T.P. II - Lecture de code (II)

Code Capytale : `406b-786899`
Les structures de contrôle permettent d'automatiser certains processus. On présente ici : les boucles, les boucles conditionnelles et les conditionnelles.

I - Les boucles `for`

Pour effectuer une opération un nombre de fois **déterminé**, on utilise la notion de boucle. La syntaxe usuelle est `for i in range(a, b):` qui peut se traduire par *pour i variant dans l'intervalle d'entiers a, a + 1,..., a + b - 1*. On pourra remplacer `range(a, b)` par d'autres objets comme nous le verrons par la suite. **Attention**, la dernière valeur prise lors de l'itération sur `range(a, b)` est **b-1**.

```
for i in range(3, 7):
    print(i)
```

affiche

```
3
4
5
6
```

La variable qui parcourt l'itérateur peut être utilisée dans des expressions :

```
u = 12
for i in range(3, 7):
    u = u + i

print("u", u)
```

qui affiche

```
u 30
```

On peut représenter l'évolution des variables dans la boucle précédente en utilisant un tableau comme ci-dessous.

i	u
	12
3	15
4	19
5	24
6	30

Exercice 1. (Boucle itérative) Déterminer le contenu de la variable `u` à la fin de la suite d'instructions suivante.

```
n = 5
u = 10
for i in range(1, n):
    u = u + i

print("u", u)
```

Exercice 2. (Boucle itérative) On définit la suite (u_n) par $u_0 = 1$ et pour tout n entier naturel,

$$u_{n+1} = 3u_n + 4^{n+1}.$$

Compléter le script suivant pour que l'appel `u(n)` renvoie la valeur de u_n.

```
def u(n):
    u = ...
    for i in range(1, ...):
        u = 3 * ... + 4...
    return u
```

II - Boucles conditionnelles `while`

Pour effectuer une opération un nombre de fois **indéterminé**, on utilise la notion de boucle conditionnelle. La syntaxe usuelle est `while condition:` qui peut se traduire par *tant que* `condition` *est vraie*. Ces boucles sont typiquement utilisées pour déterminer le premier moment où une suite dépasse une certaine valeur.

```
u = 3
n = 0
while u < 40:
    u = 2 * u
    n = n + 1

print("n", n)
```

qui affiche

```
n 4
```

On peut représenter l'évolution des variables dans la boucle précédente en utilisant un tableau comme ci-dessous.

u < 40	u	n
	3	0
True	6	1
True	12	2
True	24	3
True	48	4
False		

La variable `n` contient donc la valeur 4.

Exercice 3. (Boucle conditionnelle) Déterminer le contenu des variables `a`, `b` et `i` à la fin de la suite d'instructions suivante.

```
a, b, i = 1, 1, 0
while i <= 10:
    i = 2 * i + 1
    a = a + b
    b = 2 * a + i

b = 3 * b

print("a", a)
print("b", b)
print("i", i)
```

III - Les conditionnelles `if`, `elif`, `else`

Le mot clé `if` (si) permet de tester si une conditon est vraie. Ensuite, `else` (sinon) traite les autres cas. L'insertion d'un `elif` entre les deux permet d'ajouter des cas. Le mot-clé `elif` est la contraction de `else` (sinon) et `if` (si) et pourrait se traduire par *sinon mais si.*
Par exemple, pour définir la fonction

$$f : x \mapsto \begin{cases} 0 & \text{si } x < 0 \\ \frac{1}{1+x} & \text{si } x \in [0,1] \\ \frac{1}{2+x} & \text{si } x > 1 \end{cases}$$

on écrira :

```
def f(x):
    if x < 0:
        return 0
    elif x <= 1:
        return 1/(1 + x)
    else:
        return 1/(2 + x)

print("f(-1)", f(-1))
print("f(0.5)", f(0.5))
print("f(124.456)", f(124.456))
```

qui affiche

```
f(-1) 0
f(0.5) 0.6666666666666666
f(124.456) 0.007907888909976592
```

Exercice 4. (Conditionnnelles)

1. Déterminer la valeur renvoyée par les appels `surnom1(8)`, `surnom1(50)` et `surnom1(111)`.

```
def surnom1(numero):
    s = "" # "" représente le texte vide
    if numero <= 10:
        s = "Riri"
    elif numero <= 100:
        s = "Fifi"
    else:
        s = "Loulou"
    return s

print("surnom1(8)", surnom1(8))
print("surnom1(50)", surnom1(50))
print("surnom1(111)", surnom1(111))
```

2. Déterminer la valeur renvoyée par les appels `surnom2(8)`, `surnom2(50)` et `surnom2(111)`.

```
def surnom2(numero):
    s = "" # "" représente le texte vide
    if numero <= 10:
        s = "Riri"
    if numero <= 100:
        s = "Fifi"
    else:
        s = "Loulou"
    return s

print("surnom2(8)", surnom2(8))
print("surnom2(50)", surnom2(50))
print("surnom2(111)", surnom2(111))
```