Tema 7.
Funcții în Python

Ce ne așteaptă?

1. Definirea și apelarea funcțiilor
2. Argumentele funcțiilor
3. Vizibilitatea variabilelor funcțiilor

1. Ce reprezintă o funcție?
2. Cum se definesc funcțiile în Python?
3. Cum se apelează funcțiile în Python?
4. Ce reprezintă parametrii și valoarea returnată a unei funcții?
5. Care se apelează o funcție în cadru altei funcții?
6. Cum se atribuie o funcție către o variabilă?
7. Ce reprezintă funcția imbricată și funcția de returnează o funcție?
8. Ce reprezintă argumentele unei funcții?
9. În ce constă esența argumentelor poziționale și cuvinte-cheie?
10. În ce constă esența argumentelor implicite și de lungime variabilă?
11. Cum se clasifică variabilele în funcție de vizibilitate?
12. Care sunt caracteristicile variabilelor locale și globale?
13. Care este rolul cuvântului-cheie global și a funcție globals()?

1. Definirea și apelarea funcțiilor

Definiția funcției

- **Funcția – un bloc de declarații ce realizează o anumită sarcină**
- **Necesitatea utilizării funcțiilor:**
  - Mentenanța mai ușoară a codului
  - Re-utilizarea codului
- **Tipuri de funcții:**
  - Funcții predefinite sau încorporate
  - Funcții definite de utilizator
- **Etapele funcției:**
  - Definirea funcției
  - Apelarea funcției

Definirea și apelarea funcției

- **Sintaxa definirii funcției:**

  def *nume_functie*(*parametrii*):
  '''string doc – descrierea functiei'''
  blocul functiei unde se realizeaza operatii
  return *valoare*

- **Exemplu de definire a functiei:**

```
def sumare_num(a,b):
    '''Functia de sumare a 2 numere'''
    suma = a + b
    return suma
```

- **Sintaxa apelării funcției:**

  nume_functie(*argumente*)

- **Exemplu de apelare a functiei:**

```
sumare_num(15,24)
```

Parametrii și valoarea returnată a funcției

- **Sintaza funcție fără parametri și valoarea returnată:**

```
def nume_functie():
    '''string doc – descrierea functiei'''
    blocul functiei unde se realizeaza operatii
```

- **Exemplu de definire și apelarea a functiei fără parametrii:**

```
def afisare():
    '''Functia de afisare a unui mesaj'''
    print("acesta este mesajul functiei")

afisare()
```

Cuvântul cheie return

- **Specificații:**

  - Permite returnarea rezultatului

  - Nu este obligatoriu dacă functie nu returnează nimic

  - Funcția fără return, implicit, returnează valoarea None

  - Se situează la finalul blocului funcție și permite ieșirea din cadrul acestuia

- **Exemplu de returnare a mai multor valori:**

```
def oper_matem(a,b):
    '''Functie ce efectueaza operatiile de sumare si scadere a 2 numere
       și returneaza aceste 2 valori'''
    suma=a+b
    difer=a-b
    return suma, difer

num1, num2 = oper_matem(10, 7)
print(num1)
print(num2)
```

Apelarea unei funcții în cadrul altei funcții

- **Sintaxa :**

```
def functie_1():
    blocul functiei_1

def functie_2():
    blocul functiei_2
    functie_1()

functie_2()
```

- **Exemplu de utilizare:**

```
def f1():
    print("Informatie despre prima functie")

def f2():
    print("Informatie despre a doua functie")
    f1()

f2()
```

Atribuirea funcției unei variabile și funcția ca parametru

- Exemplu de atribuire a unei funcții către o variabilă

```
def sumare(a,b):
    suma=a+b
    return suma
#atribuirea către o variabilă
adunare=sumare
#apelarea functie
print(adunare(2,3))
```

- Exemplu de utilizarea a unei functii ca parametru al altei funcție

```
def sumare(a,b):
    suma=a+b
    return suma
def scadere(c,d):
    dif=c-d
    return dif
num = scadere(sumare(2,8),7)
print(num)
```

Funcții imbricate și funcție ce returnează o altă funcție

- **Exemplu de funcții imbricate:**

```
def f1():
    print("Informații despre prima funcție")
    def f2():
        print("Informații despre a doua funcție")
    f2()
f1()
```

- **Exemplu de funcție ce returnează o altă funcție:**

```
def f1():
    print("Informații despre prima funcție")
    def f2():
        print("Informații despre a doua funcție")
    return f2
x=f1()
x()
```

2. Argumentele funcțiilor

Noțiune de argument al funcției

- **Parametrii funcției – variabilele utilizate ca date de intrare la definirea funcției**
- **Argumentele funcției – variabilele utilizate ca date de intrare la apelarea funcției**

```
def suma(a, b):
    c = a + b # a si b sunt parametrii functiei
    print(c)
# apelarea functiei
x = 10
y = 15
suma(x, y) # x si y sunt argumentele functiei
```

- **Tipuri de argumente**
  - Argumente poziționale
  - Argumente cuvinte-cheie (keyword)
  - Argumente implicite
  - Argumente de lungime variabilă
  - Argumente cuvinte-cheie de lungime variabilă

Argumente poziționale

- Argumentele poziționale se atribuie parametrilor funcției în ordinea scrierii lor
- Exemplu de utilizarea a argumentelor poziționale:

```
def operatii(a, b, c, d):
    suma = a + b
    difer = c - d
    print(f'Rezultatul sumarii este {suma}')
    print(f'Rezultatul scaderii este {difer}')

# apelarea functiei
x, y, z, t = 10, 15, 20, 5
operatii(x, y, z, t)
```

Argumente cuvinte-cheie

- Argumentele cuvinte-cheie necesită atribuirea explicită către parametri la apelare
- Exemplu de utilizarea a argumentelor cuvinte-cheie:

```
def operatii(a, b, c, d):
    suma = a + b
    difer = c - d
    print(f'Rezultatul sumarii este {suma}')
    print(f'Rezultatul scaderii este {difer}')

# apelarea functiei
x, y, z, t = 10, 15, 20, 5
operatii(b=x, d=y, a=z, c=t)
```

Argumente poziționale + argumente cuvinte-cheie

- Argumentele poziționale au prioritate în fața argumentelor cuvinte-cheie
- Exemplu de utilizarea a argumentelor poziționale și cuvinte-cheie:

```
def operatii(a, b, c, d):
    suma = a + b
    difer = c - d
    print(f'Rezultatul sumarii este {suma}')
    print(f'Rezultatul scaderii este {difer}')

# apelarea functiei
x, y, z, t = 10, 15, 20, 5
operatii(x, z, d=t, c=y)
```

Argumente implicite

- Argumentele implicite sunt omise la apelarea funcției deoarece valoarea a fost atribuită parametrului la definirea funcției

- Exemplu de utilizarea a argumentelor implicite:

```
def operatii(b, d, a=10, c=20):
    suma = a + b
    difer = c - d
    print(f'Rezultatul sumarii este {suma}')
    print(f'Rezultatul scaderii este {difer}')

# apelarea functiei
x, y = 15, 5
operatii(x, y)
```

Argumente de lungime variabilă

- Argumentul de lungime variabilă reprezintă un număr variabil de argumente
- Argumentul de lungime variabilă se notează cu steluță * în față
- Exemplu de utilizarea a argumentului de lungime variabilă:

```
def sumare(a, *b):
    suma = a
    for i in b:
        suma += i
    print(f'Rezultatul sumarii este {suma}')

# apelarea functiei
sumare(20)
sumare(20, 30)
sumare(20, 30, 40)
```

Argumente cuvinte-cheie de lungime variabilă

- Argumentul cuvânt-cheie de lungime variabilă reprezintă un număr variabil de argumente cuvinte-cheie
- Argumentul cuvânt-cheie de lungime variabilă se notează cu dublă steluță ** în față

- Exemplu de utilizarea a argumentului cuvânt-cheie de lungime variabilă:

```
def f1(**x):
    for k, v in x.items():
        print(f'{k} = {v}')

# apelarea functiei
f1(a=10)
f1(c=100, nume="Pavel")
f1(varsta=32, name="Pavel",
prenume="Nicolaev")
```

3. Vizibilitatea variabilelor

Clasificarea variabilelor

- **În funcție de vizibilitate pot fi:**
  - variabile locale
  - variabile globale

Variabile locale

- Este declarată în interiorul unei funcții
- Vizibilitatea este limitată în interiorul funcției
- Accesarea din exteriorul funcției = NameError
- Exemple de variabile locale:

```
def m():
    a=11
    print(a)
m()

def m():
    a=11
    print(a)
def n():
    print(a)
m()
n()
```

Variabile globale

- Este declarată în afara oricărei funcții
- Este vizibilă în întreg program
- Exemple de variabile globale:

```
a=9
b=121
def m():
    print("a in interiorul functiei m(): ",a)
    print("b in interiorul functiei m(): ",b)
def n():
    print("a in interiorul functiei n(): ",a)
    print("b in interiorul functiei n(): ",b)
m()
n()
```

Cuvântul-cheie global

- Prezența unei variabile locale și globale => variabila locala cu prioritatea

```
a=1
def f1():
    a=2
    print(f"In cadrul functiei f1() variabila a va avea valoarea: {a}")
f1()
print(f"In afara functiei f1() variabila a va avea valoarea: {a}")
```

- Exemplu de utilizare a cuvântului-cheie *global*

```
a=1
def f1():
    global a
    a=2
    print(f"In cadrul functiei f1() variabila a va avea valoarea: {a}")
f1()
print(f"In afara functiei f1() variabila a va avea valoarea: {a}")
```

Funcția globals()

- Returnează un dicționar cu toate variabile globale curente
- Permite utilizarea variabilei globale în cadrul funcției
- Exemplu de utilizare:

```
a=1
def f1():
    a=2
    print(f"In cadrul functiei f1() variabila a va avea valoarea: {a}")
    print(f"In cadrul functiei f1() variabila a poate fi si globala:
{globals()['a']}")

print(globals())
f1()
print(f"In afara functiei f1() variabila a va avea valoarea: {a}")
```