UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Outils Info. TP3: Boucles, Utilisation de Fonctions, π

Important : rendu sous forme d'archive

À la fin, vos réponses seront rendues dans une archive zip : cette archive contiendra un fichier `compte-rendu.txt` (avec votre **nom**, **groupe**, et les réponses aux questions précédées d'une apostrophe) et vos fichiers pythons.

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/outils-info/
zip -r tp3.zip tp3/
```

Important : mise en place

Comme pour tous les TPs :

- travaillez dans un dossier dédié au TP, lui même dans `~/outils-info`,
- créez un fichier `compte-rendu.txt` pour écrire la date, vos noms et les réponses aux questions précédées d'une apostrophe,
- créez un sous-dossier par exercice (sauf instructions contraires).

Exercice 1 – Encore des boucles `for` (5 minutes)

Q1) Écrivez un programme `troistrois.py` qui affiche les entiers de 10 (inclus) à 25 (exclus), de trois en trois, en utilisant une boucle `for` et la fonction `range()`.

Q2) Écrivez un programme `tructruc.py` qui demande 3 entiers (`debut`, `fin`, `pas`) à l'utilisateur et affiche les nombres de `debut` (inclus) à `fin` (exclus), par pas de `pas`.

Exercice 2 – Boucles `while` (10 minutes)

Rappel : pour cet exercice, travaillez dans un dossier `ex2` lui même dans le dossier `tp3`.

Voici un exemple de programme, qui affiche les nombres de 10 à 25 (inclus) pour illustrer la boucle while, et qui se lit « tant que i est inférieur ou égal à 25, afficher i et donner à i la valeur de i plus 1 ».

```
1 i = 10
2 while i <= 25:
3     print(i)
4     i = i + 1
```

'Q3) Écrivez un programme `add.py` qui demande 2 nombres réels (`plus` et `limite`) à l'utilisateur. Faites que ce programme initialise une variable (par exemple `s`) à la valeur 0. Le programme doit ensuite, à l'aide d'une boucle `while`, ajouter successivement à cette variable la valeur de `plus`, tant que `s` n'atteint pas `limite`. Que représente le nombre d'itérations de votre boucle (nombre d'itérations = le nombre de fois que la boucle est exécutée) ?

'Q4) Écrivez un programme `mul.py` qui demande 2 nombres réels (`mul` et `limite`). Faites que ce programme fasse comme `add.py` mais commence à une valeur de 1 et multiplie à chaque fois par `mul`. Que représente le nombre d'itérations de votre boucle ?

'Q5) Copiez `mul.py` en `mulcompte.py` et faites que le programme compte le nombre d'itérations et l'affiche à la fin, en utilisant une nouvelle variable `compte`. Que donne le programme quand on tappe `10` et `12345` ?

Exercice 3 – Estimation de π (méthode de Monte-Carlo)

La méthode de Monte-Carlo permet de résoudre certains problèmes de façon approchée, par une simulation. Nous allons calculer une approximation du nombre π par la méthode de Monte-Carlo.

Si on considère un cercle de rayon r, centré au point $(0,0)$. Ce cercle est inclus dans un carré de coté $2r$. L'aire du cercle est $\pi \cdot r^2$ et celle du carré $(2r)^2$. Le rapport entre les aires est donc $\frac{\pi r^2}{(2r)^2} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$. En tirant aléatoirement n points de manière uniforme dans le carré, en moyenne $n \cdot \frac{\pi}{4}$ de ces points seront à l'intérieur du cercle.

Nous allons donc créer un programme qui tire n points (les uns après les autres) aléatoirement dans le carré. Pour chaque point, on regardera s'il se trouve à l'intérieur du cercle (dont l'équation est $x^2 + y^2 < r^2$, où x et y sont les coordonnées du point tiré et r le rayon du cercle). Le programme devra compter le nombre m de points qui se sont retrouvés dans le cercle. Une approximation de π est alors $\pi \approx \frac{4m}{n}$.

Estimation de π

Nous allons créer progressivement ce programme, dans un fichier `estimation.py`.

Q6) Faites que votre programme demande à l'utilisateur les valeurs de r et n (à l'aide de la fonction `split`) et mettez ces valeurs dans des variables `r` et `n` (après les avoir converties).

Q7) Faites que le programme affiche `n` fois « `Tirage d'un point` » à l'aide d'une boucle `for`.

Q8) En important le module `random` et en utilisant la fonction `random()` de ce module (il n'y a pas d'erreur dans le sujet, le module et la fonction ont le même nom), à chaque itération de la boucle affectez à une variable `x` la valeur renvoyée par la fonction `random()` (qui ne prends aucuns paramètres). Faites que le programme affiche aussi la valeur de `x`.

'Q9) Quel est le type de la valeur renvoyée par `random()` ? Lancez votre programme quelques fois. À partir des valeurs affichées, dans quel intervalle sont les valeurs renvoyées par `random()` ?

Q10) Calculez maintenant `x` en multipliant la valeur de `random()` par 2 et en y soustrayant 1.

'Q11) En raisonnant et en lançant votre programme, dans quel intervalle est maintenant tirée la valeur de `x` ?

'Q12) Faites que votre programme tire au hasard et affiche deux valeurs (pour `x` et `y`) entre $-r$ et r. Quelle formule avez vous utilisée ? Lancez votre programme pour vérifier que cela fonctionne.

NB : la fonction `uniform(a, b)` du module `random` permet de faire plus simplement ce que nous venons de faire pour avoir une valeur entre $-r$ et r.

Q13) Mettez en commentaire les `print` qui se trouvent dans la boucle.

'Q14) Comment exprimer en python la condition « le point de coordonnées x,y est dans le cercle de rayon r » ?

Q15) Faites que votre programme utilise la technique de l'accumulateur pour calculer `m` (le nombre de points qui tombent dans le cercle) et affiche au final l'estimation de π.

'Q16) Lancez votre programme plusieurs fois avec des valeurs de r et n différentes. Rapportez dans le compte rendu les valeurs que vous avez utilisées et les résultats obtenus (et une indication du temps que le programme mets à calculer).

Affichage

Q17) Copiez le programme précédent en tant que `affichage.py` et modifiez dorénavant ce fichier. Pensez à bien tester votre programme à chaque fois.

Q18) Sans changer le comportement par défaut (quand l'utilisateur tape 2 valeurs), et en utilisant la fonction `len`, faites que votre programme fonctionne aussi si on tape seulement 1 nombre. Dans ce cas la valeur tapée est celle de n, et la valeur de r est fixée à 300.

Q19) Importez la bibliothèque `qtido` (cf td2 ou site du cours) et faites que votre programme crée une fenêtre de 600 pixels par 600 pixels (avant la boucle) et qu'il attende que l'utilisateur ferme la fenêtre pour se terminer (après la boucle).

Q20) À chaque tirage de valeurs de `x` et `y`, affichez un cercle en $300 + x$, $300 + y$, de rayon 0.5 (vous ajoutez 300 pour que le centre du cercle soit au centre de la fenêtre).

Q21) Faites que la couleur du tracé soit du rouge si le point est dans le cercle et du vert sinon.

Q22) Faites que votre programme, toutes les 1000 itérations (quand le numéro de l'itération est multiple de 1000), appelle la fonction `re_afficher` de qtido et affiche l'estimation courante de π.