Algorithme 1 : Introduction

Mohamed NASSIRI

Objectifs :

- Comprendre les variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Savoir utiliser l'affectation (notée ← en langage naturel)

Mots – clefs :

Variable - Affectation - Type - Entier - Booléen - Flottant - chaîne de caractères -

Prérequis :

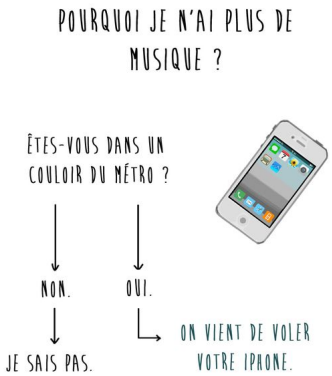
Langage Scratch

La musique du chapitre : Kery James - Constat Amer. Album : Dernier MC. Date de sortie : 2012.

Depuis la naissance de l'ordinateur, leur "performance" n'a fait qu'augmenter. On peut quand même rappeler que le premier ordinateur, l'ENIAC, faisait 30 m de long, 2,40 m de haut et pesait 30 tonnes. Le nombre d'opérations par seconde (en anglais : *floating-point operations per second* ou FLOPS) est une unité de mesure de la performance d'un système informatique. Un ordinateur personnel peut développer une puissance d'environ 200 gigaFLOPS (200.10^9 FLOPS ou encore 200 000 000 000 d'opérations par seconde).

Alors pourquoi tant de baratin ? Tout simplement pour vous dire que si nos ordinateurs sont si performants, bah on va leur demander de faire les calculs (trop longs ou trop répétitifs) à notre place ! Pour cela, on va créer des *algorithmes* (ou *scripts*).

Un exemple d'application des algorithmes sont les *assistants vocaux intelligents* comme Google Home ou Amazon Echo. Quand vous posez une question à votre assistant, vous parlez à un "algorithme" qui analyse votre question et vous répond de la façon la plus pertinente possible en fonction des réponses qu'il a en stock et suivant une logigue en *organigramme*. Comme le montre l'illustration très simplifié (et humouristique) ci-contre



> *« Je crains le jour où la technologie surpassera nos interactions humaines. Le monde aura une génération d'idiots. »*
>
> Albert Einstein

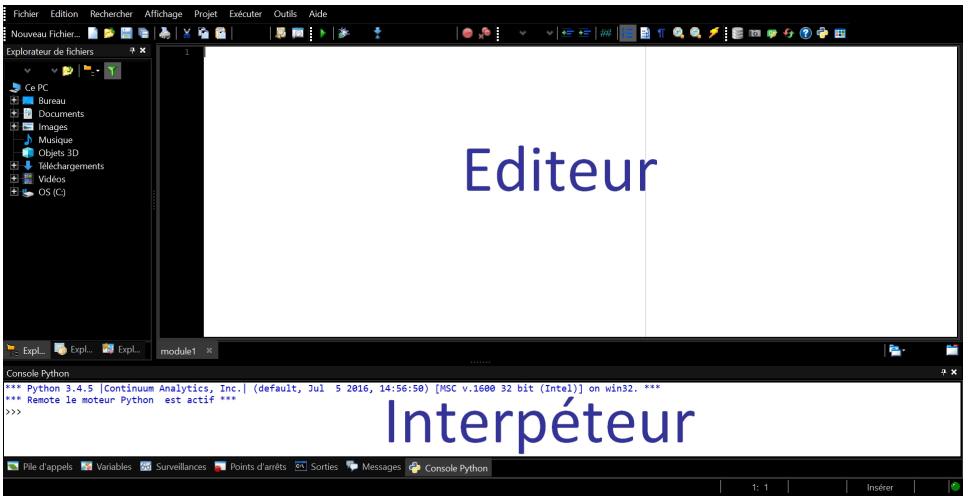
Un peu d'histoire

Muhammad Ibn Mūsā al-Khuwārizmī (780, Khiva - 850, Bagdad) généralement appelé Al-Khwarizmin (latinisé en Algoritmi ou Algorizmi), est un mathématicien, géographe, astrologue et astronome perse, membre de la Maison de la sagesse de Bagdad.
Il fut l'un des premiers à utiliser le terme *racine carrée*.
Regardez cette vidéo très réussie de france.tvéducation qui retrace l'origine des algorithmes !

1 L'interpréteur et l'éditeur

1.1 L'interpréteur

L'interpréteur, c'est lui qui contient le triple chevron >>> (qui signifie que Python attend une commande). L'esprit d'utilisation de l'interpréteur est un peu le même que celui d'une calculatrice.

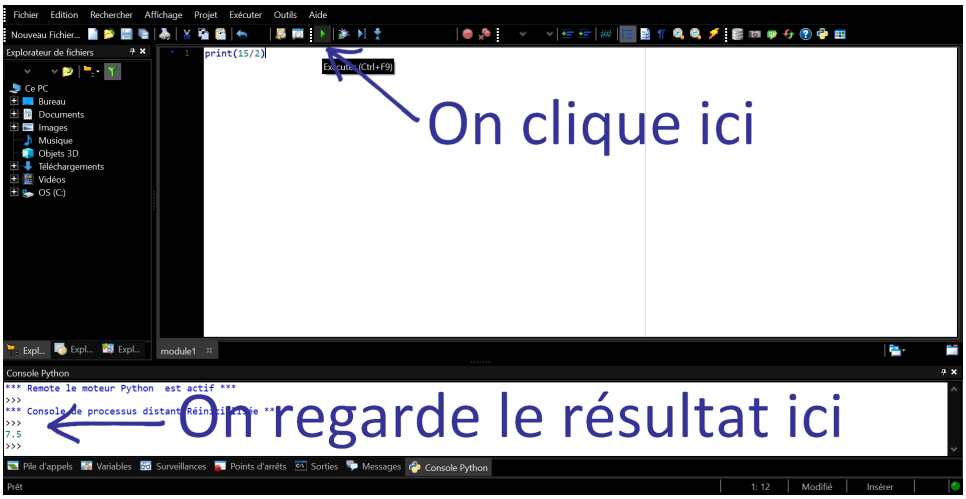



```
>>> 2+5*4
22
```

Remarque
La priorité des opérations est bien respectée !

1.2 L'éditeur

Le but de l'éditeur est de pouvoir y écrire des scripts, c'est-à-dire des (petits ou grands) programmes. Regardons un peu comment cela fonctionne :



2 Calculs, opérations

Dans le tableau ci-dessous, on donne les symboles utilisés pour les opérations de base en Python.

Opérations	**Symboles**	**Exemples**
addition	+	3 + 2 donne 5
soustraction	-	8 - 6 donne 2
multiplication	*	4 * 3 donne 12
puissance	**	2 ** 4 donne 16
division	/	10 / 2 donne 5.5
reste de division entière	%	7 % 3 donne 1
quotient de division entière	//	7 // 3 donne 2

3 Variables

La première chose à comprendre dans le langage algorithmique, c'est la notion d'**objets**, de **variables** et d'**affectations**. Les variables peuvent être de plusieurs types :

Objets	Types	Types en Python
5	entier	**int** (integer)
3.14	flottant	**float**
"Europe"	chaîne de caractères	**str** (string)
$4 > 17/3$	booléen	**bool**

Il est possible de demander à Python de quel type est notre objet avec l'instruction **type** :

```
>>> type(5)
int
>>> type(2.45)
float
>>> type("Europe")
str
```

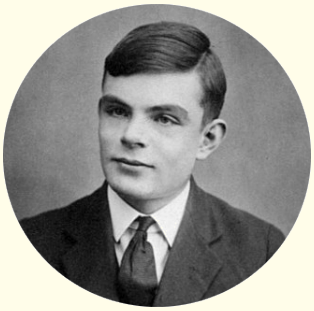
Un peu d'histoire

George Boole (2 novembre 1815, Lincoln - 8 décembre 1864, Ballintemple) est un mathématicien et logicien anglais du XIXè siècle.
Venant d'une famille très modeste, pour subvenir à ses besoins, il devient instituteur à l'âge de 16 ans ! Il fonde même sa propre école alors qu'il a à peine 20 ans !
De 1844 à 1854, il crée une algèbre binaire, dite *booléenne*, n'acceptant que deux valeurs numériques : 0 et 1. Cette algèbre aura de nombreuses applications en téléphonie et en informatique.

4 Affectation

Un peu d'histoire

Alan Mathison Turing (23 juin 1912, Londres - 7 juin 1954, Wilmslow), est un mathématicien et cryptologue britannique, auteur de travaux qui fondent scientifiquement l'informatique. Poursuivi en justice en 1952 pour homosexualité (non non ce n'est pas une blague...), il choisit pour éviter la prison la castration chimique par prise d'œstrogènes. Il est retrouvé mort par empoisonnement au cyanure le 7 juin 1954 dans la chambre de sa maison à Wilmslow. La reine Élisabeth II le reconnaît comme héros de guerre et le gracie à titre posthume en 2013.
Conseil : Allez voir l'excellent film *Imitation Game* (*The Imitation Game*), réalisé par Morten Tyldum (adaptation de la biographie d'Andrew Hodges) avec Benedict Cumberbatch dans le rôle de Turing.



Par exemple, si vous voulez stocker la valeur 3, il faudra choisir une variable (très souvent une lettre) et **stocker** la valeur 3 dans cette variable. Par exemple, en choisissant la variable "c", on peut écrire

```
>>> c=3
```

Et ainsi, vous avez stocké 3 dans la lettre c.

Scratch	
Algorithme	$c \leftarrow 3$
Python	**c=3**

Pour le vérifier, on peut demander à Python ce qu'il y a dans "c" justement avec la commande **print**.

```
>>> print(c)
3
```

Scratch	
Algorithme	Afficher c
Python	**print(c)**

Attention !
En Python, l'égalité a un "sens" ! En effet, en mathématiques, quand on écrit $x = 3$, c'est la même chose que d'écrire $3 = x$. En Python, non ! Quand on affecte une variable, c'est toujours "*variable = objet*". Toujours ! Dans notre exemple, avec **c=3**, pensez à "*c reçoit 3*" et non à "*c égal 3*".

Remarque
Depuis la version 3 de Python, la commande **print** est considérée comme une fonction (on verra les fonctions plus tard ... Soyez patients petits fougueux !) dont il faut alors mettre les arguments entre parenthèses.

On peut également tester si une égalité est vraie ou fausse en utilisant ==. On a affecté 3 à la variable c. On peut lui demander si c est égal à 3 justement :

```
>>> c==3
True
```

Tout à l'heure, nous avons que nous pouvions mettre plusieurs objets dans une variable. Ici, nous faisons un petit tableau récapitulatif

Affecter à c...	Algorithme	Python
Un nombre	$c \leftarrow 3$	**c = 3**
Une chaîne de caractères	$c \leftarrow$"Europe"	**c="Europe"**
Un booléen	$c \leftarrow (4 > 17/3)$	**c = (4 > 17/3)**

Remarque
Affecter une variable c'est lui attribuer une valeur. Le contenu précédent, s'il y en avait un, est effacé.

```
>>> c=3
>>> c=c+1.5
>>> print(c)
4.5
```

Attention !
Il ne faut pas confondre l'égalité mathématique $c = c + 1,5$ (qui est toujours fausse !) avec la nouvelle affectation de la variable c à l'aide de l'ancienne valeur. Il faut comprendre cette égalité de la façon suivante :

$$c_{nouveau} = c_{ancien} + 1,5$$

Pour devenir un crack !
On peut également faire des affectations multiples en Python :

```
>>> x = y = 30
>>> x
30
>>> y
30
```

Avec la notion de *tuple* ou de textitliste, Python permet aussi les affectations *en parallèles* :

```
>>> x, y = (17, Cacatoes)
>>> x, y = [17, Cacatoes]
>>> (x, y) = (17, Cacatoes)
>>> x, y = 17, Cacatoes
>>> x
17
>>> y
Cacatoes
```

5 Chaînes de caractères

Les chaînes de caractères sont un autre type d'objet en Python (et dans la vie de tous les jours aussi d'ailleurs). Pour afficher une chaîne de caractères, il faut utiliser la commande **print**.

```
>>> print("Bonjour Europe")
Bonjour Europe
```

Comme on l'a vu précédemment, on peut mettre ce que l'on veut dans une variable. On peut donc aussi affecter à une variable une chaîne de caractères. Il faut juste faire attention que **print(truc)** et **print("truc")** n'auront pas du tout le même effet, comme le montre l'exemple suivant :

```
>>> comp = "Dunkerque est une belle ville"
>>> print(comp)
Dunkerque est une belle ville
>>> print("comp")
comp
```

L'un des intérêts de la commande **print** couplée avec les chaines de caractères est de produire de belles phrases de fin de programmes. On peut utiliser la commande **print**, qui sait gérer différents types de paramètres pour les afficher sur une même ligne si on les sépare avec des virgules. Par exemple :

```
>>> a = 4
>>> print("Youpitralala ! Le carre de", a, "est", a**2," Youpitralala !")
Youpitralala ! Le carre de 4 est 16
```

Remarque
Remplacer a par une autre valeur, et observer le résultat.

Pour devenir un crack !
On peut aussi utiliser les fonctionnalités de formatage des chaines de caractères comme le montre l'exemple ci-dessous.

```
>>> a = 5
>>> print("Le carre de {} est {}.".format(a, a**2))
Le carre de 5 est 25.
```

6 Modules Python : "math" et "random"

Un module, c'est quoi ? C'est une sorte de bibliothèque (un regroupement de fonctions prédéfinies) qui, une fois importée, permet d'accéder à d'autres fonctions. Il en existe beaucoup ! Parmi ces derniers, j'ai choisi de présenter les deux qui, en classe de seconde au moins, me semblent les plus utiles : **math** et **random**.

6.1 Le module math

C'est un module qui permet d'avoir accès aux fonctions mathématiques les plus courantes comme le cosinus (**cos**), le sinus (**sin**), la racine carrée (**sqrt**), le nombre π (**pi**) et bien d'autres...

```
>>> from math import *  # importation du module
>>> cos(pi)             # cosinus d'un angle en radian
-1.0
>>> sqrt(25)            # racine carrée
5.0
```

Pour devenir un crack !
Attention quand vous importez une bibliothèque ! Vous pouvez par exemple n'importer qu'une fonction. Par exemple, ici, on n'a décidé de n'importer que la fonction racine carrée.

```
>>> from math import sqrt
>>> sqrt(36)
6.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
```

Le message d'erreur signifie simplement que l'on n'a pas importé la fonction cos. Donc si on recommence :

```
>>> from math import sqrt, cos
>>> sqrt(36)
6.0
>>> cos(0)
1.0
```

Le plus simple étant bien évidemment d'importer la bibliothèque **math** grâce à la commande **from math import ***.
Si vous n'écrivez que **from math**, vous devez ajouter **math.** devant toutes vos fonctions ! Un exemple pour comprendre :

```
>>> import math
>>> math.sqrt(36)
6.0
>>> math.cos(0)
1.0
```

6.2 Le module random

En anglais "*random*" signifie "*hasard*". Ce module va nous permettre d'utiliser des fonctions générant des nombres aléatoires. Les deux qui me semblent le plus utile dans un premier temps sont :
- **random()** qui renvoie un nombre aléatoire entre 0 et 1,
- **choice(liste)** qui choisit au hasard un nombre dans une liste donnée.

```
>>> from random import *
>>> random()
0.34461947461259612
>>> random()
0.024762749258158245
>>> L = [7, 3, 8, 5, 6]
```

```
>>> choice(L)
5
>>> choice(L)
3
>>> # cela marche pour une liste de chaine de caractères
>>> liste = ['Jean-Rachid', 'Manon-Imane', 'Kenji-Raymond']
>>> choice(liste)
'Jean-Rachid'
```

6.2.1 Les nombres au hasard

— **Nombre flottant (réel) entre 0 et 1 :**

```
>>> from random import *
>>> random()
0.49852220170348827
```

— **Nombre flottant entre deux bornes.**
Pour tirer un nombre au hasard entre 10 et 12.5 :

```
>>> from random import *
>>> uniform(10, 12.5)
10.005909462232747
```

— **Nombre entier entre deux bornes.**
Pour tirer un nombre entier au hasard entre 0 et 20 :

```
>>> from random import *
>>> randint(0, 20)
20
```

6.2.2 Les listes aléatoires

— **Créer une liste aléatoire de 1000 nombres entiers entre 0 et 100 :**

```
from random import *
liste = []
for i in range(1000):
    liste.append( randint(0, 100) )
```

Remarque
N'ayez pas peur si vous n'avez pas compris... Nous verrons les *boucles* **for** et la commande **append** plus tard.

— **Mélanger une liste :**

```
shuffle(liste)
```

— **Choisir au hasard un élément d'une liste :**

```
choice(liste)
```

— **Extraire au hasard k éléments d'une liste :**
Extrait aléatoirement trois éléments de la liste :

```
sample(liste, 3)
```

Heureux gagnants

Je vous ai montré comment je sélectionne les 5 personnes que je choisi au hasard pour ramasser les exercices grâce à la commande **nbrAléaEnt** sur la *TI-83 Premium CE*.
Maintenant, nous allons fonctionner différemment : nous allons créer une liste que nous allons appeler au nom de la classe, et nous allons rentrer les prénoms de tout le monde. Puis nous exécuterons la commande **sample**.

```
from random import *

Seconde14 = ['Matthéo_Ba', 'Anthony', 'Mézian', 'Raphaël', 'Matthéo_Bo', 'Fiorenzo',
'Loic', 'Lounès', 'Nathan', 'Logan', 'Maxence','Lucas', 'Raffi', 'Hicham', 'Léo',
'Andy', 'Amaël', 'Saber', 'Sébastien', 'Zayd', 'Anthony', 'Quentin']

heureuxgagnants=sample(Seconde14,5)

print("Les heureux gagnants sont",heureuxgagnants)
```

```
>>> Les heureux gagnants sont ['Zayd', 'Lounès', 'Loic', 'Maxence', 'Anthony']
```