TP IMMERSION 2020 PYTHON

Table des matières

1 Introduction

1.1 L'immersion

Ce TP d'immersion a été conçu pour vous donner un avant goût de ce à quoi vous attendre en entrant à EPITA. Vous y trouverez des exercices types que les étudiants de première année du cycle préparatoire font pour découvrir et apprendre la programmation. Vous êtes donc en quelque sorte déjà dans l'ambiance EPITA. La seule grosse différence entre les TPs que vous pourrez effectuer en première année et celui-ci est que vous allez ici découvrir Python, qui n'est pas un langage utilisé lors des TPs.

1.2 Les outils nécessaires

Pour pouvoir réussir au mieux ce TP, nous vous conseillons d'utiliser l'éditeur Visual Code Studio, qui est léger et très simple d'utilisation. Vous pouvez bien sûr utiliser tout autre type d'éditeur ou d'interpréteur si vous êtes familier avec. Nous vous conseillons également d'utiliser une version de Python au moins équivalente 3.7 pour éviter d'avoir des soucis de compilation. Si vous disposez d'une version antérieure, ne vous inquiétez pas, les chances de non-compilation sont faibles.

1.3 Visual Studio Code

Visual Studio Code (VSCode) est vraiment très simple d'utilisation, voici une présentation courte qui vous expliquera où retrouver les boutons de base pour travailler efficacement.

Tout d'abord, pour indiquer à VSCode que vous êtes en train de programmer en Python, vous devez créer un nouveau fichier via 'File -> New File', puis le sauvegarder avec 'File -> Save As...' en lui indiquant un nom suivi d'un extension ".py". Lorsque cela est fait, vous pouvez commencer à coder.

Attention !

Pour vérifier que vous disposez bien de la coloration syntaxique Python, écrivez "def " dans votre nouveau fichier. Si le mot-clé "def" se colore, tout est bon, sinon, suivez les prochaines étapes.

A gauche de votre zone de texte, cliquez sur le bouton "Extensions" (4 petits carrés) et recherchez "Python" dans la barre de recherche qui s'affiche. Installez l'extension et le tour est joué.

Pour exécuter votre programme vous pouvez appuyer sur la flèche verte en haut à droite de votre écran, ce qui devrait ouvrir une console en bas de votre écran où les différentes informations liées à votre programme s'afficheront. Dans cette console, vous apercevrez ce que renvoie la fonction print(), une fonction de Python permettant d'écrire dans la console toute information passée en argument, ou encore les erreurs de compilation de votre programme. Dans le cadre de ce TP, vous pourrez donc voir votre score, ou encore afficher vous même des informations liées à certaines fonctions qui vous seront à coder.

2 Explications

2.1 Introduction

Les prochains exercices nécessitent l'utilisation des boucles, c'est donc ce que nous allons vous présentation dans un premier temps, avant de vous introduire la fonction que vous aller devoir réaliser.

```
print("ceci est la ligne 1")
print("ceci est la ligne 2")
print("ceci est la ligne 3")
print("ceci est la ligne 4")
print("ceci est la ligne 5")
[...]
```

Lorsque vous voudrez programmer certaines fonctions, vous aurez besoin de répéter une instruction un certain nombre de fois. Au delà de l'évidente perte de temps causée par l'écriture d'un programme aussi inutile, le problème est le temps et la place que prend ce code. La portion de code dupliquée dans cet exemple ne fait qu'une seule ligne, mais pourrait en faire beaucoup plus. Vous pourriez également avoir plus de modifications qu'un simple numéro de lignes.

C'est pour cela que nous utilisons des boucles. Celles-ci permettent de remédier à deux problèmes :

Répéter une ou plusieurs actions tant qu'une condition est vraie. Ceci est la fonction de while.

Répéter une ou plusieurs actions un certain nombre de fois. Une variable est généralement utilisée comme compteur. Ce type de répétition est appelé itération en informatique ; Ceci est la fonction de for.

2.2 Les boucles

La boucle while

C'est la boucle la plus simple. Elle permet d'exécuter des lignes de code tant qu'une condition est vraie. La syntaxe ressemble à ça :

```
while (condition):
    # faire quelque chose
```

Ici, l'ordinateur comprend : tant que la 'condition' est vraie, répète les instructions indentées dans la boucle. Évidemment, si la 'condition' est fausse dès le début, on ne rentrera jamais dans la boucle. Voici un exemple basique de compteur en utilisant cette boucle :

```
counter = 0
while (counter < 10):
    print("EPITA c'est bien, mais c'est mieux avec " +
       counter + " de moyenne.")
    counter = counter + 1
```

La 'condition' est donc **counter < 10**. Vu que counter commence à 0, il va rentrer dans la boucle, afficher le texte, et incrémenter le compteur. Ainsi de suite, 10 fois.

Petite parenthèse sur le **counter = counter + 1** de l'exemple. On utilise des variables comme cela pour compter les tours de boucle qu'on a fait. Si on ajoute 1 chaque tour, on dit qu'on incrémente la variable et si on retire 1, on la décrémente. Comme vous pouvez vous en douter, vous allez en créer souvent et écrire **counter = counter + 1** à chaque fois n'est ni efficace ni beau à voir. Vous pouvez écrire **counter += 1** à la place, de même pour décrémenter une variable.

La boucle for

Cette boucle est un peu plus complexe que la boucle **while**, mais elle permet aussi de faire bien plus de choses qui ne nous serons pas utiles aujourd'hui. Voici à quoi ressemble une boucle for en Python :

```
for i in range (42):
    # Faire quelque chose
```

Ici, la variable **i** est créée lors du premier appel à la boucle et est initialisée à 0. 42 représente la borne maximum + 1 que peut atteindre **i**. En effet, dans cet exemple, **i** va de 0 à 41, soit 42 valeurs.

Variables globales

Pour utiliser des variables dans votre code, notamment dans le code de votre bot d'IA. Par exemple vous pouvez definir une variable current = 7 en dehors d'une fonction et l'utiliser dans une fonction en ajoutant une ligne au debut de votre fonction qui doit ressembler a ceci : global current

3 Exercices

Les fonctions se trouvent dans le fichier basics.py. Assurez vous d'avoir téléchargé le squelette du TP qui se trouve sur la même page que le sujet. Notez aussi que pour exécuter vos fonctions, il faut enlever le mot clé PASS des fichiers du squelette, ces derniers servent justement à passer les fonctions sans les exécuter. Maintenant vous avez les bases pour réussir les exercices qui vont sont proposés.

3.1 Perroquet

Le but de l'exercice est d'afficher la répétition d'une chaîne de caractères autant de fois que l'on veut. vous devez donc coder la fonction suivante :

```
def perroquet(count, string):
```

Voici un exemple d'exécution de la fonction :

```
Une fois de plus ne fait pas de mal.
Une fois de plus ne fait pas de mal.
Une fois de plus ne fait pas de mal.
Une fois de plus ne fait pas de mal.
Une fois de plus ne fait pas de mal.
```

Cet affichage a été réalisé grâce à l'appel suivant :

```
perroquet(5, "Une fois de plus ne fait pas de mal.")
```

3.2 Factorielle

Vous devez écrire la fonction renvoyant la factorielle du nombre n passé en entrée. Pour rappel :

$$\mathrm{n}\,! = \begin{cases} 1 & si\ n \leq 1 \\ n * (n-1)! & sinon \end{cases}$$

Vous devrez l'implémenter dans cette fonction :

```
def factorielle(n):
```

3.3 Fibonacci

Vous devez écrire une fonction qui prend en argument le rang d'un nombre de la suite de Fibonnaci 1, et renvoie ce nombre. Pour rappel, la suite de Fibonacci est définie comme ceci :

$$F_n = \begin{cases} n & si\ n \in [\![0..1]\!] \\ F_{n-1} + F_{n-2} & sinon \end{cases}$$

```
def fibonacci(n):
```

3.4 Plus ou Moins

Les fonctions se trouvent dans le fichier moreorless.py

Le but de cet exercice est de créer le jeu du plus ou moins. Le concept est plutôt simple, le joueur a un certain nombres d'essais pour deviner un nombre, à chaque essai, le programme indique si le nombre à trouver est inférieur ou supérieur au nombre donné. Nous avons déjà codé une partie du jeu pour vous simplifiez la tâche, mais il vous reste encore la partie la plus importante du jeu à réaliser !

3.4.1 Check number

Vous devrez réaliser la fonction **Check Number**. Cette méthode doit permettre de comparer le nombre donné par le joueur au nombre à trouver. Elle prend en paramètre 2 entiers, le nombre entré par le joueur et celui à trouver, elle doit retourner un booléen, indiquant si le nombre entré correspond, au nombre à trouver.

```
def checknumber(inputgiven, number):
```

De plus cette fonction doit afficher les indices pour trouver le nombre.

— Si le joueur a trouvé le nombre : La méthode doit afficher la phrase suivante :
Félicitation, le nombre était :
suivi d'un espace, du nombre à trouver, puis d'un retour à la ligne. Il faut ensuite que la fonction retourne True.

— Si le nombre entré par le joueur est inférieur au nombre à trouver : La méthode doit afficher en rouge la phrase suivante :
Le nombre correct est supérieur à
suivi d'un espace, du nombre entré par le joueur, puis d'un retour à la ligne. Il faut ensuite que la fonction retourne False ;

— Si le nombre entré par le joueur est supérieur au nombre à trouver : La méthode doit afficher en rouge la phrase suivante :
Le nombre correct est inférieur à
suivi d'un espace, du nombre entré par le joueur, puis d'un retour à la ligne. Il faut ensuite que la fonction retourne False.

3.4.2 Game

Vous allez maintenant développer la fonction **game**. Celle ci prend en paramètre le nombre cherché et le nombre d'essais disponibles. Son objectif est d'appeler la fonction checkNumber avec l'input de l'utilisateur tant que le nombre n'est pas trouvé (le résultat de checkNumber est à False). Pour demander un input à l'utilisateur en python, vous devez utiliser la fonction input() qui renvoi une chaine de caractères (string). Une deuxième fonction qui sera utile est la fonction int() qui permet de transformer une chaine de caractère en un entier. Mais attention, si l'input ne peut pas être converti en entier, Python va stopper le programme et lancer une erreur. Ce n'est pas le comportement attendu, il faut ici utiliser le try/except pour éviter de lancer les erreurs. Voici comment il fonctionne :

```
try:
    #use int() function
except:
    #just ignore and remove one try
```

3.5 Intelligence Artificielle

3.5.1 Introduction

Cette partie est la suite de l'exercice du plus ou moins. Vous allez devoir développer une "Intelligence Artificielle" (IA) qui joue à ce jeu. Son but serait de faire le plus gros score au final, c'est-à-dire de trouver le plus rapidement un nombre dans un intervalle avec un nombre d'essais donné. Bien entendu, vous n'allez pas créer une intelligence artificielle à proprement dit, car vous n'avez pas les capacités pour le faire, et car cela prendrait bien trop de temps. Cependant, vous pouvez réfléchir à des stratégies que vous pensez être efficaces pour trouver rapidement un nombre, et les implémenter dans une fonction à l'aide de conditions simples.

Par exemple, si vous trouvez pertinent de toujours choisir le nombre situé à la moitié de l'intervalle, vous pouvez le faire en définissant vous même les bornes de l'intervalle dans lequel vous jouez après chaque tour. Il y a de nombreuses manières de réfléchir à ce problème, que ce soit d'un point de vu mathématique ou simplement avec du bon sens. Attention, toute tentative de triche sera détectée.

Pour vous aider, dans votre fichier se trouve deux variables : lastPlay et lastInput qui s'actualisent à chaque nombre que vous envoyez au programme. lastPlay est égal à -1 si le nombre attendu est plus petit, 1 si le nombre attendu est plus grand et 0 si vous avez trouvé la solution. lastInput quant à lui contient le dernier nombre que vous avez envoyé. La fonction nextTurn possède un paramètre maxRange qui correspond à la borne supérieure de l'intervalle dans lequel le nombre à envoyer se trouve, la borne inférieure étant toujours 0. Il faut donc comprendre que le nombre attendu est compris entre 0 et maxRange.

Seule la fonction nextTurn sera appelée dans votre programme, elle doit renvoyer un nombre qui est celui que vous pensez être la solution du plus ou moins. Elle peut se servir d'autres fonctions et des données du fichier bot.py, imaginez des solutions.

3.5.2 A vous de jouer !

Cette partie est à coder dans le fichier bot.py impérativement

Choisissez la méthode qui vous semble la plus efficace pour trouver rapidement un nombre, cela sur de nombreux essais. Nous allons passer votre code dans une moulinette qui évaluera le nombre d'essais que vous mettez pour trouver un nombre en fonction de l'intervalle, pour au final donner un score qui apparaîtra sur un scoreboard sur le site de l'immersion !

Vous pouvez ajouter toutes les fonctions que vous le souhaitez pour aider votre programme à prendre les bonnes décisions mais **les imports de python ne sont pas autorisés**. Votre code **ne doit pas contenir de fonction print** non plus.

A vous de trouver la meilleure stratégie pour être le/la meilleur-e. N'hésitez pas à demander des conseils aux étudiants pour des indices où d'autres questions. Bonne chance !