Projet POO NSI 2022

Projet modélisant un tableau de classment dynamiquye en python pour concours de type course automobile

- **Octobre / Novembre 2022**

Le projet

Nous avons choisi de faire un tableau d'affichage pour course automobile. Or, il se trouve que en automobile, tout est déterminé par le temps effectué mais le cahier des charges exige l'utilisation d'un calcul à un moment. C'est pourquoi nous avons choisi d'attribuer un nombre de points en fonction du temps effectué, auquel nous ajouterons un nombre de points donné selon les pénalités qui peuvent être également appliquées (collision, mauvaise trajectoire, etc...) ainsi, moins le concurrent à de points, mieux il est classé.

Notre code en quelques lignes

Pour ce qui est des classes, on retrouve donc une classe `Joueur` qui a pour attributs les différentes infos relatives au pilote telles que son nom, sa description, ses résultats, ses points et son état (attente ou course effectuée). On trouve également des méthodes pour retourner ces-dites infos ainsi que ses résultats et finalement une dernière pour afficher le "diplôme" mais qui reste facultative.

Nous avons ensuite la classe `Equipe` qui permet de gérer ces différents joueurs, d'en ajouter, de les lister et d'instancier avec la classe `Joueur`. Elle prend pour attributs les joueurs sous forme de liste et le nom du joueur à ajouter. Les différentes méthodes permettent ensuite respectivement d'afficher, de lister ou d'ajouter les diférents concurrents.

Puis vient la classe `Concours` qui prend pour attribut le classement sous forme de liste et à pour méthodes classement, qui permet de trier les différents présents dans `classement`, mais aussi une méthode affiche pour afficher ce classement ; on retrouve aussi modif pour modifier un élement recl pour reclasser un participant et saisieE pour ajouter une équipe au classement.

Notre méthode de calcul de points

Comme dit précédemment, la problématique pour déterminer le classement en allant plus loin qu'un simple tri en fonction du temps effectué à l'arrivée est de déterminer un nombre de points en fonction du temps effectué. Nous sommes donc parti du principe que 1min vaut 120 points nets soit 2 points par seconde. On se retrouve donc avec un équivalent points/temps qui nous permets de déterminer aisément le classement selon le temps net effectué. Cependant, nous avons décidé d'élaborer un peu plus en rajoutant un système de pénalités prenant en compte les collisions, les mauvaises trajectoires, les bacs, spins potentiels, etc... qui vont valoir plus ou moins de points en fonction de la gravité de l'infraction. Bien évidemment, plus une infraction vaut de points, plus elle vous fera baisser dans le classement et inversement car pour mieux visualiser ce système, une pénalité de l'ordre de 10 points serait donc comme rajouter 5 secondes à votre temps et une autre de 100 points comme rajouter 50 secondes ce qui est peu négligeable.
Pour ce qui est de la formule pour calculer ces points, elle ressemble donc à :

```
nombre_points = temps_en_secondes * 2 + pénalité
```

sachant que pénalité est une variable qui prendra en compte les différentes pénalités appliquées au joueur :

```
pénalité = type_penalite1 + type_penalite2 + type_penalite3 + ...
```

L'interface graphique

Pour ce qui est de l'aspect GUI, nous voulions à la base faire sous flask en mi-html mi-python mais il s'est avéré que la référence du programme étant Tkinter, mieux valait travailler dessus et se familiariser avec. L'idée est de faire des boutons pour pouvoir ajouter, modifier, supprimer les joueurs, les équipes puis au final afficher le classement.

Nous avons également eu l'idée de faire un système de sauvegarde tout simplement en marquant les données dans un fichier qui pourra ensuite être réimporté pour reprendre en cas de fermeture du prorgramme.
Pour faire cela, nous avions le choix entre différents moyens comme les fichiers textes, json ou encore csv. Nous avons finalement opté pour le csv car il est plus simple à manipuler et à lire sans compter le fait qu'il nous semblait être au programme.

L'idée est donc de faire un bouton sauvegarder pour sauver les données actuelles dans un fichier au format csv et les enregistrer dans le dossier `/save/your_name.csv` et un bouton importer/charger avec `askopenfilename` de la librairie `tkinter.filedialog` pour pouvoir choisir le fichier à importer.

La sctructure du projet

Le projet se décomposerait donc en un fichier `main.py` qui contiendra le code principal, un fichier `classes.py` avec nos différentes classes un dossier tkinter avec les différents fichiers relatifs à l'interface graphique et enfin un dossier `save` qui contiendra les fichiers de sauvegarde. On trouve également un fichier `launch.bat` qui permet de lancer le programme en un clic tout en évitant des erreurs de version et un fichier `requirements.txt` qui contient les différentes librairies à installer pour faire fonctionner le programme.

Aperçu :

```
~ Arborescence du projet ~
├── launch.bat
├── requirements.txt
├── main.py
├── classes.py
├── tkinter
│   ├── fenetre.py
│   ├── etc...
└── save
    ├── your_name.csv
    ├── autres sauvergardes potentielles
```

Répartition du travail

Pour ce qui est de la répartition du travail, Arthur s'est occupé de la classe `Joueur` et en partie du fichier principal `main.py`. Alan quand à lui s'est occupé de la classe `Concours` ainsi qu'en majorité du Tkinter. J-B s'est donc occupé de la classe `Equipe`, du fichier `main.py` rapidement, du `launch.bat` étant le seul familier avec ce language et surtout du docstring et de la documentation.
Après cette description donne les grandes lignes car nous avons tous travaillé sur les "parties des autres" étant donné que nous travaillions essentiellement par Github donc selon un cycle de coder quelque chose, commit, ensuite les autres voient et valident ou non, corrigent, etc... Ainsi au final chacun a travaillé même sur les parties des autres ou du moins a pu les voir et les comprendre.

Bouts de code à présenter

Alan

```
class Concours:
    def __init__(self, Equipe):
        self.classement = []

    def __classement__(self):
        self.classement.sort(key=lambda a: a[3])

    def __affiche__(self):
        return f"Classement : {self.classement}"

    def saisieE(self, Equipe):
        self.classement.append(Equipe)
```

- Partie tkinter en revue rapide

Arthur

```
class Joueur:
    def __init__(self, nom, description, resultats, points, etat):
        self.nom = nom
        self.description = description
        self.resultats = resultats
        self.points = points
        self.etat = etat

    def __str__(self):
        return f"Nom : {self.nom}\nDescription : {self.description}\nResultats :
{self.resultats}\nPoints : {self.points}\nEtat : {self.etat}"

    def resultJ(self):
        return f"Résultats : {self.resultats}"
```

- le fichier `main.py` avec les fonctions principales qui le composent

J-B

```
class Equipe:
    def __init__(self, nom):
        self.joueurs = []
        self.nom = nom

    def __str__(self):
        liste = []
        for i in self.joueurs:
            i.append(liste)
        return liste

    def listeJoueurs(self):
        for joueur in self.joueurs:
            print(joueur.nom)

    def ajouterJoueur(self, Joueur):
        self.joueurs.append(Joueur)
```

- En fonction du temps quelques fonctions principales du programme dans le fichier `main.py` ou interessantes en tkinter selon ce que les autres ont déjà eu le temps de présenter
- Peut-être rapidement le fichier batch mais fichier assez optionnel donc pas forcément nécessaire

---

Projet et compte rendu proposés par :

BROUILLET Arthur
CUNIN Alan
FROEHLY Jean-Baptiste