INTRODUCTOIN A PYTHON

Drs Kyelem Yacouba

Université Joseph Ki-Zerbo

Laboratoire de Mathématiques et d'Informatique (LAMI)

Contact : kyelemy@yahoo.fr / 71025937

Plan du cours

Chapitre 2 : Les structures de contrôle en python

I. Les structures conditionnelles

1. *La condition if en Python*

Syntaxe :

if condition : code à exécuter.

Pensez bien à indiquer le : et à bien indenter le code qui doit être exécuté si la condition est vérifiée sinon votre condition ne fonctionnera pas.

La structure conditionnelle if est une structure de base qu'on retrouve dans de nombreux langages de programmation. Cette condition va nous permettre d'exécuter un code si et seulement si une certaine condition est vérifiée.

On va en fait passer une expression à cette condition qui va être évaluée par Python. Cette expression sera souvent une comparaison explicite (une comparaison utilisant les opérateurs de comparaison) mais pas nécessairement. Si Python évalue l'expression passée à vraie, le code dans la condition if sera exécuté. Dans le cas contraire, le code dans if sera ignoré.

Exemple

```
X=5
If x==5 :
        Print('x contient le nombre 5')
```

2. *La condition if… else en Python*

La structure conditionnelle if…else (« si… sinon » en français) est plus complète que la condition if puisqu'elle nous permet d'exécuter un premier bloc de code si un test renvoie vraie ou un autre bloc de code dans le cas contraire.

Syntaxe :

```
if condition :
        code à exécuter.
else :
        code à exécuter
```

Exemple

```
X=5
If x==5 :
        Print('x contient le nombre 5')
else :
        print('x contient une autre valeur')
```

3. *La condition if… elif… else en Python*

Syntaxe :

```
if condition :
        code à exécuter.
Elif condition :
        code à exécuter
elif condition :
etc.
Else :
        code à exécuter
```

La condition if…elif…else (« si…sinon si…sinon ») est une structure conditionnelle encore plus complète que la condition if…else qui va nous permettre cette fois-ci d'effectuer autant de tests que l'on souhaite et ainsi de prendre en compte le nombre de cas souhaité.

En effet, nous allons pouvoir ajouter autant de elif que l'on souhaite entre le if de départ et le else de fin et chaque elif va pouvoir posséder son propre test ce qui va nous permettre d'apporter des réponses très précises à différentes situations.

Exemple

```
>>> x = 5
>>> if x < 0:
...     print("x contient une valeur négative")
... elif x < 10:
...     print("x contient une valeur comprise entre 0 et 10 exclu")
... elif x < 100:
...     print("x contient une valeur comprise entre 10 et 100 exclu")
... else:
...     print("x contient une valeur supérieure à 100")
...
x contient une valeur comprise entre 0 et 10 exclu
```

Remarque *: Vous pouvez imbriquer une structure conditionnelle dans une autre structure, puis dans une autre, et encore une autre, etc...*

II. Les structures itératives

1. *Boucle for*

Quand on sait combien de fois doit avoir lieu la répétition, on utilise généralement une boucle for.

L'instruction for est une instruction composée, c'est-à-dire une instruction dont l'en-tête se termine par deux points : suivie d'un bloc indenté qui constitue le corps de la boucle.

On dit que l'on réalise une itération de la boucle à chaque fois que le corps de la boucle est exécuté.
Dans l'en-tête de la boucle, on précise après le mot-clé for le nom d'une variable (i dans l'exemple ci-dessus) qui prendra successivement toutes les valeurs qui sont données après le mot-clé in. On dit souvent que cette variable (ici i) est un compteur car elle sert à numéroter les itérations de la boucle.

```
for i in [0, 1, 2, 3]:
    print("i a pour valeur", i)
```

Il est possible d'obtenir le même résultat sans donner la liste des valeurs, mais en utilisant la fonction range().

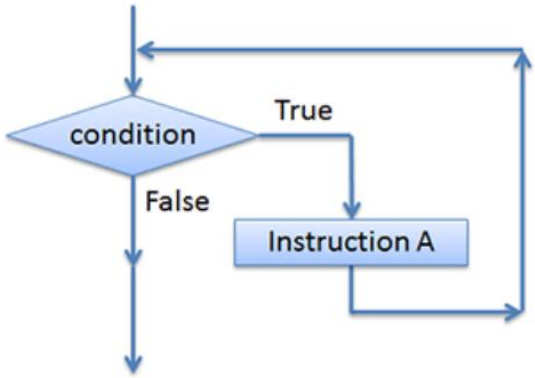
```
for i in range(4):
    print("i a pour valeur", i)
```

Pour parcourir les indices d'une liste, il est possible de combiner range() et len() comme ci-dessous :

```
c = ["Marc", "est", "dans", "le", "jardin"]
for i in range(len(c)):
    print("i vaut", i, "et c[i] vaut", c[i])
```

2. Boucle while

Si on ne connait pas à l'avance le nombre de répétitions, on choisit une boucle while.



Syntaxe :

```
while condition:
    Instruction A
```

Le mot-clé while signifie tant que en anglais. Le corps de la boucle (c'est-à-dire le bloc d'instructions indentées) sera répété tant que la condition est vraie.

Remarque : Si la condition est fausse au départ, le corps de la boucle n'est jamais exécuté. Si la condition reste toujours vraie, alors le corps de la boucle est répété indéfiniment.

Exemple :

```
x = 1
while x < 10:
    print("x a pour valeur", x)
    x = x * 2
print("Fin")
```

III. Les structures spécifiques

1. *L'instruction break*

L'instruction break permet de « casser » l'exécution d'une boucle (while ou for). Elle fait sortir de la boucle et passer à l'instruction suivante.

Exemple :

```
for i in range(10):
    print("debut iteration", i)
    print("bonjour")
    if i == 2:
    break
    print("fin iteration", i)
print("apres la boucle")
```

Affichage après execution:

```
debut iteration 0
bonjour
fin iteration 0
debut iteration 1
bonjour
fin iteration 1
debut iteration 2
bonjour
apres la boucle
```

Remarque : Dans le cas de boucles imbriquées, l'instruction break ne fait sortir que de la boucle la plus interne.

2. *L'instruction continue*

L'instruction continue permet de passer prématurément au tour de boucle suivant. Elle fait continuer sur la prochaine itération de la boucle.

Exemple

```
for i in range(4):
    print("debut iteration", i)
    print("bonjour")
    if i < 2:
        continue
    print("fin iteration", i)
print("apres la boucle")
```

Resultat

```
debut iteration 0
bonjour
debut iteration 1
bonjour
debut iteration 2
bonjour
fin iteration 2
debut iteration 3
bonjour
fin iteration 3
apres la boucle
```

Références

https://www.pierre-giraud.com/python-apprendre-programmer-cours/

https://courspython.com/boucles.html

https://www.docstring.fr/glossaire/structure-conditionnelle/

Cours Matthieu Boileau & Vincent Legoll Apprendre Python pour les sciences CNRS - Université de Strasbourg.

Cours de Python Introduction à la programmation Python pour la biologie Patrick Fuchs et Pierre Poulain https://python.sdv.univ-paris-diderot.fr/ .