PRAKTIKUM IZ PROGRAMIRANJA 1

VEŽBE 4

Milica Vasović, Jana Brzaković, Ana Vidosavljević

Definisanje korisničkih funkcija

- Python pruža mogućnost definisanja korisničkih funkcija

```
def ime_funkcije(parametri_funkcije) :
    telo funkcije
```

- Funkcija mora biti definisana pre prvog korišćenja(poziva)

```
def print_lyrics():
    print ("I'm a lumberjack, and I'm okay.")
    print ("I sleep all night and I work all day.")

print_lyrics()

print (print_lyrics)
print (type(print_lyrics))
```

Poziv funkcije iz funkcije

- U telu funkcije može se naći poziv ranije definisane funkcije

```
def repeat_lyrics():
    print_lyrics()
    print_lyrics()


repeat_lyrics()
```

```
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
```

Funkcije sa parametrima

- Funkcija može biti definisana sa jednim ili više parametara
- Poziv funkcije mora odgovarati definiciji funkcije po broju i redosledu parametara

```
def print_twice(param):
    print (param)
    print (param)

print_twice('Spam')
print_twice(17)
print_twice(math.pi)
print_twice('Spam'*4)
michael = 'Eric, the half a bee.'
print_twice(michael)
```

```
Spam
Spam
17
17
3.141592653589793
3.141592653589793
SpamSpamSpamSpam
SpamSpamSpamSpam
Eric, the half a bee.
Eric, the half a bee.
```

- Promenljive definisane i korišćene unutar funkcije nisu vidljive van funkcije

```
def saberi(a,b):
      c = a + b
      print (c)

saberi(3,5)   8
print(c)      NameError: name 'c' is not defined
```

- Promenljive definisane i korišćene unutar funkcije nisu vidljive van funkcije

```
c = 10
def saberi(a,b):
    c = a + b
    print (c)

saberi(3,5)  8
print(c)     10
```

VRAĆANJE VREDNOSTI IZ FUNKCIJE

- Funkcija kao rezultat može da vrati jednu vrednost

```
a = saberi (12,26)   38
print (a)     None

def saberi2 (a,b):
    c = a + b
    return c

saberi2 (3,5)
a = saberi2 (12,26)
print (a)  38
```

- U nekim situacijama funkcija treba da vrati više vrednosti. Na primer, želimo da pretvorimo centimetre u metre i preostale centimetre.

```
def cm_u_mcm(cm):
    return (cm // 100, cm % 100)

(m, cm) = cm_u_mcm(178)
print(178, "cm", "=", m, "m", "i", cm, "cm")
```

- Osnovna plata jednog radnika u prvom mesecu bila je 48.375 dinara. U narednom mesecu je povećana za 10%, zatim je u narednom mesecu smanjena za 5%, a zatim je u narednom mesecu opet povećana za 15%. Prilikom svake promene, plata je zaokruživana na najbliži ceo broj dinara. Koliko je iznosila plata tog radnika u četvrtom mesecu?

```
def promena_plate(plata, procenat):
    return round(plata * (1+procenat))

plata1 = 48375
plata2 = promena_plate(plata1, 0.1)
plata3 = promena_plate(plata2, -0.05)
plata4 = promena_plate(plata3, 0.15)
print(plata4)
```

- Definiši funkciju kojom se vrši skraćivanje datog razlomka.

  Pre toga definisati funkciju koja određuje NZD dva broja.

```
def nzd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def skrati(a, b):
    return (a // nzd(a,b), b // nzd(a,b))


print(skrati(12,20))
```

- Napisati funkciju kojom se dati prirodan broj rastavlja na proste faktore. Na primer, za 28 računar će štampati:

  2 2 7

```
def ispisiProsteFaktore(n):
    faktor = 2

    while faktor <= n:
        if n % faktor == 0:
            print(faktor, end = ' ')
            n /= faktor
        else:
            faktor += 1

n = int(input("Unesite prirodni broj: "))
ispisiProsteFaktore(n)
```

- Napisati program kojim se određuju i ispisuju svi savršeni brojevi **od 2 do m**. Definisati funkciju koja određuje da li je prosleđeni broj savršen. Broj je savršen ako je jednak sumi svojih delitelja isključujući njega samog. Na primer, 28 je savršen broj, jer je: 28 = 1 + 2 + 4 + 7 + 14.

```
def daLiJeSavrsen(n):
    suma = 1
    for i in range(2, n//2 + 1):
        if (n % i == 0):
            suma += i
    if (suma == n):
        return True
    return False

m = int(input("Unesi broj:"))

for i in range(2, m + 1):
    if (daLiJeSavrsen(i)):
        print(i)
```

- Pravougaonik, čije su strane paralelne koordinatnim osama, zadat je koordinatama donjeg levog i gornjeg desnog ugla. Za dva data pravougaonika Pr1 i Pr2, bez obzira na njihov raspored izračunati površinu zajedničkog pravougaonika (ako ga ima).

- Napisati program kojim se za prirodne brojeve **n** i **x** izračunava suma prvih n sabiraka:

$$S = \sum_{i=1}^{n} \frac{i^i + x^i + (x+i)!}{(1+2+3+\cdots+i)+(1+2+\cdots+n)}$$

- Sa istog startnog mesta kreću dva automobila. Prvi se kreće brzinom v1 m/s, drugi v2 m/s pri čemu je v2 > v1. Drugi automobil polazi t minuta posle prvog. Napisati program kojim se određuje posle koliko minuta od polaska drugi automobil sustiže prvi.

- Formirati i ispisati najveći mogući broj nastao nadovezivanjem dva data prirodna broja **x** i **y**. Na primer, za 717 i 9 dobija se 9717.

- Pravougaonik, čije su strane paralelne koordinatnim osama, zadat je koordinatama donjeg levog i gornjeg desnog ugla. Za dva data pravougaonika Pr1 i Pr2, bez obzira na njihov raspored izračunati površinu zajedničkog pravougaonika (ako ga ima).

```
#Temena prvog pravougaonika
dx = int(input("Unesi x koordinatu donjeg temena: "))
dy = int(input("Unesi y koordinatu donjeg temena: "))

gx = int(input("Unesi x koordinatu gornjeg temena: "))
gy = int(input("Unesi y koordinatu gornjeg temena: "))

#Temena drugog pravougaonika
x1 = int(input("Unesi x koordinatu donjeg temena: "))
y1 = int(input("Unesi y koordinatu donjeg temena: "))

x2 = int(input("Unesi x koordinatu gornjeg temena: "))
y2 = int(input("Unesi y koordinatu gornjeg temena: "))
```

```
if(x1 > dx):
    dx = x1
if (y1 > dy):
    dy = y1

if (x2 < gx):
    gx = x2
if (y2 < gy):
    gy = y2

if gx - dx <= 0 or gy - dy <= 0:
    print("Nema zajednickog pravougaonika")
else:
    print((gy - dy)*(gx - dx))
```

- Napisati program kojim se za prirodne brojeve **n** i **x** izračunava suma prvih n sabiraka:

$$S = \sum_{i=1}^{n} \frac{i^i + x^i + (x+i)!}{(1+2+3+\cdots+i)+(1+2+\cdots+n)}$$

```
import math as m

def izracunaj_sumu(n):
    s = 0
    for i in range(1, n + 1):
        s += i
    return s

n = int(input("Unesi n: "))
x = int(input("Unesi x: "))

s = 0
suma_n = izracunaj_sumu(n)
suma_i = 0
fakt = m.factorial(x)
stepen = 1
```

```
for i in range(1, n + 1):
    fakt *= x + i
    suma_i += i
    stepen *= x
    s += (i**i + stepen + fakt) / (suma_i + suma_n)

print(s)
```

- Sa istog startnog mesta kreću dva automobila. Prvi se kreće brzinom v1 m/s, drugi v2 m/s pri čemu je v2 > v1. Drugi automobil polazi t minuta posle prvog. Napisati program kojim se određuje posle koliko minuta od polaska drugi automobil sustiže prvi.

```
v1 = float(input())
v2 = float(input())
t = float(input())
x = (v1 * t) / (v2 - v1) #broj minuta do sustizanja
print(format(x, '.2f'))
```

- Formirati i ispisati najveći mogući broj nastao nadovezivanjem dva data prirodna broja **x** i **y**. Na primer, za 717 i 9 dobija se 9717.

Primer 4

```
x = int(input("Unesi x: "))
y = int(input("Unesi y: "))

s = 1
while (s <= x):
    s *= 10
a = y * s + x;

s = 1
while (s <= y):
    s *= 10
b = x * s + y;

if (a > b):
    print(a)
else:
    print(b)
```

- Za dato n ispisati elemente niza 1, 2, 4, 5, 7, 9, 10, 12, 14, 16, 17, … koji se formira tako što se polazeći od broja 1 – prikazuje jedan neparan prirodni broj, zatim sledeća dva parna – 2, 4; pa sledeća 3 neparna – 5, 7, 9; sledeća 4 parna – 10, 12, 14, 16, … itd. Poslednja serija sadrži n elemenata.

Primer:

Ulaz	Izlaz
5	1 2 4 5 7 9 10 12 14 16 17 19 21 23 25