# Documentație Exercițiul 1

* **Prerequisites și configurare:**

Este nevoie sa aveți instalată o versiune a Python pe calculator. Dacă doriți puteți avea și un editor precum Pycharm sau VSCode, dar nu este obligatoriu.

Pentru a rula codul deschideți terminalul (scrieți în bara de search cmd și dați click) și utilizați comanda python ex1.py.

Sfat: S-ar putea să apară erori fiindcă locația folosită în terminal nu coincide cu path-ul fișierului. Vă recomand să descărcați fișierul ex1.py pe Desktop, iar apoi să rulați în cmd comanda cd Desktop, după care scrieți python ex1.py.

Se vor afișa departamentele la care lucrează angajații, precum și numărul total de manageri și angajați.

* **Explicarea codului ex1\_PYTHON.py**

Clasa Employee:

1. class Employee: Definiția clasei Employee:
   * + emp\_count este o variabilă statică care numără toate instanțele Employee create.
2. def \_\_init\_\_(self, name, salary):

* Constructorul clasei acceptă numele și salariul unui angajat.
* self.name și self.salary stochează aceste informații.
* self.tasks este un dicționar care va ține evidența sarcinilor atribuite angajatului.
* Employee.empCount este incrementat de fiecare dată când se crează o instanță nouă de Employee.

1. def display\_emp\_count(self):

* O metodă pentru a afișa numărul total de angajați. Folosește Employee.empCount pentru a obține această valoare.

1. def display\_employee(self):

* Afișează informațiile de bază ale angajatului: numele și salariul acestuia.

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

* Este apelat automat atunci când o instanță Employee este distrusă (de exemplu, când programul se termină).
* Employee.empCount este decrementat pentru a reflecta scăderea numărului de angajați.

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

* Permite actualizarea salariului unui angajat cu un nou salariu specificat.

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

* Permite adăugarea sau modificarea statusului unei sarcini. status are valoarea implicită "New".

1. def display\_task(self, status):

* Afișează sarcinile cu un anumit status. Pentru fiecare sarcină, verifică dacă statusul corespunde și le afișează.

Clasa Manager:

1. class Manager(Employee):

* Moștenește din Employee.
* mgr\_count va număra toate instanțele Manager.

1. def \_\_init\_\_(self, name, salary, department):

* Constructorul clasei Manager adaugă prefixul "F27 " la departament și îl trimite la constructorul clasei părinte Employee folosind super().
* self.department va conține un text combinat dintre "F27 " și departamentul specificat de utilizator.

1. def display\_employee(self):

* Aici se schimbă metoda display\_employee din clasa Manager:
* Dacă x % 3 == 0, ar trebui să afișeze doar numele angajatului.
* Dacă x % 3 == 1, ar trebui să afișeze doar salariul angajatului.
* Dacă x % 3 == 2, afișează doar departamentul, deci am setat metoda să afișeze doar self.department.

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

* Este apelat automat când un obiect Manager este distrus.
* Manager.mgr\_count este decrementat pentru a reflecta reducerea numărului de manageri.

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

* Afișarea valoarii atributului emp\_count pentru o instanță a clasei Manager.

1. print(f"Numarul de angajati este: {Employee.emp\_count}"):

* Afișarea valoarii atributului emp\_count pentru o instanță a clasei Employee.
* **Scopul acestui cod este:**
* Crearea și validarea corectă a obiectelor: Asigură-te că fiecare obiect Employee și Manager este creat corect și că proprietățile acestora sunt setate corect.
* Contorizarea corectă a angajaților și managerilor: Verifică dacă numărul total de angajați și manageri se actualizează corect după crearea obiectelor.
* Memoria eliberată corect: Asigură-te că obiectele sunt șterse după testare pentru a preveni scurgerile de memorie.

<https://github.com/amaliapopa/tema_python>

**Bibliografie:**

W3 Schools: <https://www.w3schools.com/python/python_classes.asp>

Download Python: <https://www.python.org/downloads/release/python-3124/>