TP # 6

Etude de suites réelles (partie 2)

Suites à récurrence double

Considérons le cas classique des suites à récurrence linéaire d'ordre 2 :

$$\boxed{u_0 = x, \quad u_1 = y \quad \text{et} \quad \forall n \in \mathbb{N}, \; u_{n+2} = a u_{n+1} + b u_n.}$$

En Python, on aura besoin de travailler avec deux variables (nommées ici `u` et `v`), qui contiendront deux termes consécutifs de la suite.

Pour calculer et afficher le terme u_n (pour un $n \in \mathbb{N}$ donné) :

```
u = x  # valeur de u0
v = y # valeur de u1

for k in range(n) : # on passe n fois dans cette boucle

        w = a*v + b*u # calcul de la prochaine valeur
        u = v # mise a jour de u
        v = w # mise a jour de v

print(u) # apres n passages, on affiche ou on renvoie le resultat
```

On peut se convaincre que, dans ce programme :

- Avant la boucle, `u`= ...... et `v`= ......
- Après 1 passage, `u`= ...... et `v`= ......
- Après 2 passages, `u`= ...... et `v`= ......

⋮

- Après n passages, `u`= ...... et `v`= ....

Cette méthode fonctionne bien-sûr avec des relations de réccurence double plus générales, de la forme $u_{n+2} = f(u_{n+1}, u_n)$.

Par exemple, pour $u_0 = 1, \; u_1 = 2 \;$ et $\; \forall n \in \mathbb{N}, \; u_{n+2} = \dfrac{u_n}{2 + u_{n+1}}$:

```
u = 1  # valeur de u0
v = 2 # valeur de u1

for k in range(n) : # on passe n fois dans cette boucle

        w = u / (2 + v) # calcul de la nouvelle valeur
        u = v # mise a jour de u
        v = w # mise a jour de v

print(u) # apres n passages, on affiche ou on renvoie le resultat
```


Exercice 1

On définit la suite de Fibonacci de la manière suivante :

$$u_0 = 0, \; u_1 = 1 \;\; \text{et} \;\; \forall n \in \mathbb{N}, \; u_{n+2} = u_{n+1} + u_n.$$

Définir une fonction `fibo` qui prend en entrée un entier `n` et renvoie le terme u_n.

Compléter la valeur : $u_{15} =$ ....

Somme des premiers termes d'une suite récurrente

On revient à une suite à récurrence simple : $\boxed{u_0 = x \quad \text{et} \quad \forall n \in \mathbb{N}, \; u_{n+1} = f(u_n, n)}$

Pour calculer et afficher la somme $\displaystyle\sum_{k=0}^{n} u_k$:

```
S = 0 # valeur initiale de la somme
u = x # valeur initiale de la suite

for k in range(n+1) : # k = 0,1,2...,n

        S = S + u # on ajoute la valeur u dans la somme
        u = f(u,k) # on "met a jour" u

print(S) # on affiche ou on renvoie le resultat
```

Remarque 1

Si on veut calculer le produit $\displaystyle\prod_{k=0}^{n} u_k$, on peut changer "`S = 0`" et "`S = S + u`" en "`P = 1`" et "`P = P * u`".

Exercice 2

Soient $q \in \mathbb{R}$ et $n \in \mathbb{N}$ fixés.
Comprendre la somme calculée par le programme suivant.
(Inutile de taper ce programme !)

```
S = 0 ; u = 1

for k in range(n) :

        S = S + u
        u = q * u

print(S)
```

- Avant la boucle, on a : `S` = ... et `u` = ...
- Après 1 passage, on a : `S` = ... et `u` = ...
- Après 2 passages, on a : `S` = ... et `u` = ...
- Après 3 passages, on a : `S` = ... et `u` = ...

$\vdots$

- Après n passages, on a : `S` = ... et `u` = ...

Le programme calcule donc la somme : ....

Exercice 3

On veut écrire un programme permettant de calculer efficacement $S_n = \sum_{k=0}^{n} \frac{2^k}{k!}$.

1. Pour tout $k \in \mathbb{N}$, établir une relation simple entre $\frac{2^{k+1}}{(k+1)!}$ et $\frac{2^k}{k!}$:

$$\frac{2^{k+1}}{(k+1)!} = \frac{2^k}{k!} \times \ldots\ldots$$

2. Compléter la fonction suivante pour qu'elle renvoie la valeur de S_n.
(On n'utilisera ni "`**`", ni "`factorial`" !)

```
def somme(n) :

        S = ...... ;  u = ......

        for k in range(n+1) :

                S = ............................

                u = ............................

        return(S)
```

Pour s'entrainer

Exercice 4

Suite de Wallis. On considère la suite $(W_n)_{n\in\mathbb{N}}$ définie par :

$$W_0 = \frac{\pi}{2} \quad \text{et} \quad \forall n \in \mathbb{N},\ (n+1)W_{n+1}W_n = \frac{\pi}{2}.$$

Créer un programme qui affiche la valeur de W_{50}.

Exercice 5

Récurrence couplée. On définit les suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ de la façon suivante :

$$\begin{cases} u_0 = 1 \\ v_0 = -2 \end{cases} \qquad \begin{cases} \forall n \in \mathbb{N},\ u_{n+1} = 2u_n - v_n \\ \forall n \in \mathbb{N},\ v_{n+1} = u_n + 4v_n \end{cases}$$

Définir une fonction qui prend en entrée un entier `n` et renvoie le couple (u_n, v_n).