# Algorithmique et programmation

## Les variables et instructions élémentaires

### Définition

Une variable est un emplacement mémoire qui possède un nom et une valeur. La valeur peut être de différents types:

- un nombre entier ( `integer` ): -2; 3; 0; 726; -343; ...
- un nombre flottant ( `float` ): 2.5; -3.2; 2.0; ...
- un boolean ( `boolean` ): True; False.
- une chaîne de caractères ( `string` ): 'Bonjour'; 'Hello'; 'a'; ...

### Affectation

Pour affecter une valeur à une variable, on utilise le symbole = en python et le symbole ← en langage naturel (ou pseudo code).

En langage naturel:

```
x ← 3
est_gentil ← True
leTexte ← 'azerty'
y ← 1.4
```

En python, les mêmes instructions:

```python
x = 3
est_gentil = True
leTexte = 'azerty'
y = 1.4
```

Pour récupérer la valeur d'une variable, il suffit d'appeler la variable par son nom.

### Séquence d'instruction

Une séquence d'instruction est une suite d'instructions qui s'exécutent les unes après les autres dans l'ordre d'écriture.

En langage naturel:

> x ← 3
> y ← 5
> Afficher x + y

En python, les mêmes instructions:

```python
x = 3
y = 5
print(x+y)
```

Dans les deux cas, la séquence d'instructions affiche 8.

## Instructions conditionnelles

Une instruction conditionnelle effectue certaines instructions en fonction du résultat d'une instruction booléenne appelée condition.

### Instruction conditionnelle Si ... Alors ...

En langage naturel:

> Si age ≥ 18 alors
>  Afficher 'Tu es majeur.'
> Fin Si

En python, les mêmes instructions:

```python
if age >= 18:
    print('Tu es majeur')
```

### Instruction conditionnelle Si ... Alors ... Sinon ...

En langage naturel:

> Si age ≥ 18 alors
>  Afficher 'Tu es majeur.'
> Sinon

> Afficher 'Tu es mineur.'
> Fin Si

En python, les mêmes instructions:

```python
if age >= 18:
    print('Tu es majeur')
else:
    print('Tu es mineur')
```

**Les différents opérateurs de comparaison**

| Symbole | signification |
|---|---|
| == | est égal à |
| != | n'est pas égal à |
| < | est inférieur à |
| <= | est inférieur ou égal à |
| > | est supérieur à |
| >= | est supérieur ou égal à |

## Les boucles

Une boucle est une séquence d'instructions qui se répètent.

### Boucle bornée

Une boucle bornée est une séquence d'instructions qui se répètent un nombre de fois connu.

En langage naturel:

> Pour i allant de 1 à 10, faire:
> Afficher i
> Fin Pour

En python, les mêmes instructions:

```python
for i in range(1,11):
    print(i)
```

Dans les deux cas, La séquence d'instructions affiche:

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

### Petites précisions sur l'instruction `range`

Elle peut prendre en paramètres jusqu'à 3 valeurs entières `start`, `stop` et `step`. `start` et `stop` sont optionnels. Par défaut, `start` vaut 0 et `step` vaut 1.

`range` donenra tous les nombres entiers entre `start` (inclu) et `stop` (exclu) à intervalle régulier `step`, s'ils existent.

Exemples:

```python
>>> range(10)
>>> range(2,10)
>>> range(10,2)
>>> range(2,10,3)
>>> range(10,2,3)
>>> range(10,2,-3)
```

Si on demande d'imprimer tous les éléments de range dans ces 6 cas, on obtiendra, dans l'ordre:

```
1;2;3;4;5;6;7;8;9
2;3;4;5;6;7;8;9
rien
2;5;8
rien
10;7;4
```

**Boucle non bornée**

ne boucle non bornée est une séquence d'instructions qui se répètent un nombre de fois inconnu.

En langage naturel:

> i ← 1
> Tant que i ≤ 10, faire:
>  Afficher i
> Fin Tant que

En python, les mêmes instructions:

```python
i = 1
while i <= 10:
    print(i)
```

Dans les deux cas, La séquence d'instructions affiche:

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

# Les fonctions

## Définition

Une fonction est une séquence d'instructions qui possède un nom et qui retourne un résultat. Une fonction peut avoir des paramètres.

Pour définir un fonction en python, on utilise les instructions suivantes:

```python
def nom_de_la_fonction([paramètre1], [paramètre2], ...):
    instructions_de_la_fonction
    .
    .
    .
    return resultat
```

Exemples:

```python
def f(x):
    image = x**2
    return image
```

L'exemple précedent est la fonction dont le nom est `f` et qui retourne l'image de `x` (mis en paramètre) par la fonction carrée.

```python
def concatene(chaine1,chaine2):
    resultat = chaine1 + chaine2
    return resultat
```

L'exemple précedent est la fonction dont le nom est `concatene` et qui retourne une chaine de caractère qui est la concaténation de `chaine1` et `chaine2` (mis en paramètres).

Ainsi on peut utiliser les fonctions dans le shell de Thonny (après avoir appuyé sur `F5`):

```
>>> f(-3)
9
>>> concatene('abc', 'def')
'abcdef'
```

Mais aussi dans d'autres lignes d'instructions du programme:

```python
longueurChaine = len(concatene(ch1, ch2))
print(longueurChaine)
```

Enfin, comme une fonction retroune un résultat, on peut affecter ce résultat à une variable pour pouvoir réutiliser ce résultat=

```python
res = concatene('aze', 'rty')
print(res)
```

Pour ces dernières instructions, la première ligne affecte le résultat de `concatene('aze', 'rty')` à la variable `res`. La deuxième ligne affiche la valeur de la variable `res` à l'écran.

## Fonctions renvoyant un nombre aleatoire

Pour utiliser des nombres aléatoires en Python, on importera des fonctions de la bibliothèque `random`.

```python
from random import random
from random import randint
```

La première fonction importée (appelée `random` aussi) retourne un nombre flottant aleatoire entre 0 et 1. La deuxième fonction importée (appelée `randint`) retoune un nombre entier aléatoire entre deux bornes incluses qui sont mises en paramètres.

```
>>> random()
0.005670588016035816
>>> randint(1,15)
9
>>> randint(1,15)
1
```

Par exemple, on peut afficher une série statistique correspondant à 100 lancers de dés à 6 faces numérotés de 1 à 6:

```python
for compteur in range(100):
    print(randint(1,6))
```