Python - fiche récapitulative

Python est un language de programmation : il permet d'écrire des instructions qui seront ensuite exécutées par l'ordinateur.
Cette fiche ne décrit que les bases de Python : on ne parle pas de `class`, de dictionnaires, ou encore de libraries.

Affichage

Pour afficher du texte, des nombres ou autre, on utilise la fonction `print` :

```
print(5) # affiche «5»
print(3.8, 6)  # affiche «3.8 6»
print("bonjour !") # affiche «bonjour !»
```

Variables

En python, le symbol = n'a pas le même sens qu'en mathématiques : il sert à *assigner une variable*.

```
x = 3
print(x) # affiche «3»
```

Ici la valeur de `x` peut changer au cours du programme :

```
x = 3
print(x) # affiche «3»
x = 5
print(x) # affiche «5»
x = x + 6 # la nouvelle valeur de x est : l'ancienne valeur de x, additionnée à 6
print(x) # affiche «11»
```

Types

Chaque valeur en python a un **type** :

- **entier** (`int`) : ce sont les nombres entiers, comme `0`, `15` ou `-6`.
- **flottant** (`float`) : ce sont les nombres à virgule, comme `2.3`, `-0.75` ou `500.2`.
  À noter : en python, on utilise un point plutôt qu'une virgule.
- **booléen** (`bool`) : ce sont les valeur `True` (Vrai) et `False` (Faux).
- **chaîne de caractères** (`string`) : ce sont les morceaux de texte : par exemple, `"mon texte"` est une chaîne de caractères.

Calculs

On peut utiliser les quatres opérations de bases sur les nombres (entiers et flottants) :

- addition avec `+`
- soustraction avec `-`
- multiplication avec `*`
- division avec `/`

On peut également :

- Concaténer deux chaînes de caractères avec `+` : `"bon" + "jour !"`
- Tester si deux valeurs sont égales : `3 == 3`
- Tester si deux valeurs sont différentes : `5 != "bonjour"`
- Tester si une valeur est plus grande ou plus petite qu'une autre : `-1 < 0`, `-1 <= 0`, `0 >= 0`, `1 > 0`
- Faire une division euclidienne : `//` donne le quotient, et `%` donne le reste. Par exemple, `(13 // 3) == 4` et `(13 % 3) == 1`.

Boucles

Il y a deux manière de faire une boucle en Python :

- La boucle `for` :

```
for i in range(6):
    print(i)
# affiche «0 1 2 3 4 5»
```

- La boucle `while` :

```
i = 0
while i < 6:
    print(i)
    i += 1
# affiche «0 1 2 3 4 5»
```

Conditions

On peut dire à Python d'éxécuter du code seulement si un condition est vraie avec les mots-clé `if`, `else` et `elif` :

```
x = 3

if x < 2:       # Si x est plus petit que 2
    print("a")
elif x > 5:     # Sinon, si x est plus grand que 5
    print("b")
else:           # Sinon
    print("c")
# affiche «c», mais pas «a» ni «b».
```

Fonctions

Pour pouvoir lancer le même code avec des valeurs différentes, Python permet de définir des fonctions avec le mot-clé `def` :

```
def ma_fonction(x, y):
    print(x, y)

ma_fonction(2, 3) # affiche «2 3»
ma_fonction(5, -1) # affiche «5 -1»
```

Listes

On peut regrouper les objets dans un autres type : les **listes** (`list`) :

```
ma_liste = [1, 2, 3] # une liste de 3 éléments
print(ma_liste[0]) # affiche le premier élément (on compte à partir de 0 en Python)
ma_liste[1] = 6
print(ma_liste) # affiche «[1, 6, 3]»
```