**Documentatie Exercitiul 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 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.

**BONUS:**

Este nevoie de instalarea pytest prin rularea in cmd a comenzii **pip install pytest.** Pentru a testa funcționalitatea fișierului **ex1.py** este nevoie de crearea unui nou fișier Python de test **test\_ex1.py**. In terminal dvs trebuie să rulați comanda **pytest test\_exercitiul1.py**.

***Explicarea codului ex1.py:***

**Clasa Employee:**

1. **class Employee:** Definiția clasei Employee
   * empCount 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.
3. **def display\_emp\_count(self):**:
   * O metodă pentru a afișa numărul total de angajați. Folosește Employee.empCount pentru a obține această valoare.
4. **def display\_employee(self):**:
   * Afișează informațiile de bază ale angajatului: numele și salariul acestuia.
5. **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.
6. **def update\_salary(self, new\_salary):**:
   * Permite actualizarea salariului unui angajat cu un nou salariu specificat.
7. **def modify\_task(self, task\_name, status="New"):**:
   * Permite adăugarea sau modificarea statusului unei sarcini. status are valoarea implicită "New".
8. **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.
   * Manager.mgrCount va număra toate instanțele Manager.
2. **def \_\_init\_\_(self, name, salary, department):**
   * Constructorul clasei Manager adaugă prefixul "F22 " la departament și îl trimite la constructorul clasei părinte Employee folosind super().
   * self.department va conține un text combinat dintre "F22 " și departamentul specificat de utilizator.
3. **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.
4. **def \_\_del\_\_(self):**
   * Este apelat automat când un obiect Manager este distrus.
   * Manager.mgrCount este decrementat pentru a reflecta reducerea numărului de manageri.
5. **print(f"Numarul de manageri este: {Manager.mgrCount}") :**

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

1. **print(f"Numarul de angajati este: {Employee.empCount}"):**

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

***Explicarea codului test\_ex1.py:***

*import pytest*

*from ex1 import Employee, Manager*

Importam biblioteca **pytest**, deoarece avem nevoie de ea. Apoi din fisierul **ex1** importam clasele noastre (**Employee** si **Manager**)

***def test\_create\_employee():***

1. **Crearea unui angajat**: Se creează un obiect Employee cu numele "Ionel" și salariul 4000.
2. **Asserturile**:

* **emp.name == "Ionel"**: Confirmă că numele angajatului a fost setat corect.
* **emp.salary == 4000**: Verifică dacă salariul a fost alocat corect.
* **Employee.empCount == 1**: Asigură-te că contorizarea numărului de angajați este corectă după crearea angajatului.

1. **Ștergerea obiectului**: După verificarea asercțiilor, obiectul emp este șters pentru a elibera memoria.

***test\_create\_manager()****:*

1. **Crearea unui manager**: Se creează un obiect Manager cu numele "Denisa", salariul 5500 și departamentul "HR". Observați că prefixul "F22 " este adăugat automată pentru fiecare departament.
2. **Asercții**:

* **mgr.department == "F22 HR"**: Verifică dacă departamentul managerului este corect format.
* **Manager.mgrCount == 1**: Confirmă că contorizarea numărului de manageri este corectă după crearea managerului.

1. **Ștergerea obiectului**: După validarea asercțiilor, obiectul mgr este șters pentru a elibera memoria.

*if \_\_name\_\_ == "\_\_main\_\_":*

*pytest.main()*

* **Executarea cu pytest**: Se execută toate testele din fișier. pytest.main() va rula toate testele definite în fișierul curent și va produce un raport de testare.

**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.

***Bibliografie:***

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

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

Clasele explicate: <https://www.guru99.com/ro/python-class-objects-object-oriented-programming-oop-s.html>

Pytest: <https://realpython.com/pytest-python-testing/>

<https://testdriven.io/blog/pytest-for-beginners/>