Projet : Le memory des nombres

Nombre de personnes : 4

Évaluable : Non

Durée du projet en classe : 3 semaines - 2h par semaine (lundi)

Date de rendu : 52 novembre 2022

Modalités de rendu : messagerie ENT

Le Memory des nombres est un jeu destiné aux enfants leur permettant d'entraîner leur mémoire et de se repérer dans l'espace.

Le jeu repose sur la recherche d'un nombre dans une matrice carrée. Les différentes étapes sont les suivantes :

- ▷ Au départ, toutes les cartes sont retournées, face contre la table.
- ▷ Une carte du tas est retournée au hasard.
- ▷ Le joueur doit indiquer la carte du tasqui est identique à celle qui vient d'être retournée.
- ▷ Si les deux cartes sont identiques, elles restent retournées, face au joueur,
- ▷ Si les deux cartes sont différentes, elles sont toutes les deux retournées face contre table.
- ▷ Le jeu s'arrête lorsque toutes les cartes sont face au joueur.

1 Énoncé général du besoin

On souhaite écrire un programme qui permet de jouer au jeu du Mémory.

2 Mode de fonctionnement

Le fonctionnement doit être entièrement automatique. L'interaction s'effectue via la console du système d'exploitation (`cmd` pour Windows© ou `terminal` pour Linux©).
L'exécution du fichier `memory.py` lance le jeu.

3 Travail préliminaire

Commencez par jouer à 2.

- ▷ Identifiez le rôle du joueur et celui de l'ordinateur
- ▷ Identifiez les problèmes potentiels
- ▷ Déduisez-en des limites que vous poserez par écrit.

4 Travail à réaliser

1. Identifiez les différentes actions permettant de répondre au cahier des charges. **Faîtes attention à ne choisir que des verbes d'action**. Écrivez les actions dans le désordre, au fur et à mesure que les idées viennent.
2. Pour chacune des actions :
   - ▷ Identifiez les entrées en vous posant la question « **que faut-il connaître pour exécuter l'action ?** » puis l'écrire sur la fiche cartonnée bleue et la scotcher devant l'action.
     - ○ quel est nom de la variable ?
     - ○ que contient-elle ?
     - ○ donnez-en un exemple
   - ▷ Identifiez les sorties en se posant la question « **que va faire l'action ?** » puis l'écrire sur la fiche cartonnée rouge et la scotcher derrière l'action.
     - ○ quel est nom de la variable ?
     - ○ que contient-elle ?
     - ○ donnez-en un exemple
   - ▷ Pour chaque entrée, indiquez si elle provient d'une interaction avec l'utilisateur ou si elle provient de la sortie d'une action.

3. Chaque action sera ensuite codée par une ou plusieurs fonctions, en respectant les principes suivants :
   - ▷ une fonction ne fait pas plus de 15 lignes (hors documentation, préconditions et postconditions) ;
   - ▷ pas de structures imbriquées (double boucle, test dans une boucle, boucle dans un test, ...).

Pour chacune des fonctions, vous indiquerez :

- ▷ la documentation complète comprenant :
  - ○ ce que fait la fonction
  - ○ la définition complète des arguments (nom, type, usage)
  - ○ deux ou trois exemples d'exécution.
- ▷ les assertions sur les préconditions.
- ▷ les assertions sur les postconditions.

Remarque

- ▷ Les interactions avec l'utilisateur seront placées dans une fonction dont le nom commence par `IHM_...()`
- ▷ Il est inutile de donner un exemple d'utilisation pour ces fonctions.

Annexe 1 - Implémentation Python

Respectez quelques règles [1] :

- ▷ nommez les variables en minuscules et la constantes en majuscules ;
- ▷ donnez des noms qui expriment le contenu (pour les variables et les fonctions) ;
- ▷ chaque fonction doit réaliser une seule tâche clairement identifiée ;
- ▷ limitez les fonctions à 15 lignes maximum, sauf dans des cas exceptionnels ;
- ▷ n'utilisez pas de variables globales ;
- ▷ évitez la redondance dans le code (copier/coller). Si cela arrive, c'est qu'il manque soit une fonction, soit une boucle, soit que des tests conditionnels peuvent être regroupés.

L'organisation de chaque fichier python respectera la présentation suivante :

```
# ================================================================================
# Import des modules nécessaires
# ================================================================================


# ================================================================================
# Définition des constantes
# ================================================================================


# ================================================================================
# Définition des fonctions
# ================================================================================


# ================================================================================
# Programme principal
# ================================================================================
#- A compléter -#
```

Arborescence attendue

```
/
└── Memory
    ├── fonctions_eleve_1.py
    ├── fonctions_eleve_2.py
    ├── fonctions_eleve_3.py
    ├── fonctions_eleve_4.py
    └── memory.py
```

1. https://lms.fun-mooc.fr/asset-v1:ulb+44013+session01+type@asset+block/BonnesPratiquesPython.pdf

Annexe 2 - Déroulement du projet

La réalisation du projet s'effectue en 6 étapes, dont les durées sont précisées, ainsi que leur mode de validation. Les durées sont indicatives. Elles tiennent compte d'un travail en classe **et d'un travail à la maison**.

APP : appropriation du projet ; CONCEP : conception détaillée ; COD : codage ; TEST : test de chaque fonction ; INT : intégration ; RENDU : Rendre compte

Étape	**Activités**	**Tâche**	**Durée**	**Validation**
APP	Découverte du projet, lecture du cahier des charges, étude de l'exemple, recherche de solutions, **répartition des tâches**.	Collective	2 heures	Professeur
CONCEP	Choix de la structure de donnée Rédaction de la documentation **COMPLÈTE** (voir 4), des préconditions et des postconditions.	Individuelle	30 min par fonction	Professeur
COD	Écriture du code Python de chaque fonction	Individuelle	30 min par fonction	
TEST	Test de chaque fonction - Aller-Retour à la phase **COD**age	Individuelle		doctest ou pytest
INT	Écriture du corps du programme principal Utilisation des fonctions, initialisation des variables, affichage du résultat.	Collective	6 heures	Observation (affichage à l'écran - interaction avec le joueur)
RENDU	Rendre l'ensemble du programme Python.	Collective		Professeur

Annexe 3 - Rendus intermédiaires

Tous les rendus intermédiaires s'effectuent via la messagerie de l'ENT. Ils serviront à compléter la grille d'évaluation.

10/10/2022	Actions ; entrées-sorties ; répartition des tâches.
17/10/2022	fichier python **INDIVIDUEL** avec : ▷ la documentation complète comprenant : ◦ ce que fait la fonction ◦ la définition complète des arguments (nom, type, usage) ◦ deux ou trois exemples d'exécution. ▷ les assertions sur les préconditions. ▷ les assertions sur les postconditions.
24/10/2022	fichier python **INDIVIDUEL** avec la documentation complète, les assertions sur les préconditions, les assertions sur les postconditions **ET** le code complet vérifiant les exemples.
31/10/2022	dossier **COLLECTIF** réunissant le travail des élèves **ET** le programme principal même s'il n'est pas complet.

Arborescence attendue

```
/
└── Memory
    ├── fonctions_eleve_1.py
    ├── fonctions_eleve_2.py
    ├── fonctions_eleve_3.py
    ├── fonctions_eleve_4.py
    └── memory.py
```

Remarque

Pour chaque groupe, un chef de projet sera clairement identifié. En plus de réaliser le travail qui lui est assigné, son rôle sera :

- ▷ de veiller au respect du timing de chaque étape,
- ▷ de veiller au respect des règles d'implémentation Python (voir annexe 1),
- ▷ de veiller au respect de la cohérence des entrées-sorties de chaque fonction, dans le respect des autres fonctions,
- ▷ d'aider les autres membres du groupe dans la rédaction de la documentation et dans l'élaboration du code de chaque fonction,
- ▷ de s'assurer, éventuellement par des tests supplémentaires, de la validité du code de chaque fonction,
- ▷ de réaliser le programme principal en important les fichiers `fonctions_untel.py` de chaque élève.

Annexe 4 - Critères d'évaluation

Critères d'évaluation		Note
Rédaction de la documentation	Complète et explicite : 3 Incomplète ou peu explicite : 1-2 Non fournie : 0	
Rédaction des exemples	Complète : 3 Partielle mais en quantité/qualité suffisante : 2 Partielle et en quantité/qualité insuffisante : 1 Non fournie : 0	
Rédaction des préconditions	Complète : 2 Partielle : 1 Non fournie : 0	
Rédaction des postconditions	Complète : 2 Partielle : 1 Non fournie : 0	
Écriture du code	Valide totalement le cahier des charges : 6 Valide majoritairement le cahier des charges : 4 Valide minoritairement le cahier des charges : 2 Ne valide pas le cahier des charges : 1 Code non fourni : 0	
Écriture du code	Le code est clair et structuré : 2 Le code est clair ou structuré : 1 Le code n'est ni clair ni structuré : 0	
Écriture du code du programme principal	Toutes les fonctions sont utilisées : 2 Quelques fonctions sont utilisées : 1 Aucune fonction n'est utilisée : 0	