Les Bases de Python

LUCAS BATTAGLIA

TABLE DES MATIÈRES

Première partie

Introduction

CHAPITRE 1

L'HISTOIRE DE PYTHON

CHAPITRE 2

INSTALLATION D'IDLE ET DE PYCHARM

Afin d'exécuter les programmes écrit en Python, nous avons besoin d'un environnement de développement et d'apprentissage intégré de Python qui compilera le fichier texte que nous lui fournissons et qui exécutera le fichier compiler. Pour cela, je conseille 2 IDE.

IDLE

IDLE qui est l'éditeur proposer de base sur le site Officiel de Python. Il est simple mais marche tres bien pour nos premiers projets.

Vous pourrez donc installer IDLE directement sur site officiel (https://www.python.org/downloads/). Vous aurez le choix d'installer la dernière version ou de choisir une version précédente. Dans cette ouvrage nous utiliserons la Version 3.9 d'IDLE. Un fois le programme d'installation télécharger, exécuter le, le nom devrais ressembler a cela : python-3.9.7-amd64.exe .

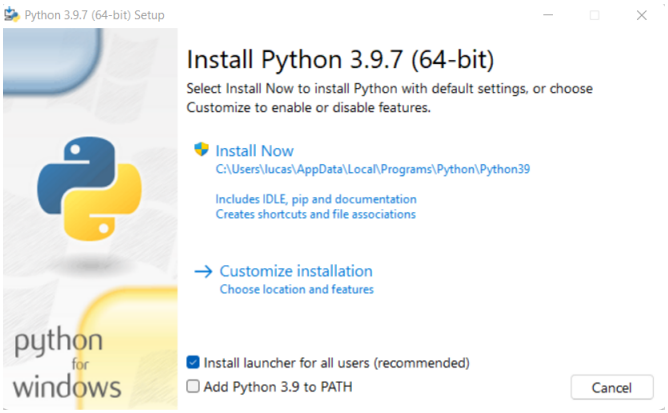

FIGURE 2.1 – Capture d'écran lors de l'ouverture du programme d'installation

Votre premier réflexe va être de vouloir ouvrir IDLE mais plusieurs "fichier" on était installer. Vous pourriez être tenté d'ouvrir celui qui s'appelle Python 3.9 (selon la version installée). Or c'es ce qu'on appelle un terminale. Il est possible de codé dans un terminale mais ce n'est pas optimale. Nous préférerons utiliser l'éditeur IDLE en ouvrant le fichier nommée IDLE(Python 3.9.7). Pour écrire un programme, aller dans le menu "File" puis "New File".

FIGURE 2.2 – IDLE a gauche et le Terminale Python a droite

Pycharm

Pycharm est l'IDE de JetBrain. Personnellement je préféré Pycharm, il permets de faire de plus gros projets dans différents fichiers et de se retrouver plus facilement. En plus de cela, il nous aide a respecter la PEP8 qui est une norme visant a ce que tout les programmeur Python programmes de la même façon et puissent, par conséquent, relire le programme des autres. Nous reviendrons la dessus.

JetBrain est un outil très utile pour les développeurs de différents langages. Il permets d'avoir des IDE pour de nombreux langages de programmations. Celui pour Python s'appelle Pycharm et est téléchargeable au lien https://www.jetbrains.com/pycharm/download. Si vous êtes étudiant, vous pouvez avoir accès au compte professionnel durant vos années d'études. Exécuter le fichier qui porte le nom pycharm-community-2021.3.2.exe (selon la version installer) et cliquer sur suivant jusqu'à l'installation. Une fois terminer, ouvrez le logiciel et accepter les conditions d'utilisation. Pour crée un programme, commencer par faire un "New Project" puis entrer le nom de votre projets. Créé un fichier en faisant un clique droit sur test en haut a gauche de la fenêtre. Sélectionner "new" puis "Python file". Nous écrirons les ici.

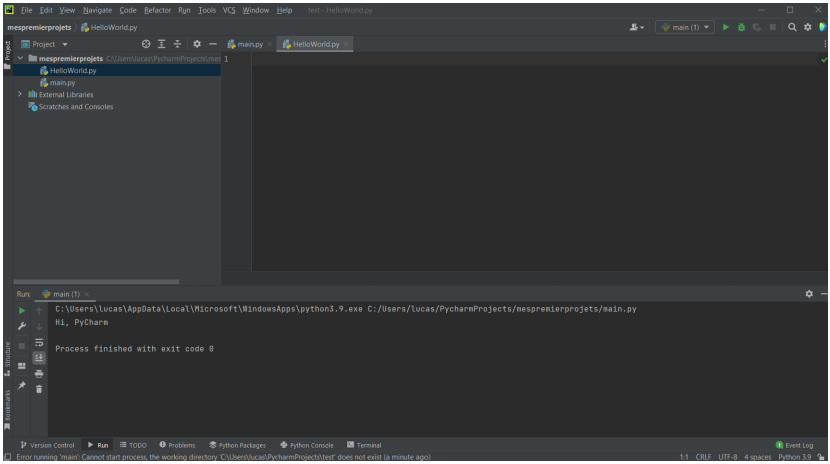

FIGURE 2.3 – Capture d'écran de l'interface de Pycharm

CHAPITRE 3

QUELQUES RÈGLE DE LA PEP8

Comme dit plus tôt la PEP 8 pose quelque regles qu'il est bien de suivre afin de faire un programme "propres" qui puissent etre relus par d'autres développeur facilement. Il n'est pas obligatoire de la suivre mais je vous conseille de prendre cette habitude, elle vous rendra de grands services quand vous aurez besoin d'aller sur les forums, de partager vos codes ou même lorsque vous reprendrez, après un moments, la programmation d'un projets. Vous pouvez trouver toutes les règles de la PEP 8 sur le site officiel de Python ou directement au lien https://www.python.org/dev/peps/pep-0008/. Nous allons tous de même en citer quelques unes. Vous ne les comprendrais peu être pas toutes des ce début de livres mais je vous inviterais a y revenir régulièrement.

1. L'indentation : c'est la seule règle obligatoire de la PEP 8. Python n'utilise pas de commande pour la fin de ses instructions. Il est donc très important de respecter l'indentation (4 espaces) afin que le compilateur sache ce qui se trouve dans la boucle ou dans l'instruction.
2. Importation : afin de savoir d'où viennent les fonctions que nous utiliserons la PEP8 conseille d'utiliser l'importation de module, de fichier et de bibliothèques avec la commande import *module* plutôt que la version scolaire from *module* import*. Après avoir importer les modules avec la première commande nous devrons appeler des fonction *module.fonction*(), on sera donc de quelle module provient

la fonction appeler. Au contraire, pour utiliser une fonction importer, avec le version scolaire, nous utiliserons la commande *fonction*(), nous ne connaissons pas module ou se trouve notre fonction. Nous devrions aussi importer les modules ligne par ligne avec les modules interne a Python en premier puis les module externe et le tout trier par ordre alphabétique. Si un nom de module es trop long on peut le raccourcir en utilisant le mots clé *as*

3. Nommage : pour savoir ce qu'est chaque objet Python, la PEP 8 défini quelque règle :

   - Le nom des variables, module et fonction sont écrit entièrement en minuscule et les mots la contenant sont séparer par des underscore. Exemple : nom_dune_variable (style : snake_case)
   - Les constante doivent être écrit en majuscule séparer avec un underscore. Exemple : NOM_DE_LA_CONSTANTE
   - Les noms de classes et d'exceptions seront ecrits avec une lettre majuscule, seulement au debut de chaque mots. Exemple : NomDuneClasse (style : CamelCase)

   Il faut également donner un nom qui a un sens a nos objets. On autorisera seulement les noms a 1 seules caractère pour les compteur de boucle et compteur d'indice ou pour des coordonnée cartésienne

4. Les espaces : Pour les opérateur (+, -, /, *, ==, !=, <=, not, in, and, or, . . .) on mets un espace avant est après. On ne mets pas d'espace juste après une parenthèse ouvrante ou juste avant une parenthèse fermante. Pareil pour les accolades et les. Les caractère , et : n'ont pas d'espace avant mais 1 après, sauf a l'intérieur des crochets ou on ne mettra aucun espace. On ne rajoute pas d'espace non plus pour faire jolis.

5. retour a la ligne : Il arrivera parfois d'avoir a écrire une ligne de code très longues. Or la PEP 8 demande a ne pas avoir de ligne execedent la taille d'un écran, sur Pycharm cette limite est fixe a 120 caractère. Nous avons donc besoin de faire des retour a la ligne.

   - Dans une suite de commande on utilisera le antislash pour faire le retour a la ligne
   - A l'intérieur de parenthèse, de crochet ou d'accolade nous pouvons le faire après une virgule ou un opérateur séparent

2 élément. Par exemple : fonction(element1, element2, element3, element4). Dans lorsqu'on veux couper une chaine de caractère on ferme les guillemets on fait un retour la la ligne puis on rouvre les guillemets. Exemple : print("Je suis en train d'écrire "
"un très grand roman de "
"plusieurs milliard de page")

- A l'intérieur des triples guillemets (utiliser pour les docstring) on peu faire des retour a la ligne directement.

6. Ligne vide : Afin de rendre le programme plus lisible, Nick Coghlan, Barry Warsaw et Guido van Rossum nous recommande de laisser des lignes vides entre chaque partie de programme. De plus, il est recommander de laisser 2 lignes vides avant les définitions de classes et de fonctions.
7. Commentaire : Ils nous conseille également de mettre des commentaire afin de détailler des ligne de notre programmes. Pour crée un commentaire, nous commençons par mettre un # puis nous insérons notre commentaire. Il est également déconseiller de mettre un commentaire en fin de ligne de commande, mais en début de ligne.
8. Docstring : Les docstrings sont la pour l'utilisateur du programmes, lorsqu'il appellera la commande help(*fonction*) il obtiendra l'aide (la docstring) que vous aurai écrite. Elle doivent commencer et terminer par des triple guillemets et doivent comprendre une explication de votre fonction aussi que le nom des paramètre avec leur type est leur rôle dans la fonction. Elle doit également contenir les effet de bort et les retours.
9. Organisation du code : Pour faire le meilleur programmes, voici les différent éléments d'un programme dans l'ordre :
    - On commence avec un Docstring décrivant le programme
    - Les credits sous cette forme
      _ _ authors _ _ = ("Lucas Battaglia", "Deuxieme Auteur")
      _ _ contact _ _ = ("developement.python@gmail.com")
      _ _ version _ _ = '1.0.0'
      _ _ copyright _ _ = "copyright"
      _ _ date _ _ = "2022/03"

    - Les importation

- Les constante
- Les définition de classe
- Les définition de fonction
- La ligne : *if _ _ name _ _ == "_ _ main _ _" :* Elle correspond au début du programme principale, c'est le point d'entrée
- Les instruction, appel de fonction, de classe, etc.

Il faut savoir qu'il existe de nombreuse PEP d'on on s'est également inspirer pour écrire cette partie, notamment la PEP 257.

Deuxième partie

Nos début en Python

CHAPITRE 1

LES TYPES DE DONNÉES

Les chaines de caractères

Les nombre entier et flottant

Les Expressions booléens

Variables, Expressions, Instructions

CHAPITRE 2

LES BOUCLES ET INSTRUCTIONS

Instruction conditionnelle "If"

La boucle conditionnelle "While"

La boucle inconditionnelle "For"

CHAPITRE 3

NOM A TROUVER

Entrées / Sorties

Les fonctions

CHAPITRE 4

LE STOCKAGE DE DONNÉE SIMPLE

Stockage a courts termes

Les Listes

Les dictionnaires

Stockage a longs termes

Les fichiers

Troisième partie

Les bibliothèques

CHAPITRE 1

NOS PREMIÈRES BIBLIOTHÈQUES

La bibliothèque math

La bibliothèque random

CHAPITRE 2

LES BIBLIOTHÈQUES GRAPHIQUES

La bibliothèque Turtle

La bibliothèque Tkinter

La bibliothèque Pygame

CHAPITRE 3

LES BIBLIOTHÈQUES SONORE