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

Solution de l'exercice 1. On représente l'évolution des variables dans un tableau :

n	i	u
5		10
	1	11
	2	13
	3	16
	4	20

Ainsi, le résultat affiché est

```
u 20
```

□

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.

Solution de l'exercice 3. On représente l'évolution des variables dans la boucle précédente en utilisant le tableau ci-dessous :

$i \leqslant 10$	i	a	b
	0	1	1
True	1	2	5
True	3	7	17
True	7	24	55
True	15	79	173
False			519

Ainsi, la suite d'instructions affiche

```
a 79
b 519
i 15
```

□

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
```

Solution de l'exercice 4.

1.

- Comme $8 \leqslant 10$, la condition après le `if` est satisfaite. On stocke la valeur `"Riri"` dans `s` puis le `elif` et le `else` ne sont pas effectués. La fonction renvoie `"Riri"`.
- Comme $50 > 10$, la condition après le `if` n'est pas satisfaite. Comme $50 \leqslant 100$, la condition après le `elif` est satisfaite. On stocke la valeur `"Fifi"` dans `s` puis le `else` n'est pas effectué. La fonction renvoie `"Fifi"`.

* Comme $111 > 10$, la condition après le `if` n'est pas satisfaite. Comme $111 > 100$, la condition après le `elif` n'est pas satisfaite. Ainsi, le `else` est effectué. On stocke la valeur `"Loulou"` dans `s`. La fonction renvoie `"Loulou"`.

2.

* Comme $8 \leqslant 10$, la condition après le `if` est satisfaite. On stocke la valeur `"Riri"` dans `s`.
  On rentre ensuite dans la seconde conditionnelle. Comme $8 \leqslant 100$, la condition après le `if` est satisfaite. On stocke la valeur `"Fifi"` dans `s` puis le `else` n'est pas effectués. La fonction renvoie `"Fifi"`.
* Comme $50 > 10$, la condition après le `if` n'est pas satisfaite.
  On rentre ensuite dans la seconde conditionnelle. Comme $50 \leqslant 100$, la condition après le `if` est satisfaite. On stocke la valeur `"Fifi"` dans `s` puis le `else` n'est pas effectués. La fonction renvoie `"Fifi"`.
* Comme $111 > 10$, la condition après le `if` n'est pas satisfaite.
  On rentre ensuite dans la seconde conditionnelle. Comme $111 > 100$, la condition après le `if` n'est pas satisfaite. Ainsi, le `else` est effecté. On stocke la valeur `"Loulou"` dans `s`. La fonction renvoie `"Loulou"`.

□