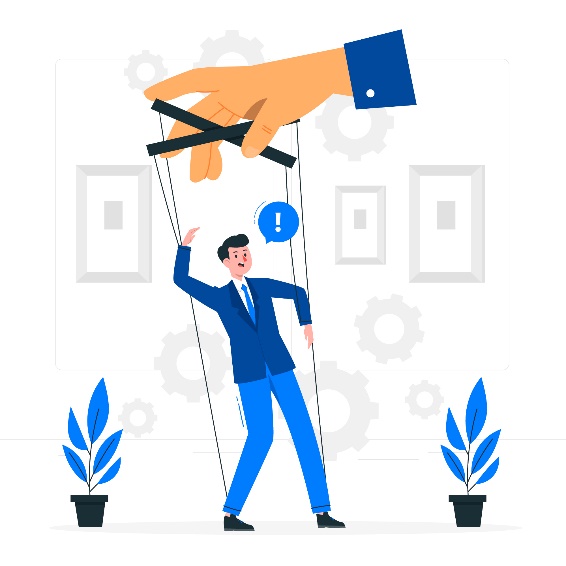
**Mini-proiect 2 – Joc Spanzuratoarea**



*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 **“Joc Spanzuratoarea”** vei **exersa cunostiintele acumulate** la curs, precum:

* **Afisarea mesajelor in consola**
* **Interactiunea cu utilizatorul din consola**
* **Formatarea string-urilor folosind f-strings**
* **Utilizarea operatorilor**
* **Utilizarea if-urilor, ciclurilor repetitive**
* **Variabile, structuri de date**
* **Tratarea exceptiilor**
* **Utilizarea notiunilor de OOP (clase, obiecte, atribute, metode)**
* **Interactiunea cu fisiere**
* **Folosirea importurilor**
* **Folosirea unei librarii standard din Python: random**
* **Dezvoltarea unui proiect folosing Github.**

1. Primul pas este **crearea** unui **repository** pe Github (sugestie nume: **hangman\_game\_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. Reguli joc Spanzuratoarea:

* Acesta este un joc in care interactiunea cu utilizatorul se va face din terminal.
* Vom avea un fisier txt, in care vom avea mai multe cuvinte care pot fi ghicite intr-un joc (fiecare cuvant pe un rand separat)
* Cand incepe jocul, un cuvant va fi ales random din fisierul txt, si utilizatorul va trebui sa il ghiceasca.
* Utilizatorul va incerca sa ghiceasca cuvantul, ghicind literele din acesta.
* Pentru fiecare litera gresita, numarul de vieti va scadea cu 1.
* Utilizatorul are, ca valoare default, 6 vieti.
* In functie de rezultat, se vor afisa mesaje sugestive catre utilizator (daca s-a ghicit cuvantul, daca s-a terminat numarul de vieti etc)

1. Structura proiectului:

* Creeaza un fisierul in care vei adauga cuvinte ce poti fi ghicite in joc (fiecare cuvant pe rand separat) (sugestie nume fisier**: words.txt**)
* Logica jocului va fi facuta prin intermediul implementarii a **doua clase**: **WordGenerator**, respectiv **HangmanGame**) 🡪 creeaza un fisier in care vei avea logica pentru fiecare clasa (in total: 2 fisiere) (sugestie denumiri: **word\_generator.py**, **hangman\_game.py**)
* Creeaza un fisier **main.py**, care va fi **entrypoint-ul in aplicatie**. Aici vom creea obiectele necesare si vom da start jocului)

1. Implementare clasa **WordGenerator**:

* Va lua **in constructor un parametru**, care va reprezenta calea catre fisierul care contine cuvintele ce pot fi ghicite, **filename**. Seteaza pentru acest parametru, ca valoare default, calea catre fisierul txt ce contine cuvintele, din proiectul tau.
* In constructor, defineste un atribut pe obiect, numit **words**, care va primi ca si valoare rezultatul unei metode din interiorul clasei, numita **get\_all\_words**. Aceasta metoda va primi ca si argument, filename-ul luat in constructor ca parametru.
* Defineste metoda **get\_all\_words**:
  + Ia un parametru, **filename** – un string ce reprezinta calea catre fisierul ce contine cuvintele din joc.
  + Va deschide fisierul, va citi cuvintele din acesta, si le va pune intr-o lista pe care o va returna.
  + Daca fisierul care se incerca a fi deschis, nu e gasit, atunci o eroare va fi aruncata. Prinde acea eroare, si in acest caz, returneaza o lista goala si afiseaza mesajul “Error: {filename} not found”.
* Defineste metoda **get\_random\_word**:
  + Nu are parametri.
  + Verfica atributul words de pe obiect sa nu fie gol (cel care contine lista cu toate cuvintele).
    - Daca nu e gol, folosind functia choice din modulul random (ce trebuie importat), va alege un cuvant din lista si il va returna.
    - Daca e gol, se va returna un cuvant default ce poate sa fie folosit in joc (e la alegerea ta care va fi acesta).

1. Clasa **HangmanGame**:

* Constructor
  + Va lua in constructor un **parametru, max\_lives**, care va reprezenta numarul de vieti maxime pe care il are un jucator. Acesta va avea **valoarea default 6**.
  + Va avea urmatoarele atribute definite pe obiect:
    - **max\_lives** 🡪 valoarea va fi setata din parametrul ce vine din construtor.
    - **word\_generator** 🡪 atribut privat, va fi o instanta din clasa WordGenerator
    - **secret\_word** 🡪 atribut privat, va lua ca valoare rezultatul metodei get\_random\_word() apelata pe obiectul word\_generator, convertit la upper case.
    - **display\_word** 🡪 un string format din ‘\_’ \* lungimea lui secret\_word.
    - **guesses** 🡪 o lista goala
    - **lives\_left** 🡪 va fi egal cu max\_lives
  + Metoda **display\_game\_status()**:
    - Afiseaza display\_word
    - Afiseaza numarul de vieti ramase
    - Afiseaza literele ghicite
  + Metoda **guess\_letter()**:
    - Ia un parametru, **letter**
    - Se converteste letter la uppercase
    - Se verifica daca letter e deja in lista cu litere ghicite:
      * Daca da, se afiseaza mesajul “You’ve already guessed the letter {letter}. Try again”
      * Daca nu, se adauga litera in lista guesses.
    - Se verifica daca litera se afla in secret\_word:
      * Daca nu, se scade numarul de vieti ramase si se afiseaza mesajul “Incorrect guess”
      * Daca da, in display\_word, se inlocuieste underscore-ul cu litera respectiva, in fiecare loc in care apare si se printeaza mesajul “Good guess!”
  + Metoda **is\_game\_over():**
    - Returneaza True daca numarul de vieti ramase este 0 sau daca nu mai exista niciun “\_” in display\_word.
  + Metoda **play():**
    - Se printeaza un mesaj “Welcome to Hangman!”
    - Cat timp jocul nu e gata, se va afisa statusul jocului, se ia o noua litera de la utilizator/jucator, se verifica daca a ghicit-o sau nu (hint: folosim o structura repetitiva)
    - Se va afisa un mesaj catre utilizator, daca a reusit sa ghiceasca cuvantul sau daca a ramas fara vieti.
* **In fisierul main.py, apelam metoda play() definita in clasa HangmanGame si rulam programul.**