TP n°3 : Suites et Boucles

1 Suites récurrentes sur Python

1.1 La boucle For

Exercice n°1

1. On considère le script suivant (**à ne pas le reproduire dans la fenêtre de script**) :

```
u=0
u=2*u+1
u=2*u+1
u=2*u+1
u=2*u+1
u=2*u+1
u=2*u+1
u=2*u+1
u=2*u+1
print(u)
```

   Préciser la valeur de la variable `u` à chaque ligne du script. Quelles est la valeur affichée ?

2. Le script précédent permet de calculer les termes d'une suite récurrente. Laquelle ? Quel terme de cette suite est affiché ?

3. Tester le script suivant et vérifier qu'il permet d'obtenir le résultat du script précédent :

```
u=0
for k in [1,2,3,4,5,6,7,8]:
      u=2*u+1
print(u)
```

Dans un programme, il est souvent nécessaire de pouvoir faire répéter à Python une même instruction plusieurs fois de suite. Lorsque l'on connaît exactement le nombre de répétitions nécessaires, la "boucle for" permet de faire cela. La variable `k` est appelée "variable incrémentale".

4. Pouvez-vous deviner ce que va afficher le script suivant ?

```
u=0
for k in [-10,23,30,4,15,6,71,82]:
      u=2*u+1
print(u)
```

   Si vous n'y arrivez pas, l'exécuter sur Python. Que remarquez-vous ?

5. On considère les suites suivantes :

$$\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = -2u_n + 5 \end{cases} \quad \begin{cases} v_2 = 2 \\ \forall n \in \mathbb{N}^{\geq 2},\ v_{n+1} = -2v_n + 5 \end{cases} \quad \begin{cases} w_0 = 1 \\ \forall n \in \mathbb{N},\ w_{n+1} = nw_n - 2^n \end{cases}$$

$$\begin{cases} t_2 = 1 \\ \forall n \in \mathbb{N}^{\geq 2},\ t_{n+1} = nt_n - 2^n \end{cases}$$

Proposer trois scripts Python permettant de calculer u_4, puis v_5, puis w_4, puis t_5.
Les valeurs à trouver sont : $u_4 = 7$, $v_5 = -1$, $w_4 = -38$ *et* $t_5 = -72$.

1.2 La commande Range

Commande range

- `range(n,p)` est l'écriture en Python de la liste des entiers de n à $p-1$.
- `range(n)` est l'écriture en Python de la liste des entiers de 0 à $n-1$.
- `range(n,p,k)` est l'écriture en Python de la liste des entiers de n à $p-1$ avec un pas de k.
- Par exemple :
  - on peut traduire en français `range(1,6)` par la liste d'entiers `[1,2,3,4,5]`
  - on peut traduire en français `range(6)` par la liste d'entiers `[0,1,2,3,4,5]`
  - on peut traduire en français `range(1,10,2)` par la liste d'entiers `[1,3,5,7,9]`

Exercice n°2

1. Deviner l'affichage avant d'exécuter les scripts suivants et expliquer la différence entre les deux scripts.

Script 1

```
for i in range(1,6):
    print(i)
print("fini !")
```

Script 2

```
for i in range(1,6):
    print(i)
    print("fini !")
```

2. On considère le Script 1 :
   (a) Que devient l'affichage si l'entête est : `for i in range(6)` ?
   (b) Que devient l'affichage si l'entête est : `for i in range(3,11,2)` ?
   (c) Que devient l'affichage si `print(i)` est remplacée par `print(i**2)` ?
3. Reprendre les scripts de l'exercice n°1 question 5. et modifier ces scripts à l'aide de la commande `range`.

1.3 Fonction permettant le calcul de u_n où (u_n) suite récurrente d'ordre 1

On peut créer une fonction qui prend en arguments un entier n et un réel u_0 et qui renvoie la valeur du terme de rang n d'une suite (u_n) récurrente d'ordre 1.

Par exemple, la fonction Python `suite(n)` suivante :

```
def suite(u0,n):
    u=u0
    for k in range(n):
        u=2*u+1
    return u
```

permet de renvoyer le terme de rang n de la suite $\begin{cases} u_0 = 0 \\ \forall n \in \mathbb{N},\ u_{n+1} = 2u_n + 1 \end{cases}$

Exercice n°3

Pour chacune des suites suivantes :

$$\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = u_n^2 + 5 \end{cases} \qquad \begin{cases} v_0 = 1 \\ \forall n \in \mathbb{N},\ v_{n+1} = \sqrt{v_n + n} \end{cases} \qquad \begin{cases} w_3 = -2 \\ \forall n \in \mathbb{N}^{\geq 3},\ w_{n+1} = w_n + \dfrac{1}{n-1} \end{cases}$$

construire une fonction Python (qu'on nommera `suite_u`, `suite_v` et `suite_w`) qui prend en argument un entier n et qui renvoie la valeur de du terme de rang n.

On pourra vérifier que les fonctions sont correctes à l'aide des données suivantes : $u_4 = 54774806$, $v_4 = 3.51282\ldots$, $w_{1000} = 4.48346\ldots$.

Exercice n°4

Construire une fonction Python permettant de calculer le terme de rang n de la suite $\begin{cases} u_0 = 0,\ u_1 = 1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$

1.4 Fonction permettant le calcul de u_n où (u_n) suite récurrente d'ordre 2

On peut créer une fonction qui prend en argument un entier n et deux réels u_0, u_1 et qui renvoie la valeur du terme de rang n d'une suite (u_n) récurrente d'ordre 2.

Par exemple, la fonction Python `suite(n)` suivante :

```
def suite(u0,u1,n):
    u=u0
    v=u1
    for k in range(n):
        z=v
        v=u+v
        u=z
    return u
```

Avec variable auxiliaire

```
def suite(u0,u1,n):
    (u,v)=(u0,u1)
    for k in range(n):
        (u,v)=(v,u+v)
    return u
```

Avec affectation simultanée

permet de renvoyer le terme de rang n de la suite $\begin{cases} u_0 = 0,\ u_1 = 1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$

Exercice n°5

Pour chacune des suites suivantes :

$$\begin{cases} u_0 = 2, u_1 = -1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_n^2 u_{n+1}^3 \end{cases} \qquad \begin{cases} v_0 = 1, v_1 = 1 \\ \forall n \in \mathbb{N},\ v_{n+2} = 4v_n - \dfrac{1}{n+1} \end{cases}$$

construire une fonction Python (qu'on nommera `suite_u` et `suite_v`) qui prend en argument un entier n et qui renvoie la valeur de du terme de rang n.

On pourra vérifier que les fonctions sont correctes à l'aide des données suivantes : $u_4 = -4194304$, $v_{10} = 742.78412\ldots$.

1.5 Somme des termes d'une suites

Exercice n°6

On considère les scripts suivantes :

```
S=0
for k in range(1,5):
    S = S + 2*k+1
print(S)
```

Script n°1

```
u=2
S=u
for k in range(3):
    u = u**2 + 1
    S = S + u
print(S)
```

Script n°2

1. On exécute le **Script n°1**. Remplir le tableau d'état suivant et déterminer la valeur renvoyée :

k	2*k+1	S

2. Si n est un entier naturel, à quoi sert le **Script n°1** ?
3. Répondre aux mêmes questions pour le **Script n°2**, on adaptera le tableau d'état.

1.6 Si la suite est définie explicitement

Soit $n \in \mathbb{N}^{\geq 2}$. Pour calculer

$$\sum_{k=2}^{n}(2k+1) = \overbrace{(2\times 2+1)}^{\text{ajouté à la boucle n°1}} + \overbrace{(2\times 3+1)}^{\text{ajouté à la boucle n°2}} + \overbrace{(2\times 4+1)}^{\text{ajouté à la boucle n°3}} + \ldots\ldots + \overbrace{(2\times n+1)}^{\text{ajouté à la boucle n°}(n-1)}$$

on peut utiliser le script suivant :

```
S=0 #On initialise une variable S qui contiendra la somme
for k in range(2,n+1):
      S = S + 2*k+1 #On peut aussi écrire S += 2*k+1
print(S)
```

1.6.1 Si la suite est définie par récurrence

Soit $n \in \mathbb{N}$. Pour calculer $\displaystyle\sum_{k=0}^{n} u_k$ où (u_n) est définie par $\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = u_n^2 + 1 \end{cases}$ on peut utiliser le script suivant

```
u = 2 #On initialise une variable u qui contiendra les valeurs de la suite
S = u #On initialise une variable S qui contiendra la somme
for k in range(n):
      u = u**2+1
      S += u
print(S)
```

Exercice n°7

Les questions qui suivent sont indépendantes.

1. Soit $(u_n)_{n\in\mathbb{N}}$ une suite arithmétique de raison 5 telle que $u_0 = -2$.
   (a) Déterminer la forme explicite de $(u_n)_{n\in\mathbb{N}}$ puis donner la valeur explicite, en fonction de n, de la somme des n **premiers termes** de $(u_n)_{n\in\mathbb{N}}$.
   (b) Compléter le script suivant pour que, pour tout entier naturel n donné par l'utilisateur, il affiche la valeur de la somme des n premiers termes de $(u_n)_{n\in\mathbb{N}}$.

```
n = int(input('Donner un entier naturel : '))
S = .....
for k in .....
      S = .....
print(S)
```

   (c) Tester le script et vérifier qu'il affiche bien la valeur demandée.
2. Construire une **fonction Python** `Somme(n)` prenant en argument un entier naturel n et renvoyant la valeur de la somme $\displaystyle\sum_{k=0}^{n}(3-2k)$.
3. Soit $(u_n)_{n\in\mathbb{N}}$ une suite telle que $u_0 = 1$ et, pour tout $n \in \mathbb{N}$, $u_{n+1} = 3u_n - 1$.
   (a) Compléter la fonction Python `Somme(n)` prenant en argument un entier naturel n et renvoyant la valeur de la somme $\displaystyle\sum_{k=0}^{n} u_k$.

```
def Somme(n):
      u = .....
      S = .....
      for k in .....
            u = .....
            S = .....
      return S
```

(b) On considère la suite $(u_n)_{n\geq 1}$ définie par $u_1 \in \mathbb{R}$ et, pour tout $n \in \mathbb{N}^{\geq 1}$, $u_{n+1} = \sqrt{1+nu_n^2}$. Construire une **fonction Python** `Somme(u1,n)` prenant en arguments la valeur de u_1, un entier naturel $n \in \mathbb{N}^{\geq 1}$ et renvoyant la valeur de la somme $\sum_{k=1}^{n} u_k$.

1.7 Algorithme de seuil et boucle While

Exercice n°8

On considère la fonction Python suivante :

```
def algo():
    u = 1
    n = 0
    while u<50:
        u = u*2
        n = n+1
    return n
```

1. Remplir le tableau d'état suivant et déterminer la valeur renvoyée par l'exécution de `algo()` :

u<50	u	n
	1	0
True		

2. Quel est le but de cet fonction ? Expliquer.

Exercice n°9

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 3$ et, pour tout $n \in \mathbb{N}$, par $u_{n+1} = u_n + 2$.

1. Donner son expression explicite puis donner $\lim\limits_{n\to+\infty} u_n$.
2. Déterminer par un calcul la plus petit valeur de n telle que $u_n \geq 10^6$.
3. Compléter la fonction suivante pour qu'elle retourne le plus petit entier n telle que $u_n \geq 10^6$.

```
def seuil():
    u = .....
    n = .....
    while .....
        u = .....
        n = .....
    return .....
```

Exercice n°10

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 5$ et, pour tout $n \in \mathbb{N}$, par $u_{n+1} = \dfrac{1}{2}u_n$.

1. Donner son expression explicite puis donner $\lim\limits_{n\to+\infty} u_n$.
2. Déterminer par un calcul la plus petit valeur de n telle que $u_n \leq 0,0001$.
3. Créer un script qui affiche le plus petit entier n telle que $u_n \leq 0,0001$.

2 Exercices de synthèse

Exercice n°11

Soit la suite définie par $u_0 = 1$ et : $\forall n \in \mathbb{N}$, $u_{n+1} = \sqrt{u_n + n}$.

1. Ecrire une fonction Python `suite(n)` qui prend en argument l'entier naturel n et qui renvoie la valeur de u_n. On doit trouver $u_{100} \approx 10,45990\ldots$
2. Ecrire une fonction Python `Somme(n)` qui renvoie $\sum_{k=0}^{n} u_k$. On doit trouver $S_{25} \approx 94,7077\ldots$
3. On admet que $\lim\limits_{n\to+\infty} u_n = +\infty$.
   (a) Ecrire une fonction `rangSup(M)` qui prend en argument un nombre M et renvoie le premier indice tel que $u_n \geq M$.
   (b) Ecrire une fonction `rangInf(M)` qui prend en argument un nombre M et renvoie le dernier indice tel que $u_n < M$.
   (c) Ecrire une fonction `termeSup(M)` qui prend en argument un nombre M et renvoie le premier terme u_n de la suite tel que $u_n \geq M$.
   (d) Ecrire une fonction `termeInf(M)` qui prend en argument un nombre M et renvoie le dernier terme u_n de la suite tel que $u_n < M$.

Exercice n°12

Une suite de Syracuse est définie par :

$$u_0 \in \mathbb{N}^{\geq 1}, \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

Dans la suite, on considère que la conjecture de Syracuse est juste : pour toute condition initiale $u_0 \in \mathbb{N}^{\geq 1}$, la suite atteint la valeur 1.

1. Programmer une fonction `syracuse(u0,n)` qui prend en argument un entier $n0 \geq 1$, et un entier $n \in \mathbb{N}$ et qui calcule le terme u_n de la suite de Syracuse.
2. Programmer une fonction `tempsVol(u0)` qui prend en argument un entier $u0 \geq 1$, et qui renvoie le premier entier n tel que $u_n = 1$.
3. Programmer une fonction `maximum(u0)` qui prend en argument un entier $u0 \geq 1$, et qui renvoie la maximum de la suite (u_n).
4. Programmer une fonction `tpsVolMax(n)` qui prend en argument un entier $n \geq 1$, et qui renvoie le plus petit entier $p \in [\![1, n]\!]$ pour lequel la suite de premier terme $u_0 = p$ a le plus grand temps de vol.