---In cadrul rezolvarii temei am folosit Python 3.13.1 si Visual Studio Code

Gogan = 5 litere => X =10

Ady-Cristian = 11 litere => Y = 11

1. 1. **class Employee:**  
   Declarația unei clase Python numită Employee. Aceasta servește drept clasă de bază pentru toți angajații.
2. **"""Common base class for all employees"""**  
   Un docstring ce descrie clasa Employee ca fiind clasa de bază comună pentru toți angajații.
3. **empCount = 0**  
   Variabilă de clasă care păstrează numărul total de angajați. Este partajată de toate instanțele clasei.
4. **def \_\_init\_\_(self, name, salary):**  
   Constructorul clasei, folosit pentru a inițializa atributele individuale ale unui angajat.
5. **self.name = name**  
   Atribuie numele angajatului atributului instanței.
6. **self.salary = salary**  
   Atribuie salariul angajatului atributului instanței.
7. **self.tasks = {}**  
   Inițializează un dicționar gol pentru a stoca sarcinile angajatului.
8. **Employee.empCount += 1**  
   Incrementează contorul general al angajaților ori de câte ori se creează un angajat.
9. **def display\_emp\_count(self):**  
   Metodă pentru a afișa numărul total de angajați.
10. **print(f"Total number of employee(s) is {Employee.empCount}")**  
    Afișează numărul total de angajați.
11. **def display\_employee(self):**  
    Metodă pentru a afișa numele și salariul angajatului curent.
12. **print("Name:", self.name, ", Salary: ", self.salary)**  
    Afișează detalii despre angajat.
13. **def \_\_del\_\_(self):**  
    Destructorul clasei, decrementând contorul de angajați la ștergerea unui obiect.
14. **def update\_salary(self, new\_salary):**  
    Metodă pentru a actualiza salariul angajatului.
15. **self.salary = new\_salary**  
    Actualizează salariul cu valoarea specificată.
16. **Comentarii pentru metodele add\_task și update\_tasks:**  
    Aceste metode sunt comentate și nu sunt active în cod. Acestea ar putea fi folosite pentru gestionarea sarcinilor.
17. **def modify\_task(self, task\_name, status="New"):**  
    Metodă pentru a adăuga sau modifica o sarcină.
18. **self.tasks[task\_name] = status**  
    Adaugă sau actualizează sarcina cu starea specificată.
19. **def display\_task(self, status):**  
    Metodă pentru a afișa sarcinile cu un anumit status.
20. **print(f"Taskuri cu statusul {status}")**  
    Afișează o introducere pentru sarcinile de un anumit tip.
21. **for name in self.tasks.keys():**  
    Iterează prin numele sarcinilor.
22. **if self.tasks[name] == status:**  
    Verifică dacă statusul unei sarcini corespunde celui specificat.
23. **print(name)**  
    Afișează numele sarcinii care corespunde criteriului.
24. **e1=Employee("ionel", 10)** etc.  
    Creează trei obiecte de tip Employee.
25. **e1.display\_employee() etc.**  
    Afișează informațiile despre fiecare angajat.

**Clasa Manager**

1. **class Manager(Employee):**  
   Declarația unei clase numite Manager, care moștenește Employee.
2. **mgr\_count = 0**  
   Variabilă de clasă specifică pentru a păstra numărul managerilor.
3. **def \_\_init\_\_(self, name, salary, departament):**  
   Constructor ce extinde constructorul din clasa Employee adăugând atributul departament.
4. **self.departament = departament**  
   Inițializează departamentul managerului.
5. **Manager.mgr\_count += 1**  
   Incrementează contorul de manageri.
6. **def display\_employee(self):**  
   Metodă suprascrisă pentru a afișa detaliile managerului, inclusiv departamentul.
7. **print("Departamanet: F04 ",self.departament)**  
   Afișează departamentul managerului.
8. **m1=Manager("ian", 5, "HR")** etc.  
   Creează trei obiecte de tip Manager.
9. **m1.display\_employee() etc.**  
   Afișează informațiile despre fiecare manager.

**Final**

1. **print("Nr of employees: ", Employee.empCount)**  
   Afișează numărul total de angajați (inclusiv managerii).
2. **print("Nr of manager: ", Manager.mgr\_count)**  
   Afișează numărul total de manageri.

Acest cod demonstrează folosirea moștenirii, metodelor, atributele de clasă și instanță, precum și manipularea datelor.

1. 2. **import matplotlib.pyplot as G**  
   Importă biblioteca matplotlib.pyplot pentru a crea grafice, atribuind aliasul G. Astfel, în loc de a scrie matplotlib.pyplot, vom folosi G.
2. **import pandas as C**  
   Importă biblioteca pandas pentru manipularea și analiza datelor, atribuind aliasul C. În loc de pandas, se va folosi C.
3. **df = C.read\_csv("data.csv")**  
   Citește un fișier CSV numit data.csv utilizând funcția read\_csv din pandas. Rezultatul este stocat într-un DataFrame numit df, o structură de date tabelară.
4. **df.plot()**  
   Creează un grafic pentru toate coloanele numerice din DataFrame. Tipul implicit de grafic este o linie.
5. **G.title("Toate valorile")**  
   Setează titlul graficului la „Toate valorile”.
6. **G.show()**  
   Afișează graficul creat anterior pe ecran.
7. **df[:5].plot()**  
   Selectează primele 5 rânduri din DataFrame folosind slicing (df[:5]) și creează un grafic pentru datele din aceste rânduri.
8. **G.title("Primele cinci valori")**  
   Setează titlul graficului la „Primele cinci valori”.
9. **G.show()**  
   Afișează graficul pentru primele 5 rânduri pe ecran.
10. **df[-11:].plot()**  
    Selectează ultimele 11 rânduri din DataFrame folosind slicing (df[-11:]) și creează un grafic pentru aceste rânduri.
11. **G.title("Ultimele 11 valori")**  
    Setează titlul graficului la „Ultimele 11 valori”.
12. **G.show()**  
    Afișează graficul pentru ultimele 11 rânduri pe ecran.

**Rezumat**

Acest cod folosește bibliotecile matplotlib și pandas pentru a citi date dintr-un fișier CSV și a genera trei grafice: unul pentru toate datele, unul pentru primele 5 valori și unul pentru ultimele 11 valori. Fiecare grafic are un titlu corespunzător.

*Referinte:*

* [*https://www.w3schools.com/python/pandas/default.asp*](https://www.w3schools.com/python/pandas/default.asp)
* [*https://www.w3schools.com/python/matplotlib\_intro.asp*](https://www.w3schools.com/python/matplotlib_intro.asp)
* *calp.python.extras de la Files > Class.Materials*
* [*https://code.visualstudio.com/docs/python/tutorial-flask*](https://code.visualstudio.com/docs/python/tutorial-flask)
* [*https://github.com/microsoft/python-sample-vscode-flask-tutorial*](https://github.com/microsoft/python-sample-vscode-flask-tutorial)
* [*https://code.visualstudio.com/docs/python/testing*](https://code.visualstudio.com/docs/python/testing)
* <https://www.w3schools.com/python/pandas/pandas_csv.asp>
* <https://www.w3schools.com/python/python_classes.asp>
* <https://www.w3schools.com/python/matplotlib_pyplot.asp>
* <https://www.youtube.com/watch?v=ZDa-Z5JzLYM>
* <https://www.youtube.com/watch?v=AFCmLSgJiJ0&list=PLSwH4ViBDl2Tau2B--9cJB6P5394XN42O>