PYTHON : BOUCLES ET BRANCHEMENT CONDITIONNEL
BILAN DE COMPÉTENCES

1 BILAN DE COMPÉTENCES

- À ce stade de l'année, vous devez savoir programmer sans hésitations :
  - **compétence 1** : le terme n d'une suite récurrente d'ordre 1, 2 ou plus.
  - **compétence 2** : le premier terme d'une suite qui vérifie une condition (par exemple $u_n \geq M$).
  - **compétence 3** : le dernier terme d'une suite qui vérifie une condition (par exemple $u_n < M$).
  - **compétence 4** : calculer une somme ou un produit.
  - **compétence 5** : tester une conjecture sur les termes d'une suite.

2 EXERCICES À SAVOIR REFAIRE

Les exercices suivants mettent en œuvre ces compétences. Ils sont à travailler en TD, puis chez vous. Vous devrez savoir les refaire rapidement sur ordinateur ou sur papier.

EXERCICE 1 (Une suite récurrente d'ordre 1)

On définit la suite (u_n) par

$$u_1 \in \mathbb{R} \qquad \text{et} \qquad \forall n \geq 1,\ u_{n+1} = \sqrt{1 + n\,u_n^2}$$

1) (**compétence 1**) Écrire une fonction **suite**(`u1,n`) qui renvoie le terme u_n de la suite correspondante.

2) (**compétence 2**) Écrire une fonction **suiteRang**(`u1,M`) qui renvoie le premier rang à partir duquel $u_n \geq M$.

3) (**compétence 2**) Écrire une fonction **suite2**(`u1,M`) qui renvoie le premier terme de la suite (u_n) supérieur ou égal à M.

4) (**compétence 3**) Écrire une fonction **suite3**(`u1,M`) qui renvoie le dernier terme de la suite (u_n) inférieur strict à M.

EXERCICE 2 (Suite récurrente d'ordre 2) On définit la suite (u_n) par

$$(u_0,\, u_1) \in \left(\mathbb{R}_+^*\right)^2 \qquad \text{et} \qquad \forall n \in \mathbb{N},\ u_{n+2} = \sqrt{u_n u_{n+1}}$$

1) (**compétence 1**) Écrire une fonction **geom**(`u0,u1,n`) qui renvoie le terme u_n de la suite.
(Utiliser l'affectation par couples.)

2) (**compétence 4**) Écrire une fonction **somme**(`u0,u1,n`) qui renvoie la valeur de la somme $\displaystyle\sum_{k=0}^{n} u_k$

3) (**compétence 4**) Écrire une fonction **produit**(`u0,u1,n`) qui renvoie la valeur du produit $\displaystyle\prod_{k=0}^{n} u_k$

EXERCICE 3 (Conjectures)

```
def mystere(u0,n):
  u = u0
  for i in range(n):
    u = log(1+u)
  return u
```

1) Que simule l'algorithme précédent ? Comment faut-il choisir `u0` (conjecture) ?

2) (**compétence 5**) Écrire une fonction **conjecture**(`u0,n`) qui renvoie **True** si la suite définie par `mystere` est décroissante jusqu'au rang n, et **False** sinon.

3) Donner une conjecture sur la nature de la suite, et sur la valeur de son éventuelle limite.

4) (*À faire chez soi*), prouver les conjectures.

3 ENTRAINEMENT

EXERCICE 4

Programmer un algorithme en Python qui calcule le n-ième terme de la suite définie par

$$u_0 = 1,\ u_1 = 1 \quad \text{et} \quad \forall n \in \mathbb{N}, \quad u_{n+2} = \sum_{k=0}^{n+1} \frac{u_k}{u_n + u_{n+1}}$$

EXERCICE 5 (Crible d'Érathostène)

On considère la fonction suivante :

```
def premier(n):
  p = True
  for k in range(2,int(sqrt(n))+1):
    if n%k == 0:
      p = False
  return p
```

1) Expliquer ce que fait cette fonction.

2) Tester **premier**(`10**15`). Qu'en pensez-vous ?

3) Donner le $10001^{\text{ème}}$ nombre premier. (le premier est 2, le deuxième est 3,... le sixième est 13...).

Indication : Vous aurez besoin d'optimiser la fonction pour que le calcul se fasse en un temps raisonnable (doit donner le résultat presque immédiatement).

4) Programmer une fonction **listePremiers** qui prend en argument un nombre entier $n \geq 2$ et qui renvoie la liste des nombres entiers premiers inférieurs ou égaux à n.

On fera appel à la fonction **premier** programmée juste avant.

5) Programmer une fonction **listePremiers2** qui renvoie le même résultat, mais à partir du crible d'Eratosthène.