1 : Introduction Python.

Table des matières

1.1 C'est quoi Python ?

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom *Python* vient d'un hommage à la série télévisée *Monty Python's Flying Circus* dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la version 3. Plus précisément, la version 3.11 a été publiée en octobre 2022. La version 2 de Python est obsolète et n'est plus maintenue, évitez de l'utiliser.

La *Python Software Foundation* est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs.

Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

- Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les miniordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).
- C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions.
- Il est relativement *simple* à prendre en main[1].
- Enfin, il est très utilisé en bioinformatique et plus généralement en analyse de données.

Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, du lycée à l'enseignement supérieur.

1.2 Notations utilisées

Dans cet ouvrage, les commandes, les instructions Python, les résultats et les contenus de fichiers sont indiqués avec `cette police` pour les éléments ponctuels ou

```
sous cette forme, sur plusieurs
lignes, pour les éléments les
plus longs.
```

Pour ces derniers, le numéro à gauche indique le numéro de la ligne et sera utilisé pour faire référence à une instruction particulière. Ce numéro n'est bien sûr là qu'à titre indicatif.

Par ailleurs, dans le cas de programmes, de contenus de fichiers ou de résultats trop longs pour être inclus dans leur intégralité, la notation `[...]` indique une coupure arbitraire de plusieurs caractères ou lignes.

1.3 Introduction au shell

Un *shell* est un interpréteur de commandes interactif permettant d'interagir avec l'ordinateur. On utilisera le *shell* pour lancer l'interpréteur Python.

Pour approfondir la notion de *shell*, vous pouvez consulter les pages Wikipedia :

- du *shell* Unix fonctionnant sous Mac OS X et Linux ;
- du *shell* PowerShell fonctionnant sous Windows.

Un *shell* possède toujours une invite de commande, c'est-à-dire un message qui s'affiche avant l'endroit où on entre des commandes. Dans tout cet ouvrage, cette invite est représentée systématiquement par le symbole dollar $, et ce quel que soit le système d'exploitation.

Par exemple, si on vous demande de lancer l'instruction suivante :

```
$ python
```

Il faudra taper seulement python sans le $ ni l'espace après le $.

1.4 Premier contact avec Python

Python est un langage interprété, c'est-à-dire que chaque ligne de code est lue puis interprétée afin d'être exécutée par l'ordinateur. Pour vous en rendre compte, ouvrez un *shell* puis lancez la commande :

```
python
```

La commande précédente va lancer l'**interpréteur Python**. Vous devriez obtenir quelque chose de ce style pour Windows :

```
PS C:\Users\pierre> python Python 3.7.1 (default,
Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
[...]
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

pour Mac OS X :

iMac-de-pierre:Downloads$ python Python 3.7.1 (default, Dec 14 2018, 19:28:38) [Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>

ou pour Linux :

pierre@jeera:~$ python Python 3.7.1 (default, Dec 14 2018, 19:28:38) [GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information. >>>

Les blocs

- PS C:\Users\pierre> pour Windows,
- iMac-de-pierre:Downloads$ pour Mac OS X,
- pierre@jeera:~$ pour Linux.

Représentent l'invite de commande de votre *shell*. Par la suite, cette invite de commande sera représentée simplement par le caractère $, que vous soyez sous Windows, Mac OS X ou Linux.

Le triple chevron >>> est l'invite de commande (*prompt* en anglais) de l'interpréteur Python. Ici, Python attend une commande que vous devez saisir au clavier. Tapez par exemple l'instruction :

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

Puis validez cette commande en appuyant sur la touche *Entrée*.

Python a exécuté la commande directement et a affiché le texte Hello world!. Il attend ensuite une nouvelle instruction en affichant l'invite de l'interpréteur Python (>>>). En résumé, voici ce qui a dû apparaître sur votre écran :

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

Vous pouvez refaire un nouvel essai en vous servant cette fois de l'interpréteur comme d'une calculatrice :

```
>>> 1+1
2
>>> 6*3 18
```

À ce stade, vous pouvez entrer une autre commande ou bien quitter l'interpréteur Python, soit en tapant la commande exit() puis en validant en appuyant sur la touche *Entrée*, soit en pressant simultanément les touches *Ctrl* et *D* sous Linux et Mac OS X ou *Ctrl* et *Z* puis *Entrée* sous Windows.

En résumant, l'interpréteur fonctionne sur le modèle :

```
>>> instruction python résultat
```

où le triple chevron correspond à l'entrée (*input*) que l'utilisateur tape au clavier, et l'absence de chevron en début de ligne correspond à la sortie (*output*) générée par Python. Une exception se présente toutefois : lorsqu'on a une longue ligne de code, on peut la couper en deux avec le caractère \ (*backslash*) pour des raisons de lisibilité :

```
>>> Voici une longue ligne de code \ ... décrite sur
deux lignes résultat
```

En ligne 1 on a rentré la première partie de la ligne de code. On termine par un \, ainsi Python sait que la ligne de code n'est pas finie. L'interpréteur nous l'indique avec les .... En ligne 2, on rentre la fin de la ligne de code puis on appuie sur Entrée. A ce moment, Python nous génère le résultat. Si la ligne de code est vraiment très longue, il est même possible de la découper en trois voire plus :

```
>>> Voici une ligne de code qui \ ... est vraiment très longue car \ ... elle est découpée sur trois lignes
résultat
```

L'interpréteur Python est donc un système interactif dans lequel vous pouvez entrer des commandes, que Python exécutera sous vos yeux (au moment où vous validerez la commande en appuyant sur la touche *Entrée*).

Il existe de nombreux autres langages interprétés comme Perl ou R. Le gros avantage de ce type de langage est qu'on peut immédiatement tester une commande à l'aide de l'interpréteur, ce qui est très utile pour débugger (c'est-àdire trouver et corriger les éventuelles erreurs d'un programme). Gardez bien en mémoire cette propriété de Python qui pourra parfois vous faire gagner un temps précieux !

1.5 Premier programme

Bien sûr, l'interpréteur présente vite des limites dès lors que l'on veut exécuter une suite d'instructions plus complexe. Comme tout langage informatique, on peut enregistrer ces instructions dans un fichier, que l'on appelle communément un script (ou programme) Python.

Pour reprendre l'exemple précédent, ouvrez un éditeur de texte (pour choisir et configurer un éditeur de texte, reportez-vous si nécessaire à la rubrique *Installation de Python* en ligne) et entrez le code suivant :

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

Ensuite, enregistrez votre fichier sous le nom test.py, puis quittez l'éditeur de texte.

Remarque

L'extension de fichier standard des scripts Python est .py.

Pour exécuter votre script, ouvrez un *shell* et entrez la commande : python test.py

Vous devriez obtenir un résultat similaire à ceci :

```
$ python test.py Hello
world!
```

Si c'est bien le cas, bravo ! Vous avez exécuté votre premier programme Python.

1.6 Commentaires

Dans un script, tout ce qui suit le caractère # est ignoré par Python jusqu'à la fin de la ligne et est considéré comme un commentaire.

Les commentaires doivent expliquer votre code dans un langage humain. L'utilisation des commentaires est rediscutée dans le chapitre 15 *Bonnes pratiques en programmation Python*.

Voici un exemple :

```
# Votre premier commentaire en Python. print("Hello world!")

# D'autres commandes plus utiles pourraient suivre.
```

Remarque

On appelle souvent à tort le caractère # « dièse ». On devrait plutôt parler de « croisillon ».

1.7 Notion de bloc d'instructions et d'indentation

En programmation, il est courant de répéter un certain nombre de choses (avec les boucles, voir le chapitre 5 *Boucles et comparaisons*) ou d'exécuter plusieurs instructions si une condition est vraie (avec les tests, voir le chapitre 6 *Tests*).

Par exemple, imaginons que nous souhaitions afficher chacune des bases d'une séquence d'ADN, les compter puis afficher le nombre total de bases à la fin. Nous pourrions utiliser l'algorithme présenté en pseudo-code dans la figure 1.

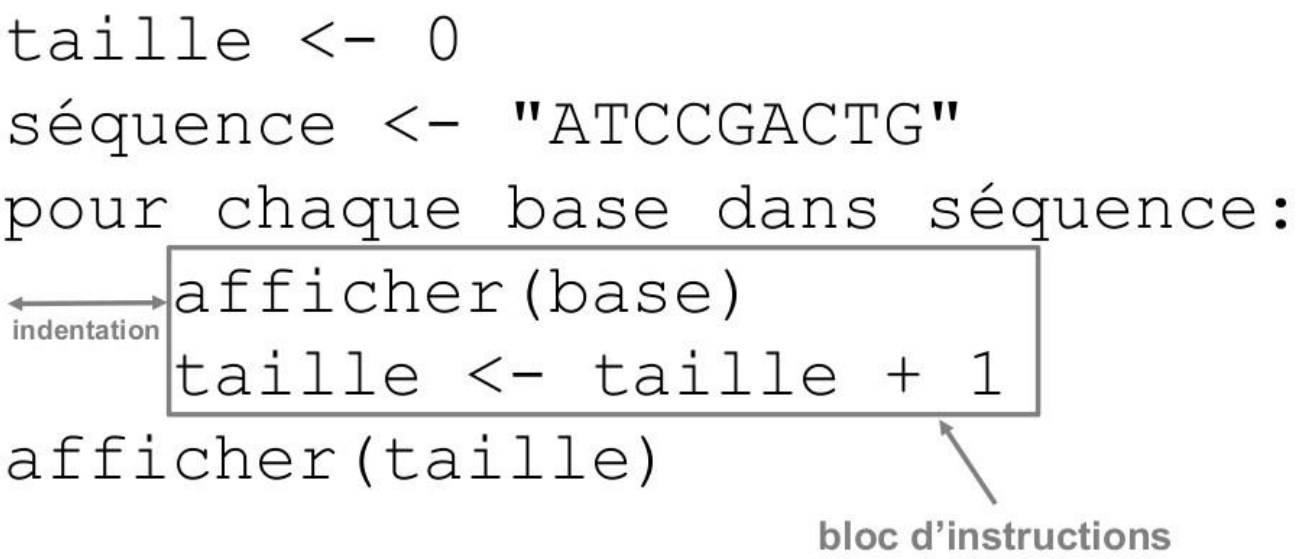

Figure 1. Notion d'indentation et de bloc d'instructions.

Pour chaque base de la séquence ATCCGACTG, nous souhaitons effectuer deux actions : d'abord afficher la base puis compter une base de plus. Pour indiquer cela, on décalera vers la droite ces deux instructions par rapport à la ligne précédente (pour chaque base [...]). Ce décalage est appelé **indentation**, et l'ensemble des lignes indentées constitue un **bloc d'instructions**.

Une fois qu'on aura réalisé ces deux actions sur chaque base, on pourra passer à la suite, c'est-à-dire afficher la taille de la séquence. Pour bien préciser que cet affichage se fait à la fin, donc une fois l'affichage puis le comptage de chaque base terminée, la ligne correspondante n'est pas indentée (c'est-à-dire qu'elle n'est pas décalée vers la droite).

Pratiquement, l'indentation en Python doit être homogène (soit des espaces, soit des tabulations, mais pas un mélange des deux). Une indentation avec 4 espaces est le style d'indentation recommandé (voir le chapitre 15 *Bonnes pratiques en programmation Python*).

Si tout cela semble un peu complexe, ne vous inquiétez pas. Vous allez comprendre tous ces détails chapitre après chapitre.