**Informții generale despre temă:**

Nume: Tofan Denis-Cristian

Variablie: X=5(numărul literelor din numele studentului)

Y=13(numarul literelor din prenumele studentului)

**Exercițiul 1:**

1.Prerequisites si Configurare

Pentru a putea implementa acest cod este recomandată o versiune de Python 3.8+, în cazul de fata codul a fost testat pe un terminal cu o versiune Python 3.13.0 pe 64bit si ca editor de cod/terminal Visual Studio Code v.1.96 pe 64 bit. De asemenea se pot folosii si alte programe precum PyCharm sau chiar in terminalul sistemului de operare.

Pasi esențiali in rularea programului:

-Se creeaza un fișier Python, numit “primul\_exercitiu.py“;

- Se insereaza codul sursa furnizat in fisierul anterior;

-Se ruleaza programul in terminal prin intermediul comenzii: python “primul\_exercitiu“.py;

Eventuale erori intalnite in rularea programului:

-Locatia folosita in terminal nu coincide cu path-ul fisierului executabil, rezolvarea este folosirea comenzii: cd C:\Users\...

-Eroare SyntaxError, in acest caz verifica daca folosesti o versiune Python 3.8+

Programul în cauză va afișa departamentele managerilor (având în vedere că X = 5, iar X % 3 == 2), datele acestora și numărul total de manageri și angajați, respectiv 4 pentru fiecare.

2.Explicatie detaliata a codului sursa

**1.Linia 1:**

class Employee:

    """Common base class for all employees"""

    empCount = 0

Se defineste clasa Employee utilizata pentru angajati, iar empCount este un atribut de clasa utilizat pentru a tine evidenta numarului total de angajati creati

**2.Linia 5**

   def \_\_init\_\_(self, name, salary):

        self.name = name

        self.salary = salary

        self.tasks = {}

        Employee.empCount += 1

Intalnim un constructor ce initializeaza un angajat cu nume si salariu si creeaza un dictionar(tasks) pentru a gestiona sarcinile si starea acestora. Contorul empCount se incrementeaza pentru un nou angajat

**3.Linia 11**

def display\_emp\_count(self):

        print(f"Total number of employee(s) is {Employee.empCount}")

Metoda pentru a afisa numarul total de angajati

**4.Linia 14**

def display\_employee(self):

        print("Name: ", self.name, ", Salary: ", self.salary)

Metoda pentru a afisa numele si salariul unui angajat

    def \_\_del\_\_(self):

        Employee.empCount -= 1

Destructor ce decrementeaza contorul anagajatilor cand un obiect e distrus

**5.Linia 19**

def update\_salary(self, new\_salary):

        self.salary = new\_salary

Actualizeaza salariul unui angajat

**6.Linia 22**

def modify\_task(self, task\_name, status="New"):

        self.tasks[task\_name] = status

Adauga sau modifica starea unei sarcini pentru angajat, oferind un status de “New”

**7.Linia 25**

def display\_task(self, status):

        print(f"Taskuri cu statusul {status}")

        for name in self.tasks.keys():

            if self.tasks[name] == status:

                print(name)

Afiseaza sarcinile pe care un angajat le are de efectuat in functie de statusul acestora

**8.Linia 31**

class Manager(Employee):

    mgr\_count = 0

Definim clasa Manager ce mosteneste clasa Employee, iar atributul mgr\_count reprezinta un contor pentru numarul de manageri

**9.Linia 34**

def \_\_init\_\_(self, name, salary, departament):

        NUME\_ECHIPA = "F31-"

        super().\_\_init\_\_(name, salary)

        self.departament = NUME\_ECHIPA + departament

        Manager.mgr\_count += 1

Acest constructor apelează constructorul clasei părinte Employee pentru a inițializa numele și salariul, iar în plus adaugă atributul departament, care stochează informații despre departamentul managerului. Totodată, incrementează contorul global al managerilor (mgr\_count).

**10.Linia 40**

 def display\_employee(self):

        print(f"Departament: {self.departament}")

Se suprascrie metoda din clasa principala pentru a afisa in acest caz doar departamentul unui manager.

**11.Linia 43**

def \_\_del\_\_(self):

        Manager.mgr\_count -= 1

Decrementam contorul anagajatilor in momentul in care distrugem un obiect de tip manager

**12.Linia 48**

manager1 = Manager("Andrei", 3800, "Tester")

    manager2 = Manager("Mihai", 3200, "Software")

    manager3 = Manager("Oana", 2200, "Marketing")

    manager4 = Manager("Alin", 3500, "Networking")

Conform cerintei atribuim 4 (Y=13, Y/3) obiecte de tip manager

**13.Linia 53**

 manager1.display\_employee()

    manager2.display\_employee()

    manager3.display\_employee()

    manager4.display\_employee()

Afisam deparatmentele fiecarui manager, conform metodei display\_employee suprascrisa

**14.Linia 58**

 print(f"Numărul de manageri este: {Manager.mgr\_count}")

    print(f"Numărul total de angajați este: {Employee.empCount}")

Afisam numarul total de manageri si de angajati, avand in contorul atributului empCount

Referinte bibliografice:

-Documentatia incarcata pe (<https://tiny.cc/pclp3pp>)

-W3Schools-Python:( <https://www.w3schools.com/python/> )

( <https://www.w3schools.com/python/python_classes.asp>)

-GeeksForGeeks OOPs Concepts(<https://www.geeksforgeeks.org/python-oops-concepts/> )

-Object Oriented Programming (OOP) in Python(<https://www.youtube.com/watch?v=MikphENIrOo&ab_channel=TraversyMedia>)

-Programiz (<https://www.programiz.com/python-programming/object-oriented-programming>)

**Exercitiul 2:**

1.Prerequisites & Configurare

Se instaleaza modulele :

1)Pandas folosit pentru a manipula si utiliza datele, instalarea se face prin scrierea comenzii “pip install pandas” in terminal

2)Matplotlid folosit pentru a genera grafice, intalarea acestuia se face prin scrierea comenzii “pip install matplotlid” in terminal

Se verfica fisierul data.csv daca se afla in folderul proiectului, in caz contar daca fisierul nu este in acelasi director inlocuim “doc=pd.read\_csv("data.csv")” cu “doc=pd.read\_csv(r”C:\Users\....”)

3)Se ruleaza programul asteptand afisarea a trei grafice, primul reprezentatand un grafic cu toate valorile din fisierul CSV, al doilea cu primele 5 valori(X=5) si al treilea cu ultimele 13 valori(Y=13)

2.Explicatie detaliata a codului sursa

**1.Linia 1:**

import matplotlib.pyplot as plt

import pandas as pd

Utilizarea bibliotecilor necesare rularii codului

**2.Linia 4**

df=pd.read\_csv("data.csv")

Citirea fisierului de tip CSV utilizand comanda pd.read\_csv(pandas), astfel rezultand un df(Data Frame) reprezentand o structura de date bidimensionala, precum un tabel

**3.Linia 6**

df.plot()

plt.title("Toate valorile")

plt.xlabel("Index")

plt.ylabel("Valori")

plt.legend(title="Parametrii")

plt.savefig("Toate valorile.png")

plt.show()

Prin comanda df.plot() se creeaza un grafic, prin restul de comenzi plt. se afiseaza titlu, etichetele axelor x respectiv pe y, ale graficului si legenda cu titlul acesteia. Prin comanda plt.savefig se salveaza imaginea graficului formata in folderul proiectului

**4.Linia 14**

df[:5].plot(color=['orange','cyan','magenta','lime'])

plt.title("Primele 5 valori")

plt.xlabel("Index")

plt.ylabel("Valori")

plt.legend(title="Parametrii")

plt.savefig("Primele 5 valori.png")

plt.show()

Se schimba afisarea utilizand slicing pentru primele 5 valori(X=5), se adauga culorile dorite pentru o observare mai usoara a datelor din tabel.

**5.Linia 22**

x=df[['Durata', 'Puls']][-13:]

x.plot(color=['brown', 'pink'])

plt.title("Ultimele 13 valori")

plt.xlabel("Index")

plt.ylabel("Valori")

plt.legend(title="Parametrii")

plt.savefig("Ultimele 13 valori.png")

plt.show()

Se afiseaza ultimele 13 valori din grafic(Y=13), utilizand slicing negative

Referinte bibliografice:

-Documentatia incarcata pe (<https://tiny.cc/pclp3pp>)

-W3Schools (<https://www.w3schools.com/python/matplotlib_pyplot.asp> )

-Code of the Future (<https://www.youtube.com/watch?v=X69y9N65Iu8&ab_channel=CodeoftheFuture>)

-CodersLegacy (<https://www.youtube.com/watch?v=I6tvyasOxaQ&ab_channel=CodersLegacy> )

Bonificatii:

* Daca la problema 1 imaginati cateva functii de test si le intregrati in solutie (ex: cu pytest)

Acest cod are rolul de a oferii un set de teste automate pentru verificarea claselor Employee si Manager din primul exercitiu aflate la fisierul primul\_exercitiu.py. Clasele fiind importate prin intermediul comenzii “from primul\_exercitiu import Employee, Manager”

Pentru ca programul sa functioneze avem nevoie de instalarea pytest prin intermediul comenzii “pip install pytest” acesta find un framework pentru a rula testele ulterioare

Verificarea instalarii se face folosind comanda “pytest –version”

Funcțiile din cod testează clasele Employee și Manager folosind assert pentru a verifica corectitudinea rezultatelor. test\_create\_employee validează crearea unui angajat, verificând că numele, salariul și contorul global al angajaților (empCount) sunt corecte. test\_update\_salary testează metoda update\_salary, asigurând că salariul este actualizat corect. test\_create\_manager verifică crearea unui manager, confirmând că departamentul este prefixat cu „F31-” și că se incrementează contorul managerilor (mgr\_count). Prin utilizarea assert, testele se asigură că metodele și atributele funcționează conform așteptărilor.

Functia “if \_name\_==”\_main\_” verifica rularea testelor si oferirea unui feedback in cazul in care totul functioneza cum trebuie