Mini-projet : Le juste prix

Objectifs :

- Découvrir la méthodologie de projet
- Travailler en équipe : collaboration et répartition des tâches pour travailler individuellement
- Mettre en application la Séquence 1 (algorithmique et programmation Python)

But du jeu

Le joueur doit deviner un prix (entier), il fait des propositions et à chaque proposition on lui dit « c'est plus » ou « c'est moins ». Il gagne lorsqu'il a trouvé le juste prix.

Vous allez donc écrire un programme Python :

- dans lequel l'ordinateur choisit au hasard un nombre entier entre 0 et 100 (qui correspond à un prix) ;
- demande à l'utilisateur de deviner ce prix en le saisissant au clavier ;
- indique à l'utilisateur si « c'est plus » ou « c'est moins » ;
- se termine lorsque l'utilisateur a deviné le juste prix.

Contrainte supplémentaire : Le jeu devra contenir deux modes de jeu :

- Mode facile : le joueur a autant de tentatives qu'il souhaite et le jeu doit s'arrêter lorsque le joueur a trouvé le nombre mystère.
- Mode difficile : le joueur n'a que 10 essais pour trouver le nombre mystère ; si le joueur trouve en moins de 10 essais il gagne et la partie s'arrête, sinon il perd.

Comme il s'agit du premier mini-projet, celui-ci sera un peu guidé, de manière à vous montrer la méthodologie de projet.

Concevoir le mode facile

Je vais vous guider dans la conception du premier mode de jeu.

Avant de commencer un projet, il est nécessaire de procéder par étapes :

1) Réfléchir à la structure du programme à écrire (qu'est-ce que votre programme doit faire ?)
2) Pour en dégager les principales tâches de programmation à accomplir et se les répartir ;
3) Travailler individuellement sur la (ou les) tâche(s) à réaliser ;
4) Mutualiser régulièrement les avancées de chacun pour effectuer des ajustements si nécessaire (ce qui est en général le cas !) ;
5) Une fois chaque tâche de programmation effectuée, réunir les différents morceaux de code de chacun pour constituer le programme final ;
6) Analyser le produit final et envisager des améliorations (sans toutefois les mettre en œuvre si manque de temps).

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

Pour vous simplifier la tâche, je vais vous présenter les étapes 1 et 2. Ce sont les étapes primordiales dans la réussite d'un projet, si elles ne sont pas faites correctement, il y a peu de chance que le projet aboutisse. Je vous guiderai ensuite sur l'étape 3 et vous serez chargés d'accomplir le reste des étapes.

Etape 1 : Réfléchir à la structure du programme à écrire

Réfléchissons à la structure du mode facile (nombre illimité de tentatives).

Le programme devra être capable de réaliser les choses suivantes :

1. Choisir un nombre entier au hasard entre 0 et 100.
2. Récupérer les réponses du joueur.
3. Comparer la réponse du joueur au juste prix
4. Montrer au joueur où il en est dans la partie (« c'est plus », « c'est moins », « Félicitations ! »)
5. Terminer la partie quand le joueur a deviné le nombre mystère.

Première analyse :

- La fonctionnalité 1 : n'est pas bien compliquée en important la bibliothèque `random`.
- Les fonctionnalités 2, 3 et 4 : vont être répétées un certain nombre de fois inconnu à l'avance donc il faudra utiliser une boucle `tant que`.
- La fonctionnalité 5 : n'est pas bien compliquée puisque la fin de la partie sera gérée par la sortie de la boucle `tant que`.

Voici une version algorithmique du jeu en français pour vous rendre compte à quoi cela ressemble :

```
Choisir un nombre entier au hasard entre 0 et 100
Tant que (le joueur n'a pas deviné le juste prix) faire
        Récupérer la réponse du joueur
        Comparer la réponse du joueur au juste prix
        Montrer au joueur où il en est dans la partie
```

Comment gérer la boucle du jeu ?

La condition dans la boucle `tant que` est « le joueur n'a pas deviné le nombre mystère ». Elle se traduit assez naturellement par « la réponse du joueur est différente du juste prix ». Ainsi, si on appelle `rep` la variable contenant la réponse du joueur et `juste_prix` la variable contenant le nombre entier choisi au hasard à la première ligne, alors cette condition devient : `rep` $\neq$ `juste_prix`.

Il reste néanmoins un problème : au premier passage dans la boucle `tant que` la variable `juste_prix` a une valeur mais la variable rep n'existe pas encore puisque le joueur n'a pas encore répondu à ce stade. Pour régler cela, il suffit d'initialiser rep à une valeur différente de `juste_prix` de façon à rentrer dans la boucle `tant que` : on peut par exemple initialiser `rep` à -1.

L'algorithme en pseudo-code pourrait donc être le suivant :

```
juste_prix = randint(0,100)
rep = -1
tant que rep ≠ juste_prix :
    rep = recupere_reponse()
    compa = compare_reponse(rep, juste_prix)
    montre_au_joueur(compa)
```

Analyse des instructions dans la répétitive `tant que` :

- Les trois fonctionnalités (2, 3 et 4) ont été transformées en des fonctions ;
- Ces trois fonctions seront donc à définir par une spécification précise et par leur algorithme en pseudo-code ;
- C'est grâce à ce « découpage » en fonctions que vous allez pouvoir vous répartir facilement les tâches de programmation et travailler de manière assez indépendante.

Etape 2 : Dégager les principales tâches de programmation à accomplir et se les répartir

Ce travail vient d'être expliqué, on peut donc découper le travail à faire en trois tâches, chacune correspondant à une des fonctions `recupere_reponse`, `compare_reponse` et `montre_au_joueur`

Comme vous allez travailler à trois, chacun sera en charge de l'une des tâches, c'est vous qui choisissez.

Remarque : ici, elles sont assez similaires, mais dans un projet plus important, la répartition se fait en général selon les goûts et aptitudes de chacun.

Se coordonner sur la nature des entrées et des sorties de chaque fonction

Avant de démarrer le travail individuel, il reste une chose importante à effectuer : si vous observez bien le pseudo-code, vous devez constater que les fonctions `compare_reponse` et `montre_au_joueur` ont des paramètres qui sont des valeurs renvoyées par une autre fonction. Plus précisément :

- la fonction `compare_reponse` prend en paramètre la variable `rep` qui est la valeur renvoyée par la fonction `recupere_reponse` ;
- la fonction `montre_au_joueur` prend en paramètre la variable `compa` qui est la valeur renvoyée par la fonction `compare_reponse`.

Vous devez donc vous mettre d'accord tout de suite sur la nature des valeurs renvoyées ! Sinon, il y a de fortes chances que vos fonctions soient incompatibles, par exemple : celui en charge de la fonction `montre_au_joueur` considère que la variable `compa` est de type `int` alors que celui en charge de la fonction `compare_reponse` va renvoyer une variable de type `str` ; l'un des deux devra recommencer son travail car les fonctions seront incompatibles.

Comme c'est votre premier (mini-)projet, je vais vous imposer ces éléments dans la suite (voir la spécification des fonctions ci-après), mais à l'avenir ce sera à vous de faire ce travail.

Etape 3 : Travailler individuellement sur la (ou les) tâche(s) à réaliser

Tâche n°1 : la fonction `recupere_reponse`

Spécification :

- Entrées (= paramètres) : aucune
- Sortie (= valeur(s) renvoyée(s)) : un entier `r`
- Rôle : demander à l'utilisateur de saisir un entier compris entre 0 et 100 et renvoyer cette valeur
- Précondition : aucune
- Postcondition : `r` est l'entier saisit par l'utilisateur

Tâche n°2 : la fonction `compare_reponse`

Spécification :

- Entrées (= paramètres) : deux entiers `m` et `n` dans cet ordre
- Sortie (= valeur(s) renvoyée(s)) : une chaine de caractères `c`
- Rôle : comparez les valeurs des entiers `m` et `n`
- Précondition : aucune
- Postcondition : `c` vaut : `"="` si `m = n` ; vaut `"+"` si `m < n` ; vaut `"-"` si `m > n`

Tâche n°3 : la fonction `montre_au_joueur`

Spécification :

- Entrées (= paramètres) : une chaine de caractères `c`
- Sortie (= valeur(s) renvoyée(s)) : aucune
- Rôle : affiche « Félicitations ! » si `c = "="` ; affiche « C'est plus » si `c = "+"` ; affiche « C'est moins » si `c = "-"`.
- Précondition : la longueur de `c` vaut 1
- Postcondition : aucune

A vous de jouer !

Vous avez désormais toutes les cartes en mains pour terminer le mode facile.

Il vous reste également le mode difficile à concevoir par vous-même. Pour ne rien vous cacher, il suffira de modifier un peu ce qui aura été fait dans le mode facile.

N'oubliez pas de réunir ensuite les deux modes dans un même script Python, le joueur ayant la possibilité de choisir à travers un menu le mode qu'il souhaite.

Travail à rendre sous forme d'un document numérique :

- La répartition des tâches au sein du groupe ;
- Le pseudo-code de chaque fonction du mode facile ;
- Le code Python global (menu + mode facile + mode difficile) ;
- Une analyse du travail réalisé et des améliorations possibles.

Date de rendu : ?

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS