**Mini-proiect 3 – Alarm Clock App**



*Proiectele sunt foarte importante in dezvoltarea ta si cimentarea cunostiintelor acumulate de-a lungul sesiunilor de curs.*

*Acestea sunt foarte apreciate si de asemenea necesare pentru angajare.*

In dezvoltarea mini-proiectului **“Alarm Clock App”** vei **exersa cunostiintele acumulate** la curs, precum:

* **Afisarea mesajelor in consola**
* **Formatarea string-urilor folosind f-strings**
* **Utilizarea operatorilor**
* **Utilizarea if-urilor, ciclurilor repetitive**
* **Folosirea instructiunilor de control pentru ciclurile repetitive**
* **Variabile, structuri de date**
* **Utilizarea notiunilor de OOP**
* **Interactiunea cu utilizatorul din terminal**
* **Interactiunea cu o librarie standard din Python time**
* **Interactiunea cu o librarie externa, numita pygame**
* **Dezvoltarea unui proiect folosind Github**

1. Primul pas este **crearea** unui **repository** pe Github (sugestie nume: **alarm\_clock\_with\_python**)

* Repository-ul va fi **private**.
* Repository-ul va avea **fisierul .gitignore** (template: Python)
* Pentru ca trainerul sa poata vedea proiectul, acesta trebuie adaugat ca si colaborator (din **Settings 🡪 Collaborators 🡪 cosminabacter01**)

**NOTA: Pentru a se putea vedea modul de lucru, este recomandat sa facem commit-uri dese, cu modificari mici pe care le aduceti proiectului (ex: modificare gitignore, adaugare structura proiect, creare clasa noua, creare fisier nou, modificare/refactorizare metoda, logica noua adaugata in cod etc)**

* **Cloneaza repo-ul pe local**.
* **Modifica** **fisierul .gitignore** astfel incat sa se ignore folder-ul .idea (generat automat cand lucram in Pycharm).

**FOARTE IMPORTANT! Trebuie sa ne asiguram mereu, ca in repo-ul de pe Github, nu avem fisiere/foldere private (a caror denumire incepe cu . “punct”) + de asemenea, folderele a caror denumire incepe cu doua underscore-uri (Ex: \_\_pycache\_\_).**

1. **Descriere aplicatie**

* Aceasta este o aplicatie prin care utilizatorul va seta o alarma din terminal.

1. **Requirements**

* **Instalare librarie pygame:**
  + Din terminal, ruleaza comanda: *pip install pygame*.
* **Descarcare sunet alarma**, format **mp3**: <https://orangefreesounds.com/alarm-clock-ringing-sound-effect/>
  + Dupa ce ai descarcat sunetul de alarma, pune fisierul in proiectul tau.

1. **Structura proiectului**

* Logica aplicatiei va fi implementata prin intermediul unei clase, numita **AlarmClock**. Creeaza in proiect un fisier numit **alarm\_clock.py**, in care sa adaugi logica pentru aceasta.
* **Entrypoint-ul** in aplicatie va fi un alt fisier, numit **main.py**

1. Fisierul **alarm\_clock.py**:

* **Importa modulul time**
* **Importa libraria pygame**
* Implementeaza clasa **AlarmClock**:
  + **Constructor**:
    - Nu ia parametri.
    - Se va seta un atribut pe obiect, numit **alarm\_time**, care va avea valoarea None.
    - Se va initializa sistemul de sunet din libraria pygame astfel: *pygame.mixer.init().* (Aceasta initializare trebuie facuta pentru a putea utiliza functionalitatile de sunet disponibile in libraria pygame).
  + Metoda **set\_alarm():**
    - Ia 2 parametri: **hour, minute** (reprezentand ora si minutul la care se va seta o alarma)
    - Se seteaza atributul de pe obiect, alarm\_time, ca fiind un tuplu, format din cei doi parametri.
    - Se afiseaza un mesaj: “Alarm set for {hour}:{minute}”
  + Metoda **check\_alarm():**
    - Este metoda care va verifica din secunda in secunda, daca e timpul ca alarma sa fie declansata.
    - Pentru implementarea logicii, se va folosi un while, sub forma “while True”.
      * In interiorul while-ului, se va lua timpul curent, si se va salva intr-o variabila.
      * Pentru a lua timpul curent, se poate folosi functia localtime, disponibila in modulul time (importat la inceput de fisier), in felul urmator:
        + current\_time = time.localtime(time.time())
        + Pe obiectul returnat, salvat in variabila current\_time, avem acces la ora, accesand atributul tm\_hour, si la minute, accesand atributul tm\_min (ex: current\_time.tm\_hour).
      * Se verifica daca ora curenta si minutele sunt egale ca cele setate pe atributul alarm\_time. Daca da, se va apela metoda trigger\_alarm(), disponibila pe self si se va intrerupe ciclul repetitiv folosind intructiunea potrivita din Python.
      * Se seteaza un time.sleep de 1 secunda (time.sleep(1)), pentru a astepta o secunda pana se trece la urmatoarea iteratie.
  + Metoda **trigger\_alarm():**
    - Afiseaza mesajul “Alarm! It’s time to wake up!”
    - Se va declansa sunetul de alarma salvat in fisierul mp3:
      * Prima data, acesta se va incarca: pygame.mixer.music.load(<calea\_catre\_fisier\_mp3>)
      * Apoi se va declansa sunetul:

pygame.mixer.play()

* + - Pentru a ne asigura ca executia programului va astepta pana cand alarma se opreste vom folosi urmatorul cod:

while pygame.mixer.music.get\_busy():

pygame.time.Clock().tick(10)

1. **Fisierul main.py**:

* Importa clasa AlarmClock
* Instantiaza un obiect din clasa AlarmClock
* Foloseste un while True in care:
  + Printeaza optiunile utilizatorului (1 Set Alarm, 2 – Quit)
  + Ia optiunea aleasa de utilizator (1 sau 2)
  + Daca optiunea aleasta este 1:
    - Cere utilizatorului ora (0-23) la care utilizatorul doreste sa seteze alarma, ca si int, si minutele (0-59), tot ca int.
    - Seteaza alarma
    - Verifica alarma (apeland metoda check\_alarm, astfel incat sa se astepte declansarea acesteia)
  + Daca optiunea aleasta este 2:
    - Afiseaza mesajul “Goodbye”
    - Intrerupe ciclul repetitiv
  + Daca alta optiunea a fost aleasa:
    - Afiseaza mesajul “Invalid choice. Please select a valid option.”