Chapitre 1 : Rappels sur la programmation en Python

Dr. Ishak ABDI — ishak.abdi@univ-jijel.dz
Département de Génie civil et hydraulique — Université de Jijel

Concepts de base en informatique et outils numériques 💻

Objectif

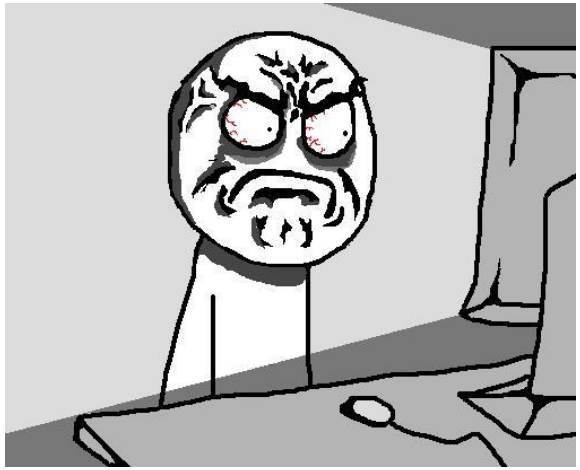
> Ce chapitre consolide les bases en **programmation Python** et en **environnement informatique**. Il traite des **systèmes d'exploitation**, des **réseaux**, de l'**installation de Python**, ainsi que des notions clés de **programmation** (variables, conditions, boucles, fonctions, modules). Il introduit enfin la **manipulation des structures de données** et l'usage de **bibliothèques scientifiques** comme *NumPy* et *Pandas*.

1. Introduction à la programmation et Python 

Pourquoi apprendre à programmer ?

- L’informatique est omniprésente dans la société contemporaine.
- Les programmes interviennent dans : communication, sciences, santé, divertissement, tâches quotidiennes.
- Comprendre et concevoir des programmes ⟶ compétence essentielle.
- La programmation ouvre des opportunités professionnelles et permet de créer des outils innovants.

Qu’est-ce qu’un programme ? 🖥️



- Un programme ➞ une suite d’instructions exécutées séquentiellement.
- Chaque instruction décrit une action précise pour l’ordinateur.
- Programmer ➞ formuler des instructions claires, logiques et structurées.
- La programmation repose sur un raisonnement algorithmique.

Les langages de programmation

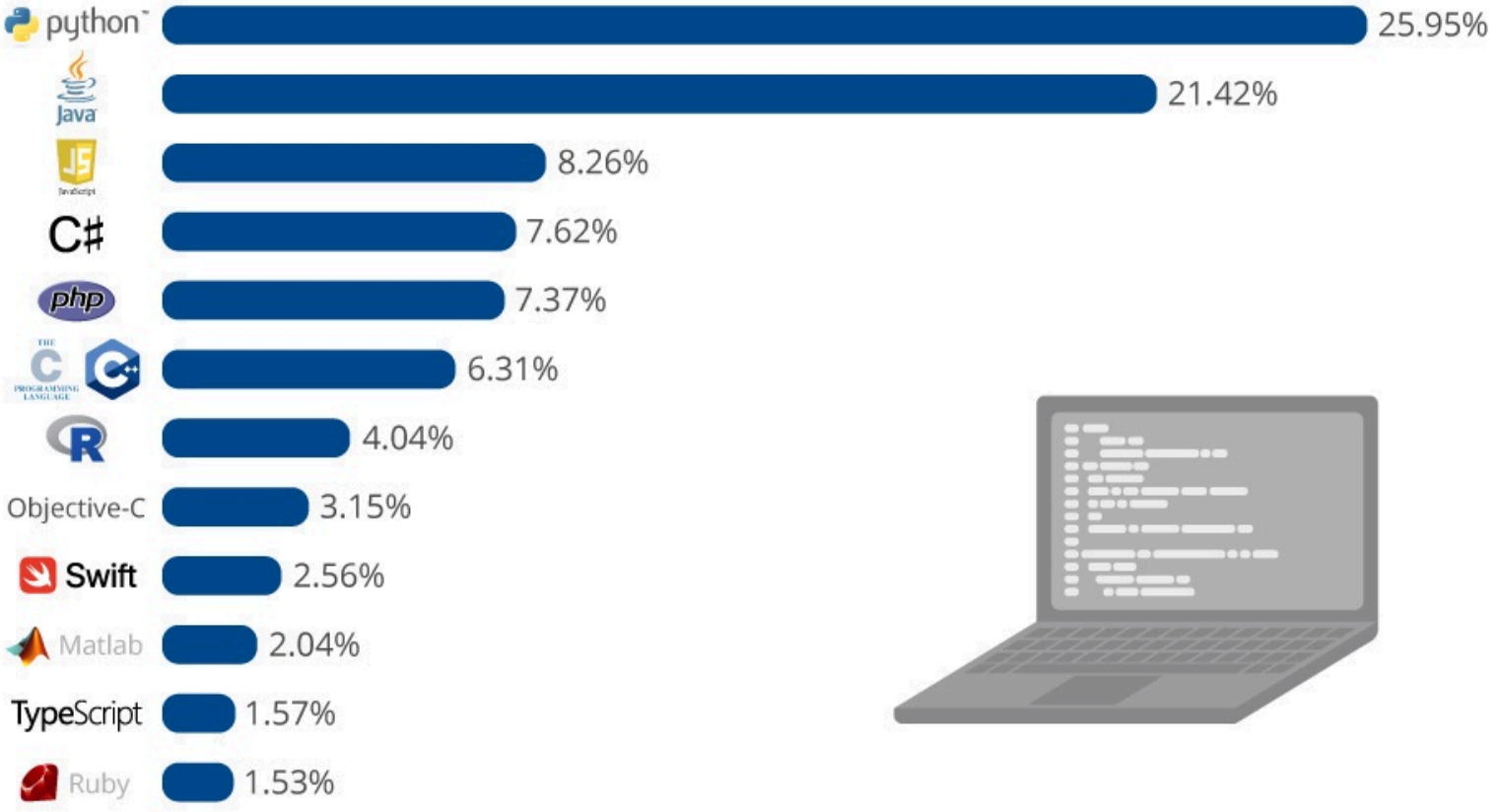
- Il existe de très nombreux langages de programmation.
- Le choix du premier langage peut sembler difficile pour les débutants.
- Chaque langage possède sa syntaxe, son usage et ses domaines d'application.

Pourquoi choisir Python ? 

- Utilisé par plus d’un million de programmeurs dans le monde.
- Langage apprécié pour sa simplicité, sa lisibilité et sa polyvalence.
- Dispose d’un vaste écosystème de bibliothèques (science, data, web…).
- Très présent dans l’enseignement, la recherche scientifique et l’industrie.

The Most Popular Programming Languages

Share of the most popular programming languages in the world*



* Based on the PYPL-Index, an analysis of Google search trends for programming language tutorials.

@StatistaCharts Source: PYPL

statista

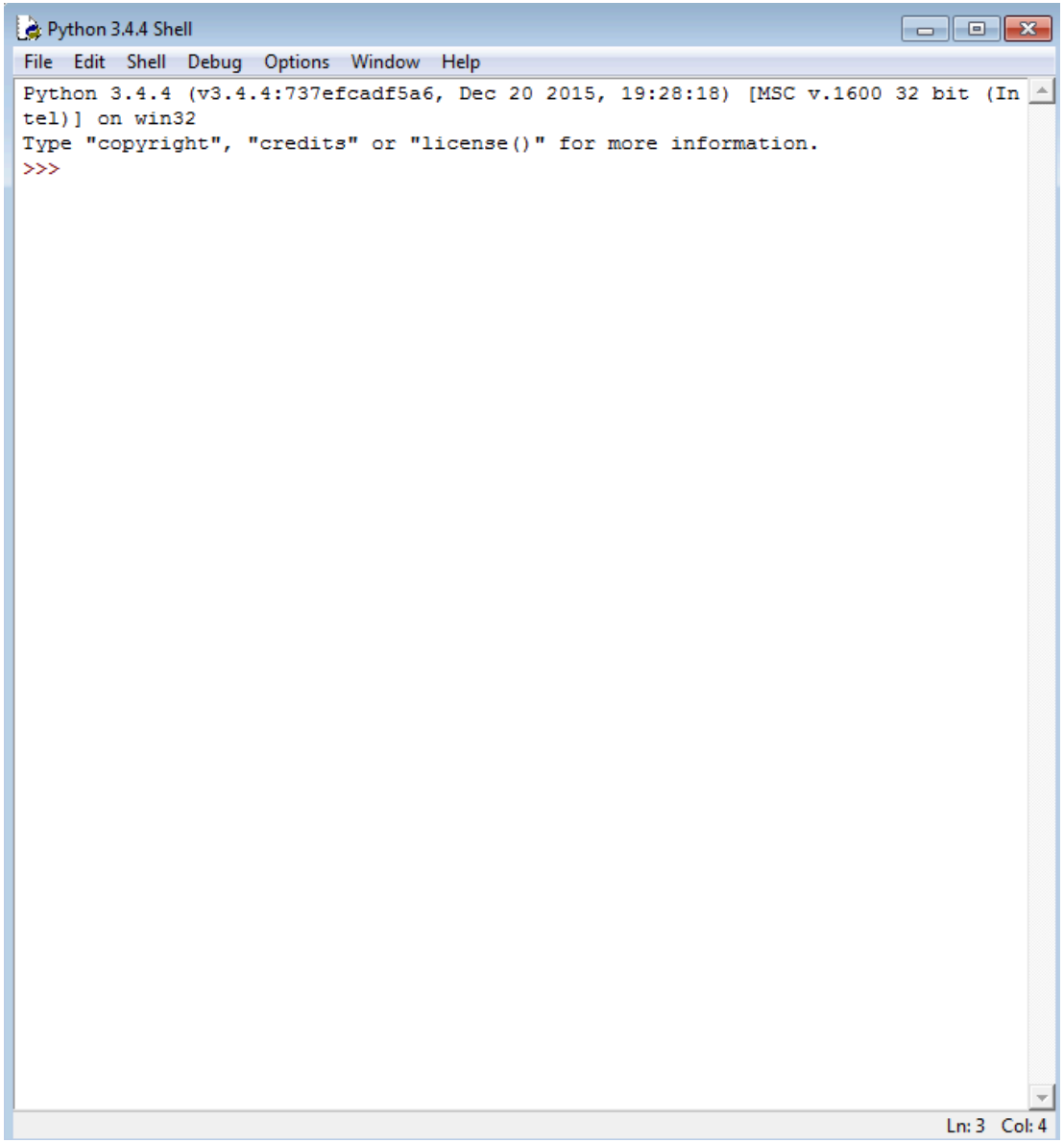
Qu’est-ce que Python ?

- Créé en 1989 par Guido van Rossum aux Pays-Bas.
- Nom inspiré de la série humoristique Monty Python’s Flying Circus.
- Première version publique en 1991.
- Version actuelle : Python 3.11 (octobre 2022), la version 2 est obsolète.
- Développement coordonné par la *Python Software Foundation* (organisation à but non lucratif).

Installation et noyau de Python (Shell)

- Téléchargement depuis le site officiel : python.org
- Le package de base comprend :
  - L'interpréteur Python,
  - Un éditeur simple ou IDE IDLE pour coder.

Ce package représente le noyau de Python : suffisant pour créer des programmes simples.


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 4

Langages interprétés vs. Compilés

Langages compilés

- Code traduit en instructions machine avant création d'un exécutable.
- Exécution rapide après compilation.
- Exemples : **C**, **C++**
- Conviennent aux applications performantes, moins flexibles.

Langages interprétés

- Code exécuté directement par un interpréteur.
- Analyse et exécution à chaque lancement.
- Exemples : **Python**, **PHP**
- Flexibles, simples, idéaux pour prototypes et simulations.

Atouts majeurs de Python

- **Un langage conçu pour la clarté**
  - Syntaxe proche du langage naturel
  - Structure simple
  - Facilite la compréhension des bases
- **Un outil efficace pour apprendre**
  - Exécution immédiate via l’interpréteur
  - Idéal pour tester des idées rapidement
  - Très utilisé en enseignement
- **Une communauté mondiale active**
  - Documentation riche
  - Nombreux tutoriels et forums
  - Soutien de la Python Software Foundation

Un langage polyvalent et moderne

- **Présent dans de nombreux domaines**
  - Analyse de données et intelligence artificielle
  - Développement web et interfaces graphiques
  - Automatisation, scripts système, simulations scientifiques
- **Un écosystème en constante évolution**
  - Mise à jour régulière du langage
  - Nouvelles bibliothèques et outils chaque année
  - Forte adoption dans l'industrie et le milieu académique
- **Un langage qui accompagne l'évolution du numérique**
  - Python s'adapte aux besoins émergents
  - Convient aux projets expérimentaux comme aux solutions industrielles
  - Un choix sûr pour un parcours informatique moderne

Limites de Python

- En raison de son interprétation, Python est généralement moins rapide que les langages compilés comme C ou C++.
- Cette limite reste toutefois peu impactante, car de nombreuses opérations intensives sont exécutées via des modules compilés.
- Les gains de productivité offerts par Python compensent largement cette différence de performance.
- Pour les applications nécessitant une exécution rapide, des bibliothèques optimisées comme **NumPy** permettent d'atteindre des performances proches du C.

Utilisateurs actuels de Python

Packages en Python

- Python inclut peu de fonctionnalités dans son noyau → il repose sur l'ajout de **packages** selon les besoins.
- Cette approche nécessite d'installer les bibliothèques avant de les importer, ce qui est courant dans l'open source.
- Un très vaste écosystème existe : calcul scientifique, web, interfaces graphiques, bases de données, etc.
- Les packages peuvent être installés individuellement ou via des distributions complètes comme **Anaconda**, qui regroupe des outils proches de MATLAB.

Bibliothèques Python pour le Calcul Scientifique (1/2)

- **NumPy**
  - Base du calcul scientifique en Python
  - Tableaux multidimensionnels (ndarray)
  - Fonctions mathématiques optimisées
- **SciPy**
  - Construite sur NumPy
  - Outils pour optimisation, algèbre linéaire, intégration, interpolation
  - FFT, traitement du signal et de l’image
  - Résolution d’équations différentielles

Bibliothèques Python pour le Calcul Scientifique (2/2)

- **Matplotlib**
  - Bibliothèque de référence pour visualisations 2D
  - Graphiques personnalisables
- **Pandas**
  - Manipulation et analyse de données performante
  - Structures DataFrame et Series
  - Nettoyage, transformation et traitement efficace des données

La Distribution Anaconda

- **Distribution complète** regroupant :
  - L'interpréteur Python
  - Un large ensemble de paquets spécialisés
  - L'éditeur **Spyder** et **Jupyter Notebook**
- Modules inclus pour le **calcul scientifique** et la **science des données**
- **Versions disponibles** :
  - Gratuite : **Anaconda Distribution**
  - Payante : **Enterprise**, pour usage professionnel
- **Compatibilité** : Windows, macOS, Linux
- **Site officiel** : https://www.anaconda.com

Éditeurs Python

- **Éditeur fourni avec Python** : Python IDLE
- **Éditeurs avancés** :
  - Visual Studio Code
  - Spyder
  - Visual Studio
  - PyCharm
  - Wing Python IDE
  - Jupyter Notebook

Choix de l'éditeur dépend de :

- L'expérience préalable avec des outils de développement
- Les compétences en programmation
- La nature des projets Python à réaliser

2. Fonctionnement de l'exécution des programmes Python

2.1 Exécution d'un programme python

- **Point de vue du programmeur :**
  - Code écrit dans un fichier texte `.py`
  - Exemple : `script0.py`

```
print('hello world')
print(2 ** 100)
```

- Sortie :

```
hello world
1267650600228229401496703205376
```

Exemple d'exécution Python

- Exécution depuis une **Invite de commandes Windows** :

```
C:\temp> python script0.py
hello world
126765060022822940149670320537 6
```

- Le script affiche une chaîne de caractères et un nombre.
- Objectif : illustrer les **principes fondamentaux de l'exécution** d'un programme Python.

Point de vue de Python

- Python ne se limite pas à exécuter du code texte via l’interpréteur.
- En arrière-plan, plusieurs étapes se déroulent avant que le code ne s’exécute.
- Connaître la structure d’exécution aide à mieux comprendre le fonctionnement global.

Compilation en bytecode

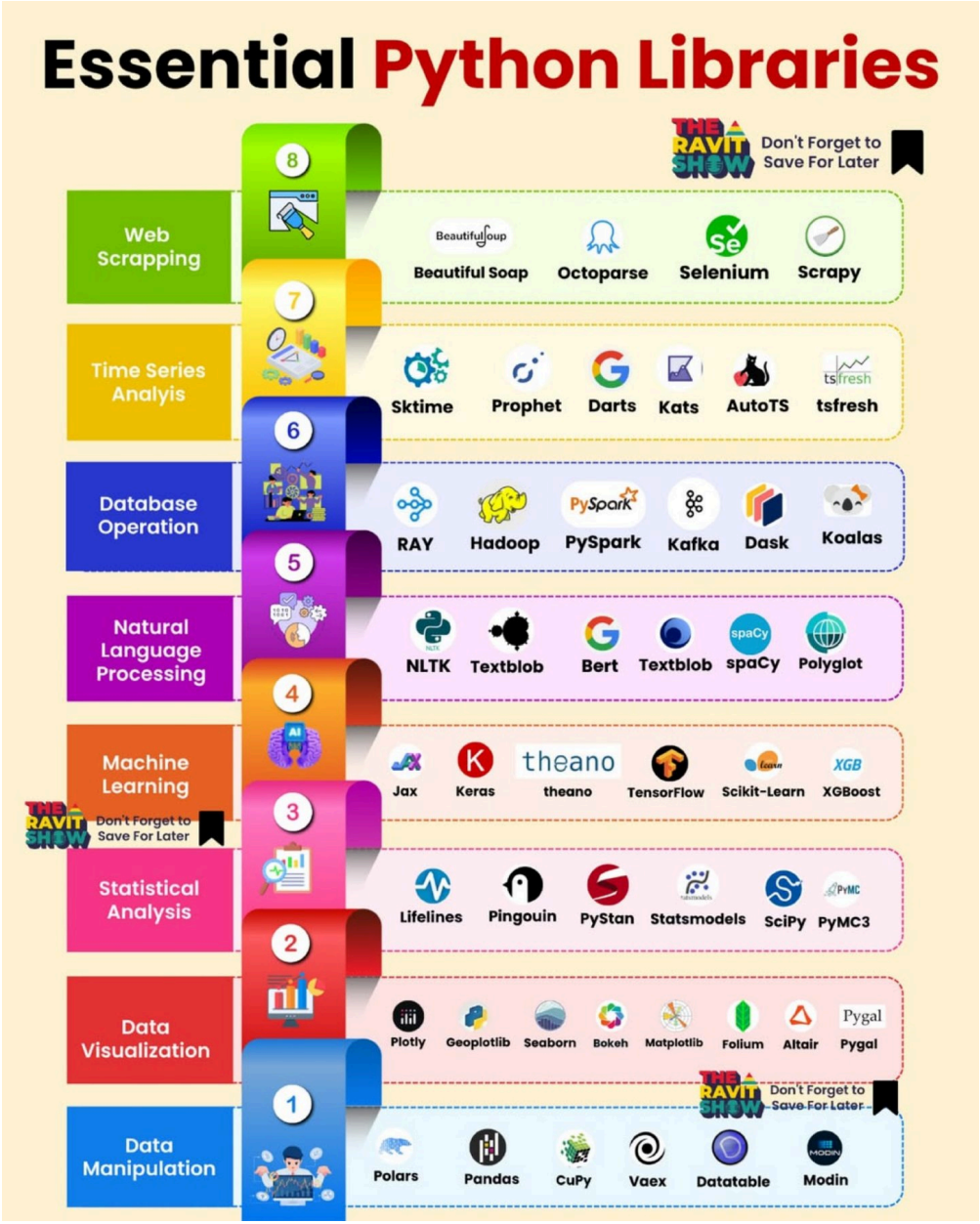
- Le code source est compilé en **bytecode**, format bas niveau et indépendant de la plateforme.
- Le bytecode s'exécute **plus rapidement** que le code source.
- Stockage : fichiers `.pyc` pour éviter la recompilation si le code n'a pas changé.
- Si `.pyc` impossible : bytecode généré en mémoire et supprimé à la fin.
- Avantages : accélère le démarrage et permet de distribuer des programmes sans source.

La machine virtuelle Python (PVM)

- La PVM exécute le **bytecode**, instruction par instruction.
- Elle constitue le **moteur d'exécution** de Python.
- Techniquement, la PVM est la dernière étape de l'**interpréteur Python**.
- Pas de programme séparé, toujours intégrée dans Python.

Implémentations de Python

- **CPython** : implémentation officielle, en C, utilise la PVM, compatible avec la plupart des bibliothèques.
- **Jython** : implémentation en Java, bytecode exécuté sur la JVM, intégration avec bibliothèques Java.
- **IronPython** : implémentation pour .NET, bytecode exécuté sur le CLR, accès aux bibliothèques .NET.


Essential Python Libraries
THE RAVIT SHOW
Don't Forget to Save For Later
8
Web Scrapping
Beautiful Soup
Octoparse
Selenium
Scrapy
7
Time Series Analyis
Sktime
Prophet
Darts
Kats
AutoTS
tsfresh
6
Database Operation
RAY
Hadoop
PySpark
Kafka
Dask
Koalas
5
Natural Language Processing
NLTK
Textblob
Bert
Textblob
spaCy
Polyglot
4
Machine Learning
Jax
Keras
theano
TensorFlow
Scikit-Learn
XGBoost
THE RAVIT SHOW
Don't Forget to Save For Later
3
Statistical Analysis
Lifelines
Pingouin
PyStan
Statsmodels
SciPy
PyMC3
2
Data Visualization
Plotly
Geoplotlib
Seaborn
Bokeh
Matplotlib
Folium
Altair
Pygal
THE RAVIT SHOW
Don't Forget to Save For Later
1
Data Manipulation
Polars
Pandas
CuPy
Vaex
Datatable
Modin