TP2 : Structures itératives et suites récurrentes

Instructions introduites dans ce TP : `input, \% //.`
On commencera par repenser aux conseils du premier TP sur :

- le fait de bien lire, lors d'erreur, le type d'erreur que Python renvoie ;
- le fait de réfléchir sur papier pour les différents exerices ;
- le fait d'enrergistrer régulièrement son travail avec un nom de la forme `VOTRENOM_TPX.py`.

Exercice TP2.1

1. Tester dans la console les instruction `15%3`, `15//3`, `15//4`, `15%3`. Expliquer le fonctionnement des commandes `//` et `%`.
2. Écrire une fonction `g(n)` dépend d'un entier n définie par $g(n) = \begin{cases} \frac{n}{2} & \text{si } n \text{ est pair} \\ \frac{n-1}{2} & \text{si } n \text{ est impair.} \end{cases}$.

Exercice TP2.2

1. Déclarer une fonction qui attend en entrée un entier $n > 0$, puis renvoie la quantité :
$$v_n = \prod_{k=0}^{n-1} \left(1 + \frac{1}{k+1}\right).$$
On rappelle que si $u_1, \ldots, u_n$ sont des complexes, le produit de u_1,...,u_n est noté $\prod_{k=1}^{n} u_k$. Tester votre programme en vérifiant que l'on obtient $v_2 = 3$.
2. Quelle valeur votre fonction renvoie-t-elle si vous lui donnez l'entier $n = 0$? Pouvez-vous expliquer pourquoi ? Cela vous semble-t-il judicieux ?
3. Calculer v_{100}, v_{1000} puis v_{10^4}. Qu'est-ce que ces valeurs nous incitent à penser ?

Exercice TP2.3

1. Déclarer une fonction `sommation` qui prend en entrée une fonction g ainsi qu'un entier n, puis qui renvoie la valeur de $\sum_{k=0}^{n-1} g(k)$.
2. En déduire la valeur de la somme des 3 premiers nombres impairs, puis des 100 premiers nombres impairs.
3. En déduire la valeur de $u_n = \sum_{k=0}^{n-1} \frac{4(-1)^k}{2k+1}$, pour $n = 100$ puis $n = 10^5$. Qu'est-ce que ces valeurs nous incitent à conjecturer ?

Exercice TP2.4

On considère la suite $(u_k)_{k\in\mathbb{N}}$ définie par $u_0 = u_1 = 1$ et $u_{k+1} = \frac{u_k + 2}{k}$ pour tout entier $k > 0$.

1. Écrire un programme qui prend en entrée un entier n puis renvoie la valeur de u_n. Tester votre programme en vérifiant que les valeurs de u_0, u_1 et u_2 sont bien celles attendues.
2. À l'aide du programme précédent, essayer de deviner la limite de $(u_n)_{n\in\mathbb{N}}$.
3. Écrire un programme qui prend en entrée un entier p puis renvoie la valeur de $v_p = \sum_{n=0}^{p} u_n$. Vérifier que votre programme donne le résultat attendu.
4. À l'aide du programme précédent, essayer de deviner la limite de $(v_p)_{p\in\mathbb{N}}$.

Exercice TP2.5

On appelle *suite de Fibonacci* la suite $(F_n)_{n\in\mathbb{N}}$ définie par $F_0 = 0$, $F_1 = 1$ et pour tout entier n :

$$F_{n+2} = F_{n+1} + F_n.$$

Écrire une fonction `Fibonacci` qui prend en entrée un entier n puis renvoie le n-ième terme F_n de la suite de Fibonacci (comme toujours on testera sa fonction pour vérifier qu'elle donne bien les valeurs attendues).

Exercice TP2.6

Écrire une fonction qui prend en entrée deux entiers strictement positifs n, p et renvoie leur plus grand diviseur commun (ou PGCD).

Exercice TP2.7

1. Écrire un programme qui calcule les 50 premiers multiples de 13, mais n'affiche que ceux qui sont des multiples de 7.
2. Écrire un programme qui affiche à l'écran les 50 premiers multiples de 13 qui sont des multiples de 7 (on commencera par bien comprendre quelle est la différence avec la question précédente).

Exercice TP2.8

1. On donne le programme suivant :
   ```
   def test():
       a = input('Entrer un objet')
       return a
   ```
   Tester ce programme et expliquer son fonctionnement. Taper dans la console `>> b = test()`, entrer un objet puis déterminer le type de `b`.
2. Construire un programme `bienvenue()` qui ne prend pas d'argument, mais qui demande le prénom et l'âge de l'utilisateur (avec la fonction `input`), puis renvoie suivant ces données un message de bienvenue du style :

   'Bonjour Damien, vous avez 18 ans'