Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrarea de laborator nr. 4

# La disciplina „Programarea Declarativa”

Tema: " Funcții de ordin înalt în Python”

A efectuat: st. gr. SI-211 S. Chirița

A verificat: V. Rusu

Chișinău – 2023

**Mersul lucrarii:**

**Exerciții:**

1. Definiți un dicționar ce folosește șiruri de caractere pe post de chei și elemente float pe post

de valori. Afișați doar cheile dicționarului și mai apoi tupluri formate din chei și valori.

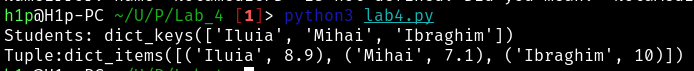
NotaMediePD ={"Iluia" : 8.9, "Mihai": 7.1, "Ibraghim":10 }

varStudent=NotaMediePD.keys()

print(f'Students: {varStudent}')

varReusita=NotaMediePD.items()

print(f'Tuple:{varReusita}')



2. Generați un număr aleator între 0 și 10000, ce reprezintă un număr de secunde. Calculați

reprezentarea numărului de secunde în ore, minute și secunde și afișați rezultatul formatat sub

forma hh:mm:ss. Alternativ, folosiți modulul datetime.

from datetime import timedelta

import random

secunde = random.randint(0, 10000)

deltaTimp = timedelta(seconds=secunde)

print(f'Timpul curent este:{deltaTimp}')

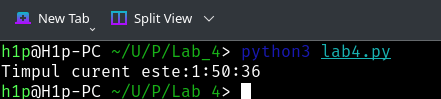
#Extragem orele,minutele,secundele

ore = deltaTimp.seconds // 3600

minute = (deltaTimp.seconds % 3600) //60

secunde = deltaTimp.seconds % 60

#print(f"Reprezentarea în format hh:mm:ss a {deltaTimp} secunde este: {ore:02d}:{minute:02d}:{secunde:02d}")



3. Definiți două obiecte de tip float și calculați suma, diferența lor utilizand funcții lamda().

Apelați toate funcțiile din listă pe rând și cu alte tipuri de date, folosiți funcția map().

obc1 = 7.77

obc2 = 6.66

suma = lambda x, y: x + y

diferenta = lambda x, y: x - y

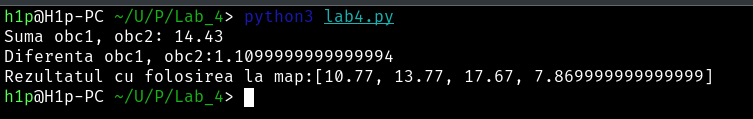
print(f'Suma obc1, obc2: {suma(obc1, obc2)}')

print(f'Diferenta obc1, obc2:{diferenta(obc1, obc2)}')

date =[3,6,9.9, 0.1]

rez=list(map(lambda x: suma(obc1, x),date))

print(f'Rezultatul cu folosirea la map:{rez}')



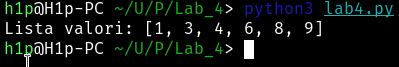
4. Definiți o listă de valori întregi și afișați doar valorile distincte din aceasta (folosiți funcția

filter())

listaValori = [1, 2, 3, 2, 4, 5, 6, 5, 7, 8, 9, 7]

valoriDistincte = list(filter(lambda x: listaValori.count(x) == 1, listaValori))

print("Lista valori:", valoriDistincte)



5. Definiți o funcție ce returnează numărul de apariții ale unui caracter într-un string (folosiți

funcția reduce()).

from functools import reduce

def numar\_aparitii\_caracter(sir, caracter\_cautat):

return reduce(lambda acc, caracter: acc + (1 if caracter == caracter\_cautat else 0), sir, 0)

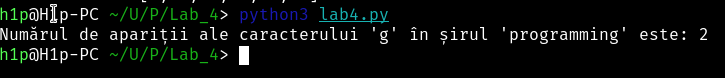
# Exemplu de utilizare

sir\_exemplu = "programming"

caracter\_cautat\_exemplu = "g"

rezultat = numar\_aparitii\_caracter(sir\_exemplu, caracter\_cautat\_exemplu)

print(f"Numărul de apariții ale caracterului '{caracter\_cautat\_exemplu}' în șirul '{sir\_exemplu}' este: {rezultat}")



6. Definiți o funcție ce concatenează oricâte stringuri sunt date la intrarea sa.

def conc(\*args):

return ''.join(args)

print(f'String concatinat:{conc("test ","this ","function.")}')



7. Definiți o listă de funcții lambda ce returnează: al doilea caracter dintr-un string; stringul cu

litere majuscule; poziția pe care se găsește un anumit caracter dat la intrare. Apelați toate

funcțiile din listă pe rând.

functii\_lambda = [

lambda s: s[1] if len(s) > 1 else None,

lambda s: s.upper() if isinstance(s, str) else None,

lambda s, c: s.index(c) if c in s else -1

]

sir = "example"

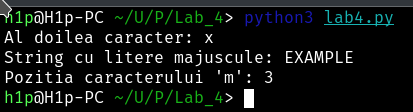
caracter\_cautat = "m"

rezultate = [functie(sir, caracter\_cautat) if i == 2 else functie(sir) for i, functie in enumerate(functii\_lambda)]

print("Al doilea caracter:", rezultate[0])

print("String cu litere majuscule:", rezultate[1])

print(f"Pozitia caracterului '{caracter\_cautat}':", rezultate[2])



8. Definiți o funcție ce calculează media a trei note sprecificate la intrare. Dacă la apel nu se

trimit toate notele, se vor folosi valori implicite egale cu 4. Apelați funcția cu diferite

combinații de argumente poziționale și keyword.

def calcul\_medie(nota1=4, nota2=4, nota3=4):

return (nota1 + nota2 + nota3) / 3

medie1 = calcul\_medie(7, 8, 9)

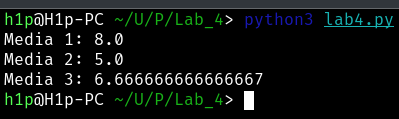
medie2 = calcul\_medie(5, 6)

medie3 = calcul\_medie(nota1=9, nota3=7)

print(f"Media 1: {medie1}")

print(f"Media 2: {medie2}")

print(f"Media 3: {medie3}")



9. Definiți o funcție recursivă ce afișează suma primelor N numere naturale.

def sumaPrime(n):

if n == 0:

return 0

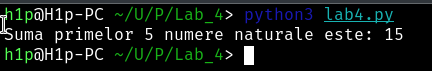
else:

return n + suma\_primele\_n(n - 1)

n = 5

rezultat = suma\_primele\_n(n)

print(f"Suma primelor {n} numere naturale este: {rezultat}")



**Concluzie:**

Funcțiile de ordin înalt în Python reprezintă o caracteristică esențială care permite manipularea și transmiterea funcțiilor ca argumente către alte funcții sau întoarcerea lor ca rezultate ale altor funcții. Această capacitate facilitează dezvoltarea de cod mai concis, modular și flexibil. Utilizarea funcțiilor lambda și funcțiilor predefinite precum map(), filter(), reduce() aduce un nivel de expresivitate și abstractizare care sporește citibilitatea codului.

Exemplul cu funcții lambda și map() într-un context de funcții de ordin înalt arată cum putem manipula și aplica operații pe date folosind abstracții puternice, fără a fi nevoie să scriem bucle sau instrucțiuni detaliat enumerate. Această paradigmă se bazează pe principiul "codul mai citibil este adesea mai bun și mai ușor de întreținut".

Top of Form

Bottom of Form