Programmation en Python (III)

Akka Zemmari
Hervé Hocquard
herve.hocquard@u-bordeaux.fr

LaBRI, Université de Bordeaux - CNRS

17 septembre, 2023

université de BORDEAUX

Plan

Ce que nous verrons dans ce cours :

- Opérateurs de base
- Types de variables
- Nombres
- Chaines de caractères
- Listes
- Tuples
- Dictionnaire
- Branchements conditionnels
- Boucles
- Fonctions
- Modules
- Fichiers I/O
- Exceptions
- Classes et Objets

université de BORDEAUX

Branchement conditionnel

Les branchements permettent d'exécuter des instructions différentes selon la valeur d'une condition logique.

```
1  if condition:
2      bloc d'instructions
3  else:
4      bloc d'instructions
```

- Attention au :
- C'est l'indentation (le décalage par rapport à la marge gauche) qui délimite le bloc d'instructions
- La partie else est facultative

université de BORDEAUX

Branchement conditionnel : exemple

```
1  pht = float(input("Prix HT :"))
2  code = int(input("Code du produit : "))
3  if (code == 1):
4      taxe = pht * 0.055
5      pttc = pht + taxe
6  else:
7      pttc = pht * 1.2
8  print("Prix TTC : " + str(pttc))
9  input("pause ...")
```

université de BORDEAUX

if imbriqués

```
1  pht = float(input("Prix HT :"))
2  code = int(input("Code du produit : "))
3  if (code == 1):
4      pttc = pht * 1.055
5  elif (code == 2):
6      pttc = pht * 1.1
7  else:
8      pttc = pht * 1.2
9  print("Prix TTC : " + str(pttc))
10 input("pause ...")
```

- `elif` n'est déclenché que si la (les) condition(s) précédente(s) a (ont) échoué.
- `elif` est situé au même niveau que if et else
- On peut en mettre autant que l'on veut

université de BORDEAUX

Instructions itératives

Les instructions itératives permettent de répéter un certain nombre de fois l'exécution d'une suite d'instructions sous une certaine condition.

Exemple :

Afficher les dix premiers entiers strictement positifs

```
1  print 1
2  print 2
3  print 3
4  print 4
5  print 5
6  print 6
7  print 7
8  print 8
9  print 9
10 print 10
```

Boucle for

Principe :
Elle ne s'applique que sur une collection de valeurs. Ex. tuples, listes,...
On peut définir des boucles indicées en générant une collection de valeurs avec `range()`.

```
1  for indice in sequence:
2      bloc d'instructions
```

`sequence` est une collection de valeurs. Elle peut être générée avec range()

université de BORDEAUX

Boucle for : exemple

Un programme qui calcule la somme des nombres entre 1 et *n* et la somme des nombres pairs dans le même intervalle :

```
1  n = int(input("n : "))
2  somme = 0
3  sommePaires = 0
4  for i in range(1,n+1):
5      #Il faut mettre £n+1£ dans \texttt{range()} pour que n soit inclus dans la somme.
6      if (i%2 == 0):
7          sommePaires = sommesPaires + i
8      somme = somme + i
9  print("Somme de tous les nombres :" + str(somme))
10 print("Somme des nombres pairs :" + str(sommePaires))
11 input("pause ....")
```

Boucle while

Syntaxe :

```
while condition :
    bloc d'instructions
```

Tant que la condition est vérifiée, le bloc d'instructions sera exécuté.

Exemple :

```
n = int(input("Entrez le nombre d'entiers : "))
compteur = 1
while compteur <= n :
    print(compteur)
    compteur = compteur + 1
```

Boucle while

Si la valeur de la condition est fausse dès le départ alors le bloc d'instructions ne sera jamais exécuté.

```
1  compteur = 1
2  while compteur < 0 :
3      print(compteur)
4      compteur = compteur + 1
```

université de BORDEAUX

Boucle while

Boucle infinie !!

Par contre si la valeur de la condition est vraie et que le bloc d'instructions ne permet pas d'altérer cette valeur alors le bloc d'instructions sera exécuté à l'infini : on a alors affaire à une boucle infinie.

```
1  compteur = 1
2  while compteur != compteur + 1 :
3      print(compteur)
4      compteur = compteur + 1
```

université de BORDEAUX