Python et variables

L'**informatique** est **omniprésente** dans le monde du travail. Être capable de s'en servir est nécessaire pour tous **futurs ingénieurs** comme **futurs chercheurs**.
L'essor de nouvelles technologies comme les smartphones puis les tablettes a permis de multiplier et diversifier l'utilisation de l'informatique. Pour autant, l'**ingénierie informatique se développe sur ordinateur**, pour des raisons simples de puissance de calcul et de stockage de données.

1 | Les langages informatiques

Les logiciels et applications que nous utilisons au quotidien sont **programmés en langage informatique**, plus exactement en **langage de programmation**.

Définition

Langage de programmation

Les développeurs informatiques communiquent avec la machine via des langages de programmation pour **ordonner les instructions** que la machine doit exécuter afin de réaliser un programme. Les langages de programmation sont un type un **langage informatique**.
Comme les langues vivantes, les **langages de programmation** ont leur propre **syntaxe**, **vocabulaire**,...

Le tableau 1 catégorise grossièrement quelques langages informatiques les plus courants :

Système embarqué / performance	Web	Applications / logiciels	Scripts / automatisation / data
C, C+	HTML, CSS, JavaScript, PHP	Java, C#	Python, Lua

Tableau 1 – Catégorisation grossière de quelques langages informatiques

1.1 Le Python 

Python a été choisi pour être utilisé en CPGE parce qu'il est **simple à prendre en main** et permet de se concentrer sur la **logique plutôt** que sur la **syntaxe**.
C'est un langage **flexible** et **polyvalent**. Il est utilisé aussi bien pour des petits scripts que pour des applications complexes. Ce n'est pas qu'un **langage d'apprentissage**, Python est utilisé dans l'**industrie** et dans la **recherche**.
Il existe de très nombreuses bibliothèques adaptées à la modélisation, aux simulations et à l'analyse de données, par exemple ; ce qui permet de ne pas réinventer la roue.

1.1.1 Interpréteur Python

Python est un langage de **haut niveau d'abstraction**. Cela signifie que l'écriture en Python ne suffit pas à la machine pour comprendre ce qui est demandé par le code informatique. La machine doit **interpréter** le code Python pour en comprendre un langage machine, on parle d'**interpréteur Python**.
C'est justement grâce à cette interprétation qu'il s'agit d'un langage simple à prendre en main et flexible. En revanche, on comprend aisément qu'interpréter nécessite plus de calcul à la machine. Python est un **langage plutôt gourmand**, notamment en comparaison des langages compilés comme le C ou le C++.

1.1.2 Langage séquentiel

Python suit une logique de programmation séquentielle. C'est-à-dire que le programme exécute des instructions **de manière successive**, ce qui donne une séquence d'instructions.

> **Remarque :** Autrement dit, en Python, la ligne 2 d'un script s'exécute après la ligne 1 et avant la ligne 3.

1.2 Les environnements de développement

Les codes informatiques sont des textes ; a priori, ils peuvent être rédigés sur n'importe quel éditeur de texte. Cependant le texte doit suivre des règles très précises afin qu'il soit bien interprété.
C'est pourquoi, les programmes informatiques sont rédigés dans des **environnements de développement** ou **IDE** (Integrated Development Environment). Ces **logiciels** sont **dédiés à la programmation**, ils exécutent le code, facilitent son développement, permettent le debug... Au lycée, l'IDE est **Spyder**. Il sera utilisé en ITC, SI, Physique, Chimie... [1].

2 Les variables

Les données sont des informations (des chiffres, des lettres...) qui sont manipulées au cours du programme informatique. Ces **données** sont **stockées** dans des **variables**. Ces dernières sont **la base de toute l'informatique** que nous allons développer en CPGE.

Pour bien comprendre les variables, il est préférable de rappeler les constituants d'un ordinateur.

2.1 Architecture d'un ordinateur

Question 1 : Faire l'association entre les textes explicatifs et le nom des composants parmi la liste suivante : processeur graphique, registre et mémoire cache, bloc d'alimentation, mémoire morte, processeur, mémoire vive, carte mère, stockage de masse.

- Le **processeur** ou **CPU** (central processing unit) est le cerveau de l'ordinateur. Il est chargé d'**interpréter les instructions et d'effectuer les opérations** nécessaires au traitement des données. Il est caractérisé en partie par sa **fréquence** d'horloge (ou cycle) exprimée en Hertz correspondant au nombre d'impulsions (et indirectement, d'instructions élémentaires) qu'il peut effectuer par seconde. Un processeur de $1GHz$ peut donc traiter un milliard d'instructions élémentaires chaque seconde.

Question 2 : Faire l'association entre les textes explicatifs et le nom des composants parmi la liste suivante : processeur graphique, registre et mémoire cache, bloc d'alimentation, mémoire morte, processeur, mémoire vive, carte mère, stockage de masse.

1. Se référer au livret de vacances pour son installation. Si vous avez déjà une distribution Python et/ou un IDE, vous êtes libres de continuer à les utiliser.

- **Mémoires proches du processeur**
  - ◇ Les **registres** et la **mémoire cache** sont des **mémoires internes au processeur** très proches donc très rapides et de très petite taille. La mémoire **cache sert**, en résumé, **de mémoire tampon**, où sont stockées temporairement les données qui vont être utilisées bientôt.
  - ◇ La **mémoire vive** ou **RAM** (Random Access Memory) est une **mémoire volatile**. Composée essentiellement de condensateurs qui se déchargent dès coupure de l'alimentation électrique. **Les données** qu'elle contient sont donc vouées à **disparaître** dès que l'**ordinateur** s'**éteint**. Elle est utilisée pour la mise en mémoire des données temporaires nécessaires qui serviront pour des calculs effectués par le processeur et se caractérise par un temps d'accès relativement court.

Question 3 : Faire l'association entre les textes explicatifs et le nom des composants parmi la liste suivante : processeur graphique, registre et mémoire cache, bloc d'alimentation, mémoire morte, processeur, mémoire vive, carte mère, stockage de masse.

- **Mémoires non volatiles**
  - ◇ La **mémoire morte** ou **ROM** (Read-Only Memory) contient entre autres le BIOS et ne peut être effacée ou reprogrammée simplement. Comme son nom l'indique, elle est destinée à une lecture seule.
  - ◇ Les **stockages de masse** sont des mémoires utilisées pour conserver des **données permanentes** (donc non volatiles). Elles peuvent aujourd'hui avoir de très grosses capacités de stockage, en revanche leur temps d'accès est plus important. *Point culture : sur les ordinateurs nous sommes passés des SSD connectés en filaire aux SSD en barrette connectés directement sur la carte mère, ce qui réduit considérablement le temps de trajet et donc le temps d'accès.*

Question 4 : Faire l'association entre les textes explicatifs et le nom des composants parmi la liste suivante : processeur graphique, registre et mémoire cache, bloc d'alimentation, mémoire morte, processeur, mémoire vive, carte mère, stockage de masse.

- Le **processeur graphique** ou **GPU** (Graphical Process Unit) est une unité de calcul **optimisée** pour effectuer des **calculs en parallèle**, ce qui est très utile par exemple pour le calcul matriciel et le rendu d'image.
- Le bloc d'alimentation adapte et sécurise l'énergie électrique du secteur pour les différents composants.
- La **carte mère** ou motherboard est une carte électronique où tous les autres composants s'y connectent. Elle abrite tous les **bus de données** et connectiques vers les **périphériques externes** (clavier, écran(s), souris...).

Remarque : À quelques différences près (notamment l'optimisation de la consommation énergétique), la logique de fonctionnement des composants d'un smartphone ou d'une tablette se recoupe avec celle d'un ordinateur.

Enfin, tout ordinateur a besoin d'un logiciel pour gérer ses **ressources matérielles** en fonction de la tâche qui lui est demandée. Les logiciels remplissant cette fonction s'appellent un **système d'exploitation**, ou **OS** (Operating System).

Exemple

Le système d'exploitation des ordinateurs du lycée est Windows. Les deux autres OS les plus courants sont macOS et Linux, qui viennent tous deux d'UNIX. Les smartphones et les tablettes ont eux aussi un OS, souvent iOS (apple) ou Android.

2.2 Variable et mémoire

La **mémoire** est **structurée** comme un **tableau** dont une case (cellule) correspond à un **octet** (**8 bits**) de données. Pour retrouver les données, chaque case (octet) est repérée par une adresse :

l'**adresse mémoire**. D'un point de vue utilisateur, nous donnons **un nom à nos données**. Le système gère un **répertoire** faisant **correspondre** le **nom** lié à cette donnée **et** son **adresse mémoire**.

Définition

Variable

Une variable est l'attribution d'un **espace mémoire** pour stocker une **donnée**. Une **variable** est notamment **caractérisée** par un **nom** et une **valeur**.
En fonction de la donnée que nous mettons dans une variable, il faut réserver plus ou moins d'espace mémoire. Aussi, la donnée est **codée différemment en fonction de sa nature** (nombre, caractère, ...). Il faut impérativement **indiquer** au processeur de quel **type** est la **donnée** ; pour cela on ajoute au nom et la valeur de la **variable**, une troisième caractéristique le **type**.

Remarques :

- Le nom des variables est à choisir intelligemment : il faut **toujours utiliser des noms explicites** (quitte à avoir des noms de variables un peu longs).
- Supprimer une variable ne fait que supprimer la correspondance et n'efface pas la (ou les) case(s) mémoire(s) correspondante(s)...
- L'ensemble des variables ainsi que les valeurs stockées dans ces variables à un instant t définissent le « contexte » (accessible sur Spyder : Figure 1)

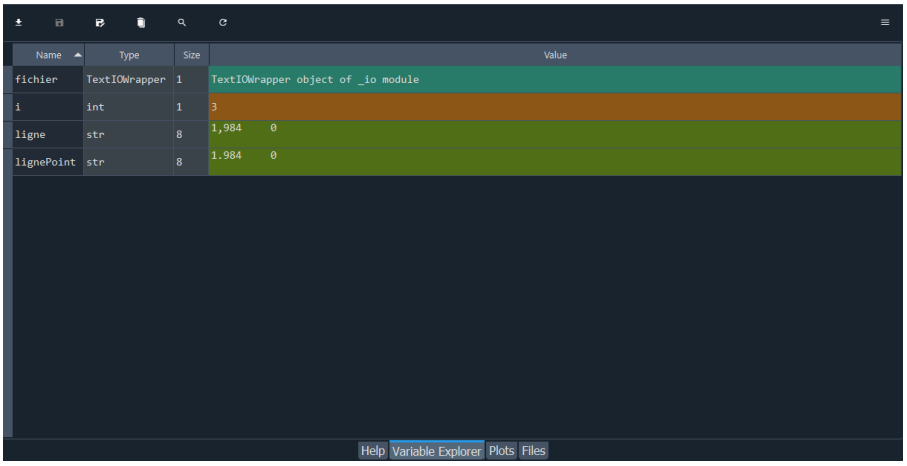

Figure 1 – Contexte des variables dans Spyder

Comme vous le savez, toutes les **informations numériques** sont stockées sous la forme de **0** et de **1**. Un octet est un regroupement de 8 bits, chacun d'eux pouvant prendre soit la valeur 0 soit la valeur 1. En réalité, ces valeurs représentent le niveau de tension à cet endroit précis du circuit électrique : pas de signal → 0 et présence de signal → 1.

Remarques :

- Nous reviendrons sur ces notions de stockage et de codage de l'information en numérique ainsi que des conséquences (arrondis, effets de bord...).
- Nous pourrons parler de la base 2, le binaire, et de la base 16, l'hexadécimal.

3 | Les types de variables en Python

Pour définir une variable, en Python il suffit d'écrire :

nom variable = valeur

Le = est une **affectation** (et non **pas une égalité**), on dit qu'on affecte la valeur à la variable. Si la variable n'était pas définie avant, on parle de **déclaration de la variable ou d'initialisation de la variable**.
En mathématiques, = est une relation d'équivalence ($a = b \iff b = a$). En programmation, l'**affectation** est **unilatérale** avec **toujours à droite la valeur et à gauche la variable** qui reçoit cette valeur. Ainsi, tant qu'une variable n'apparaît pas à gauche d'une =, sa valeur n'est pas modifiée [2].

En Python, une variable est un objet.

Le Tableau 2 résume les types les plus courants en Python.

Nombre			**Booléen**	**Séquences**				**Autres objets**
Entier	Flottant	Complexe		Chaîne de caractères	n-uplet	Liste	Tableau	
`int`	`float`	`complex`	`bool`	`str`	`tuple`	`list`	`array`	Dict, set, file, fonction, ...
42	42.0	1+2j	True/False	"42"	(4,2)	[4,2]	array([4,2])	...

Tableau 2 – Principaux types d'objets en Python

Remarques :

- Pour connaître le type d'un objet, par exemple une variable s'appelant `maVariable`, il suffit d'écrire `type(maVariable)`.
- L'imaginaire noté i en mathématiques est noté j en Python, comme en SI et en physique.
- Le type `array` est importé de la bibliothèque `numpy`, très souvent utilisée en CPGE.

3.1 Nombres et booléen

Le type d'un **nombre** dépend de la valeur qui lui est affectée. Python distingue principalement trois types numériques :

- les **entiers**, de type `int` (*integer*) : `0`, `42`, `-7` ;
- les **nombres à virgule flottante**, de type `float` (*floating point number*) : `3.14`, `-2.0` ;
- les **nombres complexes**, de type `complex` : `1+2j`, `-3j`.

Les **booléens**, de type `bool` (*boolean*), ne peuvent prendre que deux valeurs : `True` (vrai, équivalent à 1) et `False` (faux, équivalent à 0).

Question 5 : Remplir le tableau suivant en renseignant la nature de la valeur affectée ainsi que le type de la variable.

2. Ce sera le cas en CPGE mais c'est une simplification qui ne tient pas compte de notions plus avancées.

Code	Nature de la valeur	Type de la variable
`a = 5.6`	nombre à virgule flottante	float
`b = 2`	entier	int
`c = 2.`	nombre à virgule flottante	float
`d = 2 - 5j`	nombre complexe	complex
`e = True`	booléen	bool
`f = 6j`	nombre complexe	complex
`test = a > 0`	booléen	bool

Remarques :

- Un `float` est une valeur approchée. Ainsi, certains calculs décimaux ne sont pas exacts : il faut éviter de tester l'égalité de deux flottants calculés sans tenir compte d'une tolérance.
- Il est possible de convertir explicitement une valeur à l'aide de `int(...)`, `float(...)`, `complex(...)` ou `bool(...)`. Par exemple, `float(3)` renvoie `3.0` et `int(3.8)` renvoie 3.

3.2 Opérations sur les nombres et booléens

3.2.1 Opérations arithmétiques

Les opérateurs arithmétiques usuels s'appliquent aux entiers et aux flottants. La plupart s'appliquent également aux complexes.

Opération	Opérateur	Exemple
Addition	+	`7 + 3` renvoie 10
Soustraction	-	`7 - 3` renvoie 4
Multiplication	*	`7 * 3` renvoie 21
Division	/	`7 / 3` renvoie 2.3333333333333335 un `float`
Quotient entier	//	`7 // 3` renvoie 2
Reste (modulo)	%	`7 % 3` renvoie 1
Puissance	**	`2 ** 4` renvoie 16

Comme en mathématiques, les **parenthèses** permettent d'imposer l'**ordre des calculs**. Sans parenthèses, Python applique les priorités mathématiques : puissance, puis multiplication et divisions, puis addition et soustraction.

Question 6 : Déterminer le résultat renvoyé par l'instruction suivante :

```
1  12 + 5 * 2**(4-1)
```

Réponse 6 : Les parenthèses indiquent l'ordre de priorité : $4-1=3$ puis la puissnce $2^3=8$, la multiplication $5\times 8=40$ et enfin l'addition $12+40=52$

Remarques :

- Il existe un moyen d'écrire la **notation scientifique** en Python : `6e2` $\Longleftrightarrow$ `6*10**2`.
- Lorsqu'une variable doit être modifiée à partir de sa valeur actuelle, par exemple `a = a + 1`, Python propose une écriture abrégée : `x += 1`. La logique est identique pour `-=`, `*=`, `/=`, `//=`, `%=` et `**=`.

3.2.2 Comparaisons et opérations booléennes

Une comparaison produit toujours un booléen. Les opérateurs à connaître sont :

==	égal à	!=	différent de
<	strictement inférieur à	<=	inférieur ou égal à
>	strictement supérieur à	>=	supérieur ou égal à

Remarques :

- **Attention !** L'égalité se teste avec ==. Le symbole = est réservé à l'affectation d'une valeur à une variable.
- Les nombres complexes peuvent être testés avec == et !=, mais ils ne sont pas ordonnés : les opérateurs <, <=, > et >= ne leur sont pas applicables.

Question 7 : Déterminer les valeurs des variables a, b, c, d et e à la fin du code suivant :

```
1 a = True
2 b = a == False
3 c = b != a
4 a = False
5 d = 12 < 5
6 e = 6 <= 6
```

Réponse 7 :

- `b = a == False`, avec `a = True`. La comparaison devient `True == False`, ce qui n'est pas vrai donc cette comparaison renvoie `False`.
- `c = b != a`, avec `a = True` et `b = False`. Les deux valeurs sont bien différentes, la comparaison renvoie `True`.
- `d = 12 < 5` : 12 n'est pas inférieur à 5 la comparaison renvoie `False`.
- `e = 6 <= 6` : 6 est inférieur ou égal à 6 la comparaison renvoie `True`.
- La dernière fois où la variable a est à gauche d'un = (affectation) est la ligne 4, qui écrase la valeur d'affectation de la ligne 1. Ainsi a a pour valeur `False`.

Les booléens se combinent avec les opérateurs **not** (négation), **and** (et logique) et **or** (ou logique).

a	b	not a	a and b	a or b
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Question 8 : Déterminer la valeur renvoyée par le code suivant :

```
1 a = True
2 b = False
3 not (a and b)
```

Réponse 8 : Les parenthèses indiquent l'ordre de priorité, ainsi `a and b`, soit `True and False`, renvoie `False` et `not False` renvoie `True`.