Algorithme 2 : Boucle "if"

Mohamed NASSIRI

Objectifs :

- Comprendre la structure conditionnelle (if...else).
- Savoir utiliser la structure conditionnelle (notée "Si...alors" en langage naturel)

Mots – clefs :

Comparaisons - Structure conditionnelle - if - else - elif

Prérequis :

Langage Scratch

La musique du chapitre : Woodkid - I Love You. Album : The Golden Age. Date de sortie : 2013.

« J'ai hâte de voir le jour où je dirais à Dieu la solitude. »

Roy McBride, Ad Astra, 2019.

Sources :

Python Doctor FRANCE
Débuter avec Python au lycée
OpenClassrooms : Apprenez à programmer en Python - Créez des structures conditionnelles

1 Les comparaisons possibles et les les structures conditionnelles

1.1 Les comparaisons possibles

En programmant, vous aurez souvent à comparer des nombres. Voici une liste des opérateurs que vous pourrez utiliser :

```
x == y   #est vrai quand x est égal à y,
x != y   #est vrai quand x est différent de y,
x > y    #est vrai quand x est strictement supérieur à y,
x < y    #est vrai quand x est strictement inférieur à y,
x >= y   #est vrai quand x est supérieur ou égal à y, et
x <= y   #est vrai quand x est inférieur ou égal à y.
```

1.2 Les structures conditionnelles

Les mots clé **if**, **elif** et **else** (que l'on va voir juste dans quelques instants) cherchent à savoir si ce **True** (en anglais "True" signifie "Vrai"). Donc si c'est la valeur est **True**, les instructions concernant la condition seront exécutée.

Comment savoir si la valeur qu'on soumet à l'interpreteur est True ? Comme on l'a déjà vu, il est possible de le voir directement dans l'interpréteur.

Demandons à python si 7 est égal à 9 :

```
>>> 7 == 9
False
```

Ouf ! Il nous répond gentiment que c'est **False** (nous voilà rassurés !), c'est-à-dire que c'est faux.

On peut aussi donner une valeur à une variable est on va lui demander si la valeur correspond bien à ce que l'on attend.

```
>>> hein = 6
>>> hein == 6
True
```

1.3 Les conditions *and* et *or*

Il est possible d'affiner une condition avec les mots clé **and** qui signifie "et" et **or** qui signifie "ou".

On veut par exemple savoir si une valeur v est plus grande que 3 mais aussi plus petite que 12, c'est-à-dire $3 \leq v \leq 12$:

```
>>> v = 20
>>> v > 3 and v < 12
False
```

Essayons avec la valeur 10 :

```
>>> v = 10
>>> v > 3 and v < 12
True
```

Pour que le résultat soit **True**, il faut que **les deux conditions soient remplies**.

Regardons maintenant comment fonctionne la condition **or**. On veut par exemple savoir si une valeur v est plus grande que 7 ou plus petite que 23 :

```
>>> v = 10
>>> v > 7 or v < 23
True
```

Le résultat est **True** parce qu'au moins **une des deux conditions est respectée**.

```
>>> v = 3
>>> v > 7 or v < 23
False
```

Dans ce cas là, **aucune condition n'est respectéee**, le résultat est donc **False**.

Pour devenir un crack !
On peut combiner deux booléens entre eux avec les opérateurs and et or. Ci-dessous nous demandons à Python d'écrire les tables de vérité de ces deux opérateurs.

```
>>> for a in [False, True]:
...     for b in [False, True]:
...         print(a, "and", b, "vaut", a and b)
...
False and False vaut False
False and True vaut False
True and False vaut False
True and True vaut True
```

```
>>> for a in [False, True]:
...     for b in [False, True]:
...         print(a, "or", b, "vaut", a or b)
...
False or False vaut False
False or True vaut True
True or False vaut True
True or True vaut True
```

2 Condition if

Cette notion est l'une des plus importante en programmation. L'idée est de dire que si *telle variable a telle valeur alors faire cela sinon cela.* Voici un tableau qui donne la correspondance avec le langage naturel et Scratch

Scratch	
Algorithme	**Si** condition **Alors** instructions **Fin Si**
Python	**if** condition : instructions

Prenons un exemple, on va donner une valeur à une variable, notée "var", et si cette valeur est supérieur à 7, alors on va incrémenter la valeur de 1

```
>>> var = 12
>>> if var > 7:
...     var = var + 1
```

```
...
>>> var
13
```

Mais que ce serait-il passé si la valeur était inférieure à 7 ?

```
>>> var = 5
>>> if var > 7:
...     var = var + 1
...
>>> var
5
```

On remarque que si la condition n'est pas remplie, les instructions dans la structure conditionnelle sont ignorées.

3 Condition if else

Il est possible de donner des instructions quelque soit les choix possibles avec le mot clé **else**. Voici un tableau qui donne la correspondance avec le langage naturel et Scratch

Scratch	
Algorithme	**Si** condition **Alors** instructions 1 **Sinon** instructions 2 **Fin Si**
Python	**if** condition : instructions **else :** instructions 2

Voyons un premier exemple :

```
>>> age = 21
>>> if age >= 18: # Si age est supérieur ou égal à 18
...     print("Vous êtes majeur.")
... else: # Sinon (age inférieur à 18)
...     print("Vous êtes mineur.")
```

Grâce aux conditions **and** et **or**, on va pouvoir tester à la fois si, par exemple, une valeur est supérieur ou égal à 2 et inférieur ou égal à 8. On peut donc réduire ainsi les conditions imbriquées :

```
if a>=2 and a<=8:
    print("a est dans l'intervalle.")
else:
    print("a n'est pas dans l'intervalle.")
```

Sur le script qui suit, nous allons chercher à savoir si a n'est pas dans l'intervalle. La variable ne se trouve pas dans l'intervalle si elle est inférieure à 2 ou supérieure à 8. Voici donc le code :

```
if a<2 or a>8:
    print("a n'est pas dans l'intervalle.")
else:
    print("a est dans l'intervalle.")
```

Un dernier exemple qui va servir de transition.

```
>>> a = 42
>>> if a > 0:
...     print("a est supérieur à 0.")
... else:
...     print("a est inférieur ou égal à 0.")
```

Cependant, j'aimerais la différence entre les nombres positifs, négatifs et nuls (j'aimerais donc avoir trois conditions), il va falloir utiliser une condition intermédiaire.

4 Condition elif

Il est possible d'ajouter autant de conditions précises que l'on souhaite en ajoutant le mot clé elif , contraction de "else" et "if", qu'on pourrait traduire par "sinon".

```
>>> a = 5
>>> if a > 5:
...     a = a + 1
... elif a == 5:
...     a = a + 1000
... else:
...     a = a - 1
...
>>> a
1005
```

Dans cet exemple, on a repris le même que les précédent mais nous avons ajouté la conditions "Si la valeur est égale à 5" que se passe-t-il ? Et bien on ajoute 1000.