unutar neke procedure i njihov “život" je ograničen na tu proceduru.

  Svaka lokalna varijabla postoji od trenutka svoga nastanka, pa sve do kraja odgovarajučeg bloka naredbi.
- Lokalne varijable su one koje se koriste u određenom dijelu koda (nekoj funkciji, primjerice) i važeće su samo za taj dio koda.
- One mogu imati identično ime kao varijable u nekom drugom dijelu koda, ali bez obzira na to Python ih neće tretirati kao jednu te istu varijablu, već kao različite, svaku u svom opsegu djelovanja (tzv. scope)

```
def funkcija():
    x = 'Blabla‘
    print (x)
x = 10
print ("Ispis globalne varijable x")
print (x)
print ("\nIspis varijable x unutar funkcije")
funkcija()
```

Rezultat izvršavanja programa je ovo:

Ispis globalne varijable x
10
Ispis varijable x unutar funkcije
Blabla

- Uočite kako je u definiciji funkcije dodan redak u kojem piše "global x" – to je znak Pythonu da varijablu ne tretira kao lokalnu, već da radi sa originalnom varijablom x.

```
def funkcija():
    global x
    x = 'Blabla‘
    print (x)
x = 10
print ("Ispis varijable x ")
print (x)
print ("\nIspis varijable x unutar funkcije")
funkcija()
print ("\nPonovni ispis varijable x nakon povratka iz funkcije")
print (x)
```

Ispis je ovakav:

Ispis varijable x

10

Ispis varijable x unutar funkcije

Blabla

Ponovni ispis varijable x nakon povratka iz funkcije

Blabla

```
def funkcija1():
    b = 2
    print (a,b,c)
    funkcija2()

def funkcija2():
    c = 3
    print (a,b,c)

a = 1
funkcija1()
print (a,b,c)
```

```
def funkcija1():
    b = 2
    print (a,b,c)
    funkcija2()

def funkcija2():
    c = 3
    print (a,b,c)

a = 1
funkcija1()
print (a,b,c)
```

Varijabla a je dostupna funkciji1 jer je nastala u glavnom programu.

Varijabla c nije dostupna funkciji1 jer je dio podfunkcije (funkcija2) koja nastaje iz te funkcije, pa će izvršavanje ovog koda prijaviti grešku.

```
Traceback (most recent call last):
  File "C:/Python352/a.py", line 11, in <module>
    funkcija1()
  File "C:/Python352/a.py", line 3, in funkcija1
    print (a,b,c)
NameError: name 'c' is not defined
```

Uklonite li tu grešku brisanjem varijable c u prvoj print() naredbi, program će se izvršiti do zadnje naredbe print() i ponovo prijaviti grešku, jer niti varijabla b niti varijabla c nisu dostupne glavnom programu, one su definirane kao lokalne u navedenim podfunkcijama i zato nedostupne glavnom programu; varijabla a pritom je dostupna svima.

```
1 2
Traceback (most recent call last):
  File "C:/Python352/a.py", line 11, in <module>
    funkcija1()
  File "C:/Python352/a.py", line 4, in funkcija1
    funkcija2()
  File "C:/Python352/a.py", line 8, in funkcija2
    print (a,b,c)
NameError: name 'b' is not defined
```

```
def funkcija1():
    b = 2
    print (a,b)
    funkcija2(b)

def funkcija2(b):
    c = 3
    print (a,b,c)

a = 1
funkcija1()
```

Varijabla b nije dostupna funkciji2, pa je treba prenjeti kao ulazni parametar funkciji2.

- Upisati riječ. Ispisati tu riječ na ekran samo ako joj je suma znamenki duljine te riječi prost broj.

```
#unos riječi
rijec=input("Upiši riječ")
print("Suma znamenki suljine riječi ", rijec, " je ", sumaznam(len(rijec)))
if prosti( sumaznam(len(rijec))==1:
    print("Riječ ", rijec, " ima duljinu koja je prost broj")
else:
    print("Riječ ", rijec, " ima duljinu koja nije prost broj")
```

```
def sumaznam(n):
    suma=0
    while n>0:
        znam=n%10
        suma=suma+znam
        n=n//10
    return suma
```

```
def prosti(n):
    f=0
    for i in range(2,n):
        if n%i==0:
            f=1
            break
    if f==0:
        return 1
    else:
        return 0
```


python™