Cours Python Basics

Avec

Mr. Emmanuel KOUPOH

(emmanuelkoupoh@gmail.com)

Les Objectifs

1. Apprendre l'algorithmique ;
2. Apprendre la programmation ;
3. Apprendre le langage Python 3 ;

Les Notions

1. Vocabulaire et Définitions ;
2. Variables, Structures (de données) et Opérations ;
3. Contrôle de Flux ;
4. Boucles itératives ;
5. Fonctions et Classes ;
6. Fichiers, Modules, etc.

1. Vocabulaire et Définitions

Algorithme

Un algorithme est une suite finie **d'instructions**, qui une fois exécutée, conduit à un résultat donné. Ce résultat est la solution à un problème.

PS: Une instruction est généralement une ligne de l'algorithme.

1. Vocabulaire et Définitions

Programme Informatique

Un programme informatique, est la transcription d'un algorithme dans un langage de programmation informatique, c'est-à-dire un langage que l'ordinateur comprend. Dans notre cas, ce sera **Python3**.

1. Vocabulaire et Définitions

Variable

Une variable est une boîte qui permet de stocker une donnée (ou information) afin de la mettre à disposition de notre programme durant son exécution.

1. Vocabulaire et Définitions

Donnée

Une donnée est une information que l'on manipule durant l'exécution du programme. On la stocke généralement dans une variable pour la conserver afin de la manipuler plusieurs fois ou de la modifier.

1. Vocabulaire et Définitions

Type d’une donnée

Le type d’une donnée décrit la nature de l’information qui constitue cette donnée.

Ex: Le type numérique, texte, booléen, ...

1. Vocabulaire et Définitions

Déclaration d'une variable

Déclarer une variable ou la créer, c'est en fait lier un identificateur à l'adresse d'un espace mémoire dans lequel nous sauvegardons une donnée.

PS: Le nom ou identificateur de la variable obéit à des règles de nomenclature.

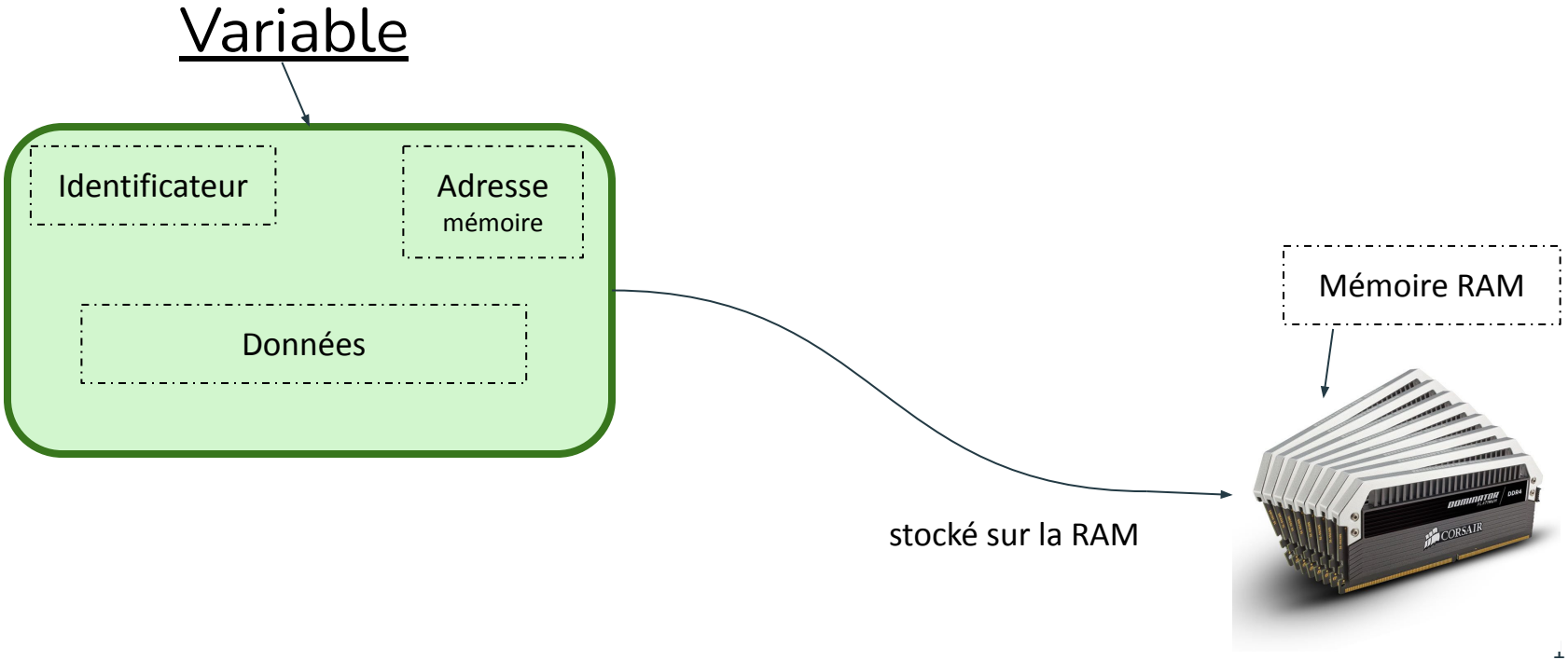
1. Vocabulaire et Définitions

Affectation à une variable

Affecter une valeur ou une donnée à une variable, c'est stocker ladite valeur dans l'espace prévu pour la variable sur la mémoire

PS: Le nom ou identificateur de la variable obéit à des règles de nomenclature.

1. Vocabulaire et Définitions



1. Vocabulaire et Définitions

Règles de nomenclature de variables

Le nom d'une variable doit respecter certaines règles :

- Le premier caractère n'est pas un chiffre ;
- Il ne contient pas de signes de ponctuation, ni d'espace, ni de caractères spéciaux sauf underscore ;
- Il n'est pas un **mot clé protégé** de Python.

1. Vocabulaire et Définitions

Mots clés protégés de Python 3

Ce sont des mots réservés, il ne faut pas les utiliser pour en faire des identificateurs de variables :

```
and, as, assert, break, class, continue, def, del, elif, else, except,
finally, for, from, global, if, import, in, is, lambda, not, or, pass,
raise, return, try, while, with, yield, True, False, None, nonlocal
```

```
or = 4
```

```
  File "<ipython-input-42-5f54c92963ec>", line 1
    or = 4
     ^
SyntaxError: invalid syntax
```

2. Variables, Structures et Opérations

Variable

```python
# Ceci est notre premier code Python
# Les carateres a droite du # sont des commentaires

# Declaration de variables
cours         = "Programmation Python"
classe        = "Licence 2"
ville         = "Kaolack"
duree_en_jour = 3  # on peut aussi mettre un commentaire ici
```

2. Variables, Structures et Opérations

Les Types de variables ou de Données

En programmation informatique, un type de donnée, ou simplement un type, définit la nature des valeurs que peut prendre une donnée ou une variable, ainsi que les opérateurs qui peuvent lui être appliqués.

Type	Description	Exemples
`int` , `float` , `complex`	Valeurs numeriques	`1` , `3.14` , `3+3j`
`string`	Texte	`"Bonjour"`
`bool`	Vrai ou Faux : assertion	`True` , `False`

En Python on utilise la methode predefinie `type()` pour connaitre le type d'une donnée ou d'une variable.

```
type(3+3j)
```

```
complex
```

2. Variables, Structures et Opérations

Les convertions ou changements de types (Casting)

Nous avons parfois besoin de traiter des valeurs d'un type en particulier pour realiser certaines operations, pour ce faire il existe des fonctions Python qui porte le meme nom que les types de bases, a savoir les fonctions `float` , `int` , `complex` , `str` ,..., qui permette de convertir une variable ou une donnee d'un type vers un autre.

2. Variables, Structures et Opérations

Structure de données

Une structure de données est une variable complexe. Elle contient plusieurs données ou informations. Il en existe de plusieurs types : Les listes, les dictionnaires, les ensembles, ...

2. Variables, Structures et Opérations

Structure de données

Les structures de donnees sont des variables qui contiennent plusieurs donnees et/ou plusieurs autres variables. Il y'a (04) type de structures de donnees en Python.

Type	Structure	Manipulation	Particularites
List	ordonné	indice	modifiable
Dict	non ordonné	clé	modifiable
Tuple	ordonnée	indice	non modifiable
Set	non ordonné	valeur	modifiable, unicité des elements

2. Variables, Structures et Opérations

Structure de données

```python
# Les structures de données en Python
une_liste = ["Bonjour", 3,14, ]
dictionnaire = {
                "cours" : "Python3",
                "duree" : 24,
                }

# ensemble des nombres pairs de l'intervalle ouvert 0 - 10
un_ensemble = {2, 4, 6, 8, } # ceci est un ensemble mathematique

# Tuple ou n-uplet
# (x, y, z)
coordonnes = (5.4, 12, 9)
```

2. Variables, Structures et Opérations

Opérations

En Python il existe plusieurs opérations selon le type des variables et des structures de données.

2. Variables, Structures et Opérations

Opérations arithmétiques

Pour effectuer des operations arithmetiques, nous utilisons des operateurs :

Operateur	Objectif	Exemple	Resultat
`+`	Addition	`2 + 3`	`5`
`-`	Soustraction	`3 - 2`	`1`
`*`	Multiplication	`8 * 12`	`96`
`/`	Division	`100 / 7`	`14.28..`
`//`	Division Entiere	`100 // 7`	`14`
`%`	Modulo	`100 % 7`	`2`
`**`	Exponentiation	`5 ** 3`	`125`

2. Variables, Structures et Opérations

Opérations arithmétiques

Lorsque nous effectuons des operations avec des variables, d'autres operateurs peuvent intervenir:

Operateur	Objectif	Exemple	Detail	Resultat
=	Affectation de valeur	`x = 50`	`x = 50`	50
+=	Incrementation	`x += 3`	`x = x + 3`	53
-=	Decrementation	`x -= 1`	`x = x - 1`	52
*=	Multiplication	`x *= 3`	`x = x * 3`	156
/=	Divivsion	`x /= 2`	`x = x / 2`	78.0
%=	Modulo	`x %= 52`	`x = x % 52`	26.0
//=	Divivsion	`x //= 7`	`x = x // 7`	3.0
**=	Exponetiation	`x **= 2`	`x = x ** 2`	9.0

2. Variables, Structures et Opérations

La comparaison

Outre les opérations arithmétiques, Python fournit également plusieurs opérations pour comparer des nombres et des variables.

Operateur	Description
`==`	Vérifiez si les opérandes sont égaux
`!=`	Vérifiez si les opérandes ne sont pas égaux
`>`	Vérifiez si l'opérande gauche est supérieur à l'opérande droit
`<`	Vérifiez si l'opérande gauche est inférieur à l'opérande droit
`>=`	Vérifiez si l'opérande gauche est supérieur ou égal à l'opérande droit
`<=`	Vérifiez si l'opérande gauche est inférieur ou égal à l'opérande droit

```
print( 10 > 15)
```

```
False
```

```
print(8 < 9)
```

```
True
```

```
print(2 < 1)
```

```
False
```

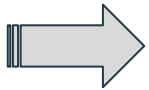
Le résultat d'une opération de comparaison est soit «Vrai» `True` , soit «Faux» `False` . Ce sont des mots clés spéciaux en Python. Essayons quelques expériences avec des opérateurs de comparaison.

2. Variables, Structures et Opérations

Opérations logiques ou booléennes

Operateur	Description
`or`	Vérifiez si l'un de opérandes (ou enonce) est «Vrai» `True`
`and`	Vérifiez si les de opérandes (ou enonce) sont «Vrais» `True`
`not`	Inverse la valeur de l'opérande (ou enonce)

Operateur	Exemple	Explication	Resultat
`or`	(8 < 9) or (2 < 1)	vrai ou faux	`True`
`and`	(8 < 9) and (2 < 1)	vrai et faux	`False`
`not`	not (2 < 1)	pas (faux)	`True`



```
print(( 10 > 15))
```

```
False
```

```
print(not( 10 > 15))
```

```
True
```

```
print((8 < 9) and (2 < 1))
```

```
False
```

```
x = 36

print((x > 13) and (x < 27))
```

```
False
```

```
x = 20

print((x > 13) and (x < 27))
```

```
True
```

2. Variables, Structures et Opérations

Opérations sur les chaînes de caractères

Fonction	Description
`len()`	regardez l'exemple plus bas
`.count()`	regardez l'exemple plus bas
`.upper()`	regardez l'exemple plus bas
`.lower()`	regardez l'exemple plus bas
`.capitalize()`	regardez l'exemple plus bas
`.title()`	regardez l'exemple plus bas
`.split()`	regardez l'exemple plus bas

```
texte = """« Fais ce que ton humanité t'ordonne, n'attend d'applaudissement de personne d'autre que
 toi-même.
Il vit le plus noble et meurt le plus noble, celui qui suit les règles qu'il a créées lui-même.
Toute autre Vie n'est qu'une Mort-Vivante, un monde peuplé de fantômes. »"""
```

```
print(texte)
```

```
« Fais ce que ton humanité t'ordonne, n'attend d'applaudissement de personne d'autre que toi-même.
Il vit le plus noble et meurt le plus noble, celui qui suit les règles qu'il a créées lui-même.
Toute autre Vie n'est qu'une Mort-Vivante, un monde peuplé de fantômes. »
```

```
print( len(texte) )
```

```
268
```

```
print( texte.count("f") )
```

```
1
```

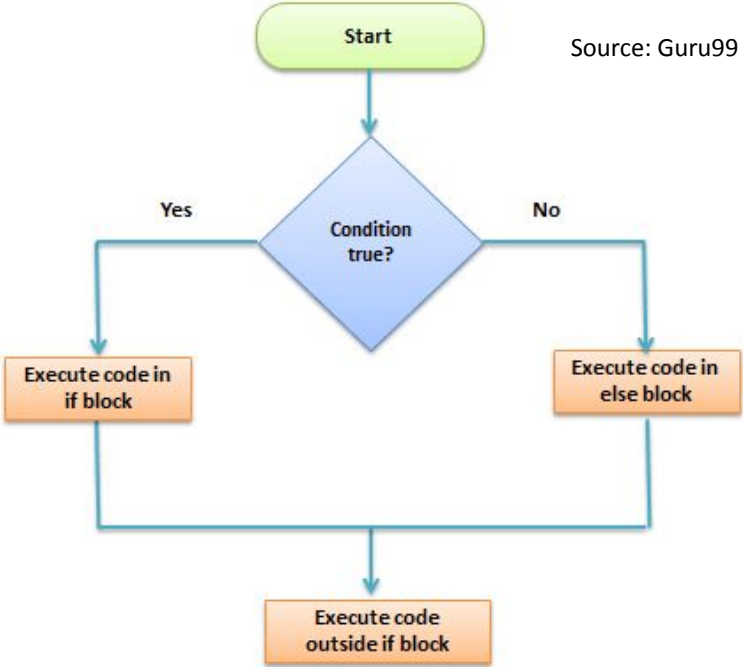
```
print( texte.upper() )
```

```
« FAIS CE QUE TON HUMANITÉ T'ORDONNE, N'ATTEND D'APPLAUDISSEMENT DE PERSONNE D'AUTRE QUE TOI-MÊME.
IL VIT LE PLUS NOBLE ET MEURT LE PLUS NOBLE, CELUI QUI SUIT LES RÈGLES QU'IL A CRÉÉES LUI-MÊME.
TOUTE AUTRE VIE N'EST QU'UNE MORT-VIVANTE, UN MONDE PEUPLÉ DE FANTÔMES. »
```

3. Contrôle de Flux

Définition

C’est un outil qui permet de changer le comportement d’un programme selon conditions importantes. Les conditions peut être vraies ou fausses.



Source: Guru99

3. Contrôle de Flux

Ici, l'ordinateur va parcourir le code de haut en bas.
Chaque condition sera vérifiée jusqu'à la "condition vraie" et les instructions, du bloc qui est vrai, seront exécutées. Puis on sortira du contrôle de flux.

```
# Petit exemple
age = input("Tapez votre age svp...\n>>")# la valeur récupérée avec input est de type string
# Convertir la texte entre en valeur numérique, en entier
age = int( age )

if age < 0 :
    print("Il y'a un souci, vous n'etes pas encore venu au monde...")

elif ( age >= 0 ) and ( age <= 14 ):
    print("Vous etes un ado")

elif ( age > 18 ) and ( age <= 24 ):
    print("Vous etes un jeune adulte..., il faut penser a carrière professionnelle")

elif ( age > 24 ) and ( age <= 65 ):
    print("J'espere que vous avez un travail..., il faut bosser maintenant")

else:
    print("Comment se passe votre retraitre ?")
```

4. Boucles itératives

Définition

Ce sont des outils qui permettent d'exécuter plusieurs fois un ensemble d’instructions. L'exécution de l’ensemble d’instructions est répétée, soit un nombre de fois défini **(boucle for)**, soit un tant qu’une condition est vraie **(boucle while)**.

4. Boucles itératives -- For (pour)

Ici, l'ordinateur va parcourir le code de haut en bas.
Chaque élément de la liste **petite_liste** sera récupéré et stocké de façon temporaire dans la variable **pointeur**. Puis on utilisera la variable pointeur à l'intérieur de la boucle.

```
#Exemple boucle for
compteur = 1 # Pour compter le nombre d'iterations

petite_liste = [0, 1, 2, 3, 4] # Liste a parcourir

for pointeur in petite_liste:
    print(f"iteration {compteur} -- valeur : {pointeur}")

    compteur += 1 # Incrementer le nombre d'iterations
    #compteur est egal a 1 au debut du programme, puis 2...
    #apres la derniere execution compteur sera 5+1
```

4. Boucles itératives -- While (tant que)

Avant d'entrer dans la boucle et après chaque itération la condition sera vérifiée. Les instructions du bloc ne seront exécutées que si la "condition vraie". Le print sera exécuté lorsque la condition sera fausse.

```python
#Exemple boucle while
compteur = 1 # Pour compter le nombre d'itérations

valeur = 2 # élément de la condition

# Tant que la condition est vraie on execute le bloc (les instructions indentées)
while (valeur < 7) :
    print(f"iteration {compteur} -- valeur : {valeur}")

    valeur += 1 # on incrémente la variable pour ne pas que rester à : valeur = 2

    compteur += 1 # Incrémenter le nombre d'itérations
    #compteur est egal à 1 au debut du programme, puis 2...
    #apres la derniere execution compteur sera 4+1

print("La boucle while a fini son exécution.")
```

5. Fonctions et Classes

Fonction

Une fonction est un bloc de code réalisant une tâche spécifique (et/ou redondante) de notre programme principal. Pour une meilleure organisation on définit une fonction qu'on appelle en faisant varier ses arguments à notre guise. Le bloc de code devient donc une instruction.

Ex: La fonction de calcul de somme, de moyenne, ...

5. Fonctions et Classes

Fonction : Compléments

- Il y’a des fonctions qui retournent des informations, celles qui n’en retournent pas sont appelées des procédures.
- Les fonctions peuvent avoir des arguments ou paramètres qui permettent de leur transmettre à l’appel des données à utiliser durant leur exécution.

5. Fonctions et Classes

Classe

Une classe est un moule ou un modèle qui permet de créer des super-variables qui peuvent même disposer de fonctions particulières.

Ex: La **classe Voiture**, qui permet de créer plusieurs représentations de voiture, des **objets Voiture**.

6. Fichiers, Modules, etc.

Fichier

Un fichier est une donnée ou un ensemble d'informations sauvegardé sur une mémoire permanente (disque dur, clé usb, etc) avec une extension (.pdf, .mp4) qui décrit son contenu et la manière de le manipuler.

6. Fichiers, Modules, etc.

Module

Un module est un ensemble de classes et de fonctions écrites dans un ou plusieurs fichiers et qu'on destine à une utilisation spécifique ou un domaine d'activité (manipulation de coordonnées Géographiques).

6. Fichiers, Modules, etc.

Package ou Librairie

Un package est un ensemble de modules qu'on organise pour servir dans un domaine d'activité (la télédétection, les mathématiques, la physique).

6. Fichiers, Modules, etc.

Utilisation de package

Pour utiliser une package en Python, on l'installe quand il n'est pas pré-intégré, puis on l'importe ou on importe l'un de ses module, classe, ou fonction qu'on utilise enfin dans notre programme.

FIN

Pour toutes modifications ou ajouts, veuillez me contacter par mail.

emmanuelkoupoh@gmail.com