Informatique - TP 4

Utilisation des boucles `while`

M. Marmorat, M. Morel

16 Novembre 2023

Remarque 1. Dans tout ce TP, vous éviterez d'utiliser la fonction `print` au sein d'une fonction. Lorsque l'on vous demandera de coder une fonction **renvoyant** une certaine quantité, il faudra utiliser l'instruction `return`. Merci !

Exercice 1 Relisons le cours

Q1 On rappelle la syntaxe d'une boucle `while` :

```
while condition:
    #bloc d'instructions
#instructions hors de la boucle while
```

Que se passe-t-il si la condition `condition` n'est pas vérifiée ?

Q2 Quelles différences existent entre une boucle `while` et une boucle `for` ?

Q3 Que se passe-t-il à l'exécution du script suivant ?

```
compteur = 20
while compteur > 10:
    print(compteur)
    compteur += 1
```

Pourquoi ?

Remarque 2. Pour arrêter une boucle infinie, faites un clic droit sur le shell et cliquez sur "Interromptre".

Exercice 2 Retour sur la factorielle

Q1 Implémenter la fonction `seuil_factorielle` vue en cours, qui prend en argument un nombre réel positif M et qui **renvoie** le plus petit nombre entier n tel que $n! \geq M$.

Q2 Rappeler (en calculant par une méthode de votre choix, avec votre intelligence ou avec Python) les valeurs de $n!$ pour n allant de 0 à 7.

Q3 Vérifier le fonctionnement de votre fonction `seuil_factorielle` sur les nombres d'entrée $M = 5$, $M = 530$ et $M = 1000$.

Exercice 3 A partir d'un certain rang

On considère la suite (u_n) définie par $u_0 = 0$ et

$$u_{n+1} = u_n + \sqrt{n}.$$

Q1 Écrire une fonction `suite` qui prend en argument un entier n et qui renvoie le nombre u_n (on utilisera une boucle `for` ; pour utiliser la fonction $\sqrt{\cdot}$ on pourra l'importer grâce à l'instruction

```
from math import sqrt
```

que l'on placera au tout début du script).

Q2 Écrire une fonction `seuil` qui prend en entrée un nombre M et qui renvoie le plus petit entier n tel que $u_n \geq M$ (on utilisera la fonction `suite` de la question précédente. On vérifiera que `seuil`(30)=13.

Q3 Si l'on s'intéresse à l'efficacité de la fonction `seuil`, était-t-il judicieux d'appeler la fonction `suite` ? Pourquoi ? Proposer une implémentation plus efficace et vérifier que les résultats sont les mêmes.

Exercice 4 Des sauts de puce

Imaginons une puce se déplaçant aléatoirement sur une ligne allant soit en avant soit en arrière (pas de 1 ou -1). Par exemple, si cette dernière est à l'emplacement 2, elle peut sauter aléatoirement à l'emplacement 1 ou 3.

Q1 Avec une boucle `while` , écrire une fonction simulant le mouvement de cette puce de l'emplacement initial 0 à l'emplacement final 5. La fonction prendra en argument l'emplacement d'arrivée et renverra le nombre de sauts effectués par la puce.

Remarque 3. Pour simuler le comportement aléatoire, on utilisera

```
import random # A positionner en debut de script
x = random.choice([-1,1]) # A positionner dans votre script
```

Après ce script, la variable `x` a 50% de chances d'être égale à -1, 50% de chances d'être égale à 1.

Q2 Combien de sauts sont nécessaires pour réaliser ce parcours ? Obtient-on le même résultat à chaque fois ?

Q3 Aurait-on pu réaliser cette simulation avec une boucle `for` ?

Q4 Écrire un programme affichant les résultats de plusieurs essais de la simulation. Calculer le nombre moyen de sauts effectués sur ces essais.

Exercice 5 Approximation de π

La formule de Brent-Salamin (des noms de deux mathématiciens des années 1970) permet d'obtenir rapidement une bonne approximation de π (elle fut utilisée en 1999 pour obtenir plus de 206 millions de décimales de π !).

Elle consiste à définir $a_0 = 1$, $b_0 = 1/\sqrt{2}$, $t_0 = 1/4$ et $p_0 = 1$ puis pour tout $n \geq 0$,

$$t_{n+1} = t_n - p_n \left(\frac{a_n - b_n}{2}\right)^2, \quad a_{n+1} = \frac{a_n + b_n}{2}, \quad b_{n+1} = \sqrt{a_n b_n}, \quad p_{n+1} = 2p_n.$$

et affirme que lorsque a_n et b_n sont proches, la valeur

$$\frac{(a_n + b_n)^2}{4t_n}$$

est une bonne approximation de π.

Écrire une fonction `approx` prenant en argument un réel `epsilon` et renvoyant l'approximation de π obtenue par cette méthode lorsque l'on s'arrête dès que $|a_n - b_n| \leq$ `epsilon`.

Testez votre fonction, puis écrivez une fonction `approx_bis` pour savoir également combien d'itérations ont été nécessaires pour obtenir le résultat.

Exercice 6 La suite de Syracuse

Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction définie par

$$f(n) = \begin{cases} \dfrac{n}{2} \text{ si } n \text{ est pair,} \\ 3n+1 \text{ si } n \text{ est impair.} \end{cases}$$

On appelle suite de Syracuse de l'entier N la suite (u_n) définie par $u_0 = N$ et $\forall n \in \mathbb{N},\ u_{n+1} = f(u_n)$.

Q1 Calculer à la main les termes de la suite de Syracuse de l'entier 3. Que se passe-t-il lorsque la suite atteint la valeur 1 ?

Remarque 4. La conjecture de Syracuse affirme que toutes les suites de Syracuse des entiers positifs atteignent la valeur 1 au bout d'un certain temps. Cette conjecture a été vérifiée pour tous les entiers naturels N inférieurs à 2^{62}, mais on ignore encore si elle est vraie.

Q2 Écrire une fonction f prenant en argument un entier k et renvoyant $f(k)$.

Q3 Ecrire une foncton `syracuse` prenant en argument deux entiers N et n et renvoyant le nombre u_n, où la suite (u_n) est la suite de Syracuse de l'entier N. On vérifiera que `syracuse`(15, 9)=40.

Q4 Ecrire une fonction `temps_de_vol` prenant en argument un entier N et renvoyant le premier entier n tel que $u_n = 1$, où la suite (u_n) est la suite de Syracuse de l'entier N. On vérifiera que `temps_de_vol`(15)=17.

Q5 Tracer les valeurs de `temps_de_vol`(N) pour $1 \leq N \leq 1000$.

Remarque 5. Pour tracer des graphiques :

— Taper l'instruction

```
import matplotlib.pyplot as plt
```

au tout début du fichier.

— On pourra dessiner les points du graphique un par un en utilisant la commande `plt.plot(x,y,'bo')` pour tracer un point bleu aux coordonnées (x, y).

— Utiliser la commande

```
plt.show()
```

à la toute fin du script.