Boucles for — TP3

Exercice 1 — La fonction `range`

Remarque

On rappelle qu'on dispose d'une fonction `range` telle que : si `a,b,r` sont des entiers alors :

- `range(a,b)` =
- `range(b)` =
- `range(a,b,r)` =

On rappelle aussi que la fonction `print` permet d'afficher la valeur d'une variable.

Q1 Écrire un programme (pas une fonction...) affichant tous les nombres entiers de 6 à 11 inclus.

Q2 Écrire un programme affichant tous les nombres impairs entre 101 et 121 (inclus !).

Q3 Écrire un programme affichant tous les nombres multiples de 5 de 20 à -20 (inclus !). On respectera bien ici le sens de l'affichage souhaité : on part de 20 pour aller à -20 !

Exercice 2 — Proche du cours

Q1 Écrire une fonction permettant de calculer pour $n \in \mathbb{N}$ la valeur de $S_n = \sum_{k=3}^{n+1} \sqrt{2k+1}$. Vérifiez le résultat de votre fonction en calculant S_4 à la main.

Q2 Écrire une fonction `fact` qui prend en argument un entier naturel `n` et qui renvoie la valeur de la *factorielle* de `n` c'est-à-dire le nombre $1 \times 2 \times \cdots \times (\texttt{n}-1) \times \texttt{n}$. On note ce nombre $n!$. Testez votre fonction.

Q3 Écrire une fonction prenant en argument un réel x et un entier naturel n et renvoyant la valeur de $\sum_{k=0}^{n} \frac{x^k}{k!}$. On réutilisera la fonction `fact`. Testez votre fonction.

Q4 Écrire une fonction prenant en argument un entier naturel n et renvoyant le terme u_n de la suite définie par : $u_0 = 2$ et $\forall n \in \mathbb{N},\ u_{n+1} = 2u_n + \frac{1}{u_n}$. On vérifiera que u_{10} vaut environ $2379,37$.

Q5 Même question pour la suite (v_n) définie par $v_1 = 1,4$ et : $\forall n \in \mathbb{N}^*,\ v_{n+1} = v_n(3 - v_n)$. On vérifiera que v_{10} vaut environ $2,1626$.

Exercice 3 — Proche du cours bis

Q1 Écrire une fonction prenant en argument $n \in \mathbb{N}$ et renvoyant u_n où $(u_n)_{n\geq 0}$ est la suite donnée par : $u_0 = 3$ et $\forall n \in \mathbb{N},\ u_{n+1} = \frac{u_n + 3n}{2n+1}$. On vérifiera que u_5 vaut environ $1,501587$.

Q2 Écrire une fonction prenant en argument $n \in \mathbb{N}^*$ et renvoyant v_n où $(v_n)_{n\geq 1}$ est la suite donnée par : $v_1 = 1$ et $\forall n \geq 2,\ v_n = (v_{n-1} + n)^2$. On vérifiera que v_5 vaut $480\,004\,281$.

Q3 Écrire une fonction prenant en argument $n \in \mathbb{N}$ et renvoyant $S_n = \sum_{k=0}^{n} \frac{k}{2k+n+1}$. On vérifiera que S_{10} vaut environ $2,30543$.

Exercice 4 Suite et somme

On considère la suite $(u_n)_{n\geq 0}$ définie par $u_0 = 1$ et $\forall n \in \mathbb{N},\ u_{n+1} = 3\sqrt{u_n} + 1$.

Q1 Écrire une fonction `suite` qui prend en argument un entier naturel `n` et qui renvoie la valeur de $u_{\mathtt{n}}$. (Tester votre fonction sur de petites valeurs de `n`).

Q2 En déduire une fonction `somme`, utilisant la fonction `suite`, qui prend en argument un entier naturel `n` et qui renvoie la valeur de $\sum_{k=0}^{\mathtt{n}} u_k$. (Tester votre fonction sur de petites valeurs de `n`).

Q3 Combien de fois Python calcule-t-il la valeur de u_1 pour évaluer `somme(5)` ?

Q4 Écrire une fonction `somme_bis` qui renvoie le même résultat que `somme` mais sans le défaut évoqué à la question précédente.

Q5 La fonction `somme_bis` devrait être plus "efficace" que la fonction `somme`. Pour le constater, on peut chronométrer le temps que met Python à faire un calcul. Pour chronométrer le temps d'exécution d'une commande on utilisera la syntaxe suivante :

```
import time # À positionner en début de script
start_time = time.time()
# Vos instructions
print("--- %s seconds ---" % (time.time() - start_time))
```

Constater que `somme_bis` est plus rapide que `somme` sur le calcul de u_{100} ou u_{1000}.

Exercice 5 Évolution d'une population de bactéries

On cultive des bactéries en laboratoire et on sait que chaque jour le nombre de bactéries augmente de $\tau\,\%$ où τ est un nombre réel fixé. On appelle B_0 le nombre de bactéries au début de l'expérience, et B_n le nombre de bactéries après n jours.

Q1 (Sans ordinateur). Écrire la relation entre B_n, B_{n+1} et τ.

Q2 Écrire une fonction `population` prenant en arguments le nombre initial de bactéries, le taux de croissance τ, et le nombre `n` d'années considérées. Cette fonction renverra le nombre de bactéries après `n` jours.

Q3 Vérifier que pour une population initiale de 425 bactéries et une croissance de 1.94% par jour, on compte 756 bactéries après 30 jours.

Exercice 6 Primalité

Un entier naturel non nul est dit premier lorsque ses seuls diviseurs positifs sont 1 et lui-même. On rappelle que la commande `a%b` renvoie le reste dans la division euclidienne de `a` par `b`, c'est-à-dire que $r =$ `a%b` lorsque $r \in [\![0, \mathtt{b}-1]\!]$ et que $\mathtt{a} = d\,\mathtt{b} + r$ pour un certain entier d.

Q1 Écrire une fonction `divisible` prenant en arguments deux entiers a et b et renvoyant `True` si a est divisible par b et `False` sinon. On prendra garde à ne pas écrire quelque chose comme :

```
if booleen :
    return True
else :
    return False
```

Q2 Écrire une fonction `premier` prenant en argument un entier et renvoyant `True` s'il est premier et `False` sinon. Testez votre fonction, en particulier avec la valeur 25.

Q3 Écrire un programme Python affichant tous les nombres premiers inférieurs à 100.