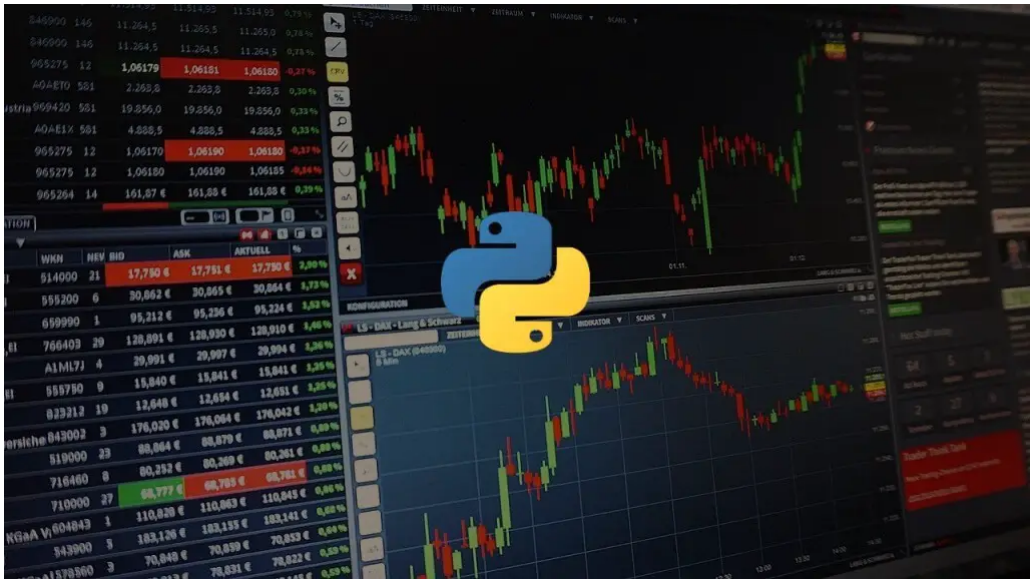
Informatique
Les bases du langage Python
Cours

Soeung Raphaël

22 juillet 2023



SOMMAIRE

I Préliminaires

A Comment travailler avec Python ?

L'informatique est de nos jours ubiquitaire [1] et peut être pratiqué sur tout ordinateur, tablette ou smartphone [2]. Je recommande toutefois de le travailler sur un ordinateur, d'une part pour avoir une taille d'écran suffisante, d'autre part pour ressentir la sensation de taper sur un clavier. Il faudra en premier lieu télécharger l'interprète Python, puis un simple éditeur de texte suffit pour composer les codes. Toutefois, par souci de lisibilité et d'intelligibilité [3], nous privilégions l'utilisation d'un environnement de développement intégré (ou IDE en anglais) comme PyCharm Community.

1. Se dit de l'informatique lorsqu'elle est omniprésente dans un environnement.
2. Python est un langage interprété par un logiciel qu'on appelle l'interprète. L'interprète exécute chaque ligne du code source (celui que l'on tape) indépendamment de la machine.
3. Aisance à comprendre.

II Concepts primordiaux

A Python, un langage de programmation multi-paradigme

Définition — Informatique
L'informatique est la science du traitement automatique et rationnel de l'information par un système concret ou abstrait.

Puisque l'informatique traite de l'information, il faut nécessairement un langage. Or, la maîtrise d'un langage nécessite la maîtrise des définitions des mots employés. Il est alors essentiel d'apprendre les définitions des concepts.

Remarque
Tandis qu'en anglais on distingue l'informatique comme science abstraite (Computer Sciences) et l'informatique comme science concrète appliquée (Computer Engineering), en français on parle seulement d'informatique pour désigne les deux car l'une ne va pas sans l'autre. Il en est de même pour la physique : que ferait-on de la physique théorique sans expérience ? Que ferait-on de l'informatique sans l'électronique.

Définition — Abstraction
Une abstraction consiste à représenter des objets qui appartiennent au monde réel dans le monde du programme que l'on écrit. en identifiant et regroupant des caractéristiques et traitements communs applicables à des entités ou concepts variés ; une représentation abstraite commune de tels objets permet d'en simplifier et d'en unifier la manipulation.

Définition — Paradigme
Un paradigme de programmation est un ensemble d'abstractions qui régissent la manière de transcrire les problèmes réels en langage informatique.

Python est un langage multi-paradigme, dans le sens où il permet de programmer selon différents paradigmes, en l'occurrence trois : le paradigme impératif, le paradigme objet et le paradigme fonctionnel.

Définition — État
Dans le contexte des processus, un état est l'ensemble formé par les données entrées, les données temporaires, les données calculées et le pointeur d'instruction (et généralement l'état des registres) par un processus. informatique.

Définition — Paradigme impératif
Le paradigme impératif s'attache à décrire des séquences d'instructions (ordres) qui agissent sur un état interne de la machine (contexte). L'impératif explicite le comment procéder pour exécuter un programme. Cette programmation se rapproche de la logique électronique des processeurs.

Définition — Paradigme objet
Ce paradigme est une déclinaison de l'impératif et propose de décrire un programme comme l'interaction entre des objets à définir. Une classe est un type d'objet qui possède des attributs et des comportements. Ces caractéristiques sont encapsulées et peuvent être masquées à l'utilisateur d'un objet : cela permet de protéger l'intégrité de l'objet et de garantir une cohérence dans la manipulation des données.

Définition — Paradigme fonctionnel
Le paradigme fonctionnel est une déclinaison du déclaratif qui considère qu'un programme n'est qu'un calcul et qu'un calcul est le résultat d'une fonction. Le mot fonction est ici à prendre au sens mathématique du terme (lambda calcul) : une fonction appelée avec les mêmes paramètres produit le même résultat en toute circonstance.

B Algorithmes, programmes et processus

L'informatique, tout comme la physique ou la chimie, fait appel aux mathématiques pour modéliser un problème et démontrer des propriétés. Mais à la différence des mathématiques qui se permettent souvent de manipuler des objets qu'on ne sait pas construire, l'informatique est une science constructiviste : lorsqu'on résout un problème en informatique, on construit la solution à l'aide un algorithme.

Définition — Algorithme
Un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre une classe de problèmes.

Les bonnes pratiques dans l'élaboration d'un algorithme exigent que :
— les entrées et sorties soient identifiées clairement,

— la description de l'algorithme soit suffisamment abstraite pour rendre son implémentation indépendante du langage de programmation choisi.

Exemple — Algorithme calculant le produit de deux entiers naturels

Algorithme 1 Produit de deux entiers naturels

1: **Fonction** PRODUIT(a, b) ▷ $a \in \mathbb{N}$ et $b \in \mathbb{N}$.
2: $p \leftarrow 0$
3: $c \leftarrow 0$ ▷ c et p sont des entiers naturels.
4: **tant que** $c \leqslant a$ **faire**
5: $p \leftarrow p + b$
6: $c \leftarrow c + 1$
7: **renvoyer** p

L'élaboration d'un algorithme se fait en bricolant, en tâtonnant, et parfois en dessinant.

Définition — Programme
Un programme est un ensemble d'instructions dans un langage de programmation donné.

Exemple — Programme en Python qui traduit l'algorithme calculant le produit de deux entiers naturels

```
def produit(a, b):
  p = 0
  c = 0
  while c <= a:
    p = p + b
    c = c + 1
  return p
```

Le programme est alors l'expression concrète de l'algorithme qui est abstrait, c'est-à-dire qu'il est interprétable soit directement par un processeur, soit par un interpréteur (machine virtuelle). On peut donc demander à une machine d'exécuter un programme, mais on ne peut pas lui demander d'exécuter un algorithme.

Définition — Processus
Un processus est un programme en cours d'exécution sur une machine, c'est-à-dire un processeur muni d'une mémoire et d'un système d'exploitation.

III Fonctions, affichage d'un objet et lecture d'une entrée en Python

A Procédures et fonctions

Définition — Routine ou procédure
En informatique, une routine est un bloc de code nommé qui correspond à une suite d'instructions créées pour effectuer une tâche précise, regroupées ensemble et qu'on va pouvoir exécuter autant de fois qu'on le souhaite en "l'appelant" avec son nom.

Définition — Fonction
En informatique, une fonction est une routine qui renvoie une valeur de fin.

On peut ainsi faire une analogie avec les mathématiques : une fonction est une relation, qui à chaque valeur de la variable d'entrée, fait correspondre au plus une valeur de y.

Malheureusement, beaucoup de gens confondent routine et fonction et par abus de langage, appellent fonctions les procédures.

Bien entendu, puisque nous sommes des gens honnêtes et rigoureux. Nous ne devons pas confondre ces deux concepts.

En Python, on définit une fonction de la manière suivante :

```
def nom_de_la_fonction(variable_d_entree):
  suite des instructions
  return valeur_de_fin
```

Exemple — Définition d'une fonction somme qui à tout couple $(a, b) \in \mathbb{R}$ associe $a + b$

```
def somme(a,b):
  return a+b
```

B Affichage d'un objet et lecture d'une entrée

Nous ne rentrerons pas dans les détails de ce qui se passe quand on fait afficher en Python un objet car il est trop ambitieux et hors de ma portée de faire cela. Nous nous contenterons de voir comment (quelle commande entrer pour) faire afficher un objet.

Dans un script Python, pour afficher un objet, on fait généralement, à notre niveau, appel à la commande `print` (qui est une fonction). L'appel à la fonction `print` doit se faire avec des parenthèses comme dans l'exemple ci-dessous. L'affichage se fait alors dans le shell[4] intéractif.

Exemple — Affichage en Python de la chaîne de caractères "Hello world !"

```
print("Hello world!")
```

Exercice ★

Écrire un code en Python pour afficher la chaîne de caractère "Je suis une bonne élève." (il n'y a pas de honte à s'encourager de temps, quitte à mentir).

On peut aussi lire une entrée dans le shell d'un utilisateur avec la commande `input`. Cette fonction provoque une interruption dans le programme courant. L'utilisateur est invité à entrer des caractères au clavier et à terminer avec <Enter>. Lorsque cette touche est enfoncée, l'exécution du programme se poursuit, et la fonction fournit en retour une valeur correspondant à ce que l'utilisateur a entré. Cette valeur peut alors être assignée à une variable quelconque. On peut invoquer la fonction `input` en laissant les parenthèses vides. On peut aussi y placer en argument un message explicatif destiné à l'utilisateur.

Exemple — Code en Python pour invoquer un drôle de perroquet

```
print(input("Je suis un bon perroquet, croyez-moi. Dites quelque
    chose que je vous le repete"))
```

4. Ceci est un anglicisme informatique. En français, on appelle cela l'interface utilisateur d'un système d'exploitation. C'est un programme destiné à lancer d'autres programmes et gérer leurs intéractions. Dans le langage courant de l'informaticien, le shell désigne d'une interface en ligne de commande.

IV Types et opérateurs

A Types

Puisque l'informatique traite l'information, il faut être capable de stocker l'information sous une forme accessible au traitement informatique. On choisit un type de données pour représenter l'information qui diffère selon la nature de cette dernière.

Définition — Typage
Le typage désigne l'action de choisir une représentation à une donnée selon ses caractéristiques. Cette représentation est nommée type. Le typage est effectué soit par le programmeur, soit par le compilateur soit par l'interpréteur.

Définition — Type simple
Les types simples correspondent à des informations comme les nombres, des constantes ou des valeurs booléennes.

En Python, les types simples sont :
— `int` qui permet de représenter un sous-ensemble des entiers relatifs,
— `float` qui permet de représenter un sous-ensemble des nombres décimaux,
— `complex` qui permet de représenter un sous-ensemble des nombres complexes,
— `bool` qui permet de représenter une valeur booléenne, `True` ou `False`.

Définition — Type composé
Un type composé est un type de données qui agrège des types simples dans un objet homogène (tableau, énumération) ou inhomogène (structure, union). Dans les types composés (in)homogènes, les données (ne) sont (pas) toutes du même type.

En Python, les types composés sont les chaînes de caractères, les tuples, les listes, les ensembles et les dictionnaires.

On peut obtenir le type d'un objet avec la fonction `type`.

Exemple — Affichage du type de divers objets

```
print(type(2)) # <class 'int'>
print(type(2.)) # <class 'float'>
print(type("a")) # <class 'str'>
print(type((3, "abc"))) # <class 'tuple'>
print(type(print)) # <class 'builtin_function_or_method'>
```

```
print(type([4, "cba"])) # <class 'list'>
print(type({3, 4, "ceci est un element d'un ensemble"})) # <class '
    set'>
print(type({3:2, 4:"ceci est la valeur liee a la clef 4", "ceci est
    une clef du dictionnaire":1})) # <class 'dict'>
```

Définition — Inférence de type
L'inférence de types est un mécanisme qui permet à un compilateur ou un interpréteur de rechercher automatiquement les types associés à des expressions, sans qu'ils soient indiqués explicitement dans le code source. Il s'agit pour le compilateur ou l'interpréteur de trouver le type le plus général que puisse prendre l'expression. Les avantages à disposer de ce mécanisme sont multiples : le code source est plus aéré, le développeur n'a pas à se soucier de retenir les noms de types, l'interpréteur fournit un moyen au développeur de vérifier (en partie) le code qu'il a écrit et le programme est peu modifié en cas de changement de structure de données.

C'est l'inférence de type en Python qui fait que l'interprète Python reconnaît 2 comme étant un int et 2. comme étant un float.

Définition — Typage explicite
Un langage est dit à typage explicite s'il exige de toute donnée qu'elle soit déclarée selon un type. Sinon c'est un langage à typage implicite.

Python est un langage à typage implicite car il n'impose pas que l'on déclare chaque variable avec son type.

Définition — Typage dynamique
Un langage est dit à typage dynamique si une variable peut changer de type au cours de l'exécution du programme. Sinon, on parle de typage statique.

L'exemple ci-dessous illustre le fait que Python est un langage à typage dynamique.

Exemple — Python, un langage à typage dynamique

```
a = 6
print(type(a)) # <class 'int'>

a = 6.
print(type(a)) # <class 'float'>
```

Définition — Transtypage
Transtyper une variable c'est modifier son type en opérant une conversion de la donnée.

Remarque
En anglais, le transtypage se dit *type casting* ou *type conversion*.

Exemple — Transtypage de variables en Python

```
a = 2
b = float(a)
print(type(a), b, type(b)) # <class 'int'> 2.0 <class 'float'>

a = "2"
b = int(2)
print(type(a), b, type(b)) # <class 'str'> 2.0 <class 'int'>
```

Exercice ★
Remplir le code en Python suivant afin qu'il n'y ait pas d'erreur à l'exécution. Expliquez alors ce qui est affiché dans la console (shell).

```
a = 3
b = "567"
b =
print(a+b)
```

B Opérateurs

Les opérateurs peuvent être classés en catégories selon la nature de leur action, le type ou le nombre de leurs opérandes.

B.1 Opérateurs arithmétiques

En Python, il y a un transtypage implicite (que l'interprète fait) qui permet d'utiliser les mêmes opérateurs pour les entiers et pour les flottants. Ces opérateurs sont les transpositions informatiques des opérateurs mathématiques : ils agissent sur des représentations de nombre en machine, les types numériques. Pour éviter toute

ambiguïté dans l'interprétation, Python attribue une priorité à chaque opérateur. Le tableau ci-dessous présente les priorités des opérateurs ainsi que leur description.

Priorité	Opérateurs	Description des opérateurs
1	()	parenthèses
2	**	exponentiation
3	+x -x ~x	plus et moins unaire, négation bits à bits
4	* / // %	multiplication, division, division entière, modulo
5	+ -	addition, soustraction
6	<< >> >>= <<=	décalage binaire
7	&	et bits à bits
8	^	ou exclusif bits à bits
9	|	ou bits à bits
10	== =! > >= < <= is,is not,in,not in	identité comparaison appartenance
11	not	négation logique
12	and	et logique
13	or	ou logique

Tableau – Priorités des opérateurs en Python : dans l'ordre d'apparition, du plus prioritaire (1) au moins prioritaire (13)

Bien que les opérateurs +, - et * s'appliquent indifféremment aux types `int` et `float`, le résultat d'une opération entre `int` sera un `int` et si l'une des deux opérandes est un `float` alors le résultat est un `float`. C'est encore une fois le transtypage implicite qui fait que l'interprète Python transtype un `int` en `float` pour réaliser l'opération de manière transparente.

B.2 Opérateurs logiques

Définition — Opération logique
Les opérateurs logiques produisent une valeur booléenne à partir d'autres valeurs booléennes en les combinant. Ils prennent le plus souvent deux opérandes.

Les opérateurs logiques en Python sont :

— la conjonction `and` (`a and b` renvoie `True` si et seulement si `a` et `b` sont tous les deux vrais),
— la disjonction `or` (`a or b` renvoie `False` si et seulement si `a` et `b` sont tous les deux faux),
— la négation `not` (`not a` renvoie `True` si et seulement si `a` est faux).

B.3 Opérateurs d'affectation

Définition — Affectation
L'affectation est l'opération qui consiste à assigner une valeur à une variable et donc de modifier son état en mémoire.

En Python on réalise l'affectation simple à une variable avec l'opérateur `=` après le `nom` d'une variable. On peut aussi réaliser une affectation combinée avec une addition `+=`, une soustraction `-=`, une multiplication `*=`, etc. L'affectation combinée évite à l'interprète de créer une variable intermédiaire pour le calcul. L'opération se fait alors en place, c'est-à-dire sur l'espace mémoire même associé à la variable.

B.4 Opérateurs sur les chaînes de caractères

Les chaînes de caractères sont des objets immuables[5] en Python. L'opérateur `+` utilisé entre deux chaînes de caractères les concatène dans une nouvelle chaîne de caractères. L'opérateur `*` prenant un entier naturel `n` et une chaîne de caractères `c` pour opérandes, fait la concaténation `n` fois de `c`.

Exemple — Opérateurs sur les chaînes de caractères

```
1 c = "Je suis jeune ! "
2 print("Suis-je jeune ? " + c) # Suis-je jeune ? Je suis jeune !
3 print(3 * c) # Je suis jeune ! Je suis jeune ! Je suis jeune !
```

5. Qu'on ne peut pas modifier

V Les listes

Les listes sont des objets très puissants en Python qui sont muables[6] et qui permettent de stocker une collection d'objets (pas forcément du même type) indexés par leur ordre dans la liste.

A Constructeurs de listes

A.1 Crochets

On peut définir une liste à l'aide de crochets.

Exemple — Utilisation des crochets pour définir une liste

```
L = [] # Ceci est une liste vide.
print(L, type(L)) # [] <class 'list'>
L = [1, 2, 3, "surprise !"]
print(L) # [1, 2, 3, "surprise !"]
```

6. Qu'on peut modifier.