

# Programmation en Python

Polytech Marseille

Séverine Dubuisson, Simon Vilmin

`severine.dubuisson@univ-amu.fr`,

`simon.vilmin@univ-amu.fr`

2024 - 2025



# Contenu du cours

## Objectifs :

- maîtriser les bases de Python
- faire un peu d'algorithmique

## Volume horaire :

- 14 séances de Cours/TD
- 10 séances de TP

## Évaluation :

- un examen terminal

## Ressources :

- supports sur [AMeTICE](#) au fur et à mesure

Contributeur·ices aux supports : Séverine Dubuisson, Simon Vilmin

# Sources

## Autres cours :

- ancien cours de M. Bulot,
- cours d'intro à Python de l'Université de Grenoble ([Caséine](#))

## Livres :

- Mark Lutz, « Learning python : Powerful object-oriented programming », O'Reilly Media Inc, 2013 (5ème édition).
- Luciano Ramalho, « Fluent Python : Clear, concise, and effective programming », O'Reilly Media Inc, 2015.
- Gérard Swinnen, « Apprendre à programmer avec Python 3 », Eyrolles, 2012.
- Anthony Shaw, « CPython Internals : Your Guide to the Python 3 Interpreter », Real Python, 2021.

Sites web : la [doc](#), [Wikipédia](#) et [stack overflow](#) !

# Le langage Python


Le Python en quelques mots :


- créé dans les années 80 par Guido Van Rossum
- dernière version : Python 3.12 (bientôt 3.13)
- langage considéré comme *interprété*
- conçu sur le paradigme *orienté-objet*... Mais avec lequel on fait de l'impératif, de l'orienté-objet, du fonctionnel, etc

Des ressources utiles :

- documentation : [docs.python.org](https://docs.python.org)
- bonnes pratiques : [peps.python.org](https://peps.python.org), [clean code](#)


# Langage « interprété » ?

 **Question** : comment l'ordinateur exécute-t-il un programme ?

 **Réponse** : il faut d'abord le traduire le langage « humain » en langage machine ! La machine peut ensuite effectuer les opérations.

Deux grands moyens de faire cette traduction :

- *Compilation* : le programme est traduit une fois pour toute (compilé) en langage machine exécutable
- *Interprétation* : le programme est traduit au fur et à mesure et à chaque lancement. On peut exécuter des instructions à la volée via l'*interpréteur*

 **Remarque** : Python est un langage *interprété*.

# Pourquoi Python ?

- langage très utilisé et très haut niveau (proche du pseudo-code)
- des applications variées (web, science des données, embarqué, ...)

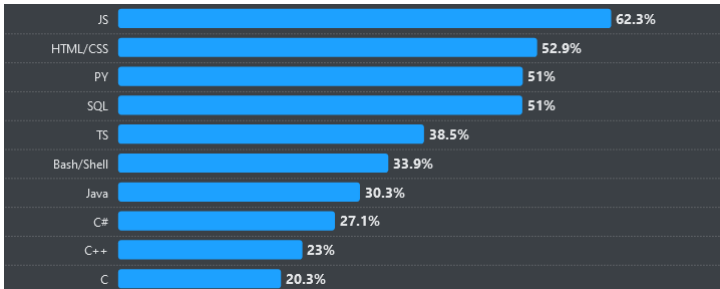


Figure – les 10 langages les plus utilisés sur l'année 2023-2024,  $\simeq$  60k répondant.e.s (source [survey.stackoverflow.co/2024/](https://survey.stackoverflow.co/2024/))

**!** **Attention :** cette popularité dépend aussi du domaine d'application !

# Pourquoi pas Python ?

2022 International Conference on ICT for Sustainability (ICT4S)

## Finding Significant Differences in the Energy Consumption when Comparing Programming Languages and Programs

Science of Computer Programming 205 (2021) 102909

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

### Ranking programming languages by energy efficiency

Rui Pereira<sup>a,b,\*</sup>, Marco d'Ávila<sup>c</sup>, Jácóme Cunha<sup>b</sup>, João P. Araújo<sup>d</sup>

<sup>a</sup> C4 – Centro de Competências em Cloud Computing, Covilhã, Portugal  
<sup>b</sup> HASLab/INESC Tec, Portugal  
<sup>c</sup> Universidade de Alentejo, Portugal  
<sup>d</sup> Departamento de Engenharia Informática

Abstract—The EU is regarding climate change as a major challenge. Reducing the energy consumption of software is one of the ways to do this. We take a software analysis approach to study the energy consumption of different programming languages. We find that the choice of writing your program in a certain programming language can significantly affect the energy consumption of the program.

Software Quality Journal  
<https://doi.org/10.1007/s11219-024-09690-4>

RESEARCH

### Programming languages ranking based on energy measurements

Alberto Gordillo<sup>1</sup> • Coral Calero<sup>1</sup> • M<sup>a</sup> Ángeles Moraga<sup>1</sup> • Félix García<sup>1</sup> • João Paulo Fernandes<sup>2</sup> • Rui Abreu<sup>3</sup> • João Saraiva<sup>4</sup>

Accepted: 28 June 2024  
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

#### Abstract

Software is developed using programming languages whose choice is made based on a wide range of criteria, but it should be noted that the programming language selected can affect the quality of the software product. In this paper, we focus on analysing the differences in energy consumption when running certain algorithms that have been developed using different programming languages. Therefore, we focus on the software quality from the perspective of greenability, in concrete in the aspects related to energy efficiency. For this purpose, this study has conducted an empirical investigation about the most suitable

# Pourquoi pas Python ?


**Table 3** Mean values of energy and time of the 14 program languages SwE vs. HwM


PL	Time SwE (s)	Time HwM (J)	Energy SwE (J)	Energy HwM (J)	Type	Paradigm
C	2.364	1.201	66.643	206.135	Compiled	Imperative
C++	2.286	1.227	67.162	219.207	Compiled	OO, Imperative
Ada	3.501	1.941	96.143	312.421	Compiled	OO, Imperative
Java	3.992	1.904	113.273	337.046	Virtual Machine	OO
Pascal	7.601	4.486	157.002	671.455	Compiled	Imperative
Haskell	8.236	5.293	206.918	684.366	Compiled	Functional
JavaScript	13.367	8.203	192.353	851.364	Interpreted	OO, Scripting
Dart	13.773	8.765	199.232	956.296	Interpreted	OO, Scripting
PHP	78.338	39.583	2380.527	6462.751	Interpreted	OO, Scripting
Erlang	83.407	41.769	2805.190	6868.359	Virtual Machine	Imperative, Functional
JSRuby	113.818	80.307	3527.398	12,492.886	Interpreted	Scripting
Ruby	166.949	105.054	5636.157	17,183.644	Interpreted	Scripting, Functional
Python	143.993	120.475	6286.523	19,067.487	Interpreted	OO, Scripting
Perl	191.956	115.179	6641.842	19,229.640	Interpreted	OO, Imperative, Functional

Figure – Source : Gordillo et al., « Programming languages ranking based on energy measurements », Software Quality Journal, 2024



# Pourquoi pas Python ?

 **Attention** : alerte esprit critique, il faut penser à remettre les chiffres présentés dans le contexte de l'étude et voir quelles en sont les limitations !

 **Astuce** : un choix technologique se fait sur plusieurs critères : efficacité, popularité, consommation, usages, ...

# Comment faire du Python ?

Sur sa machine personnelle :

1. télécharger et installer **Python**
2. écrire ses programmes dans des fichiers `.py` via un *éditeur de texte* ou un *environnement de développement intégré* (IDE)
3. les exécuter avec l'*interpréteur Python*, par exemple depuis Powershell :

```
python3 chemin programme.py
```

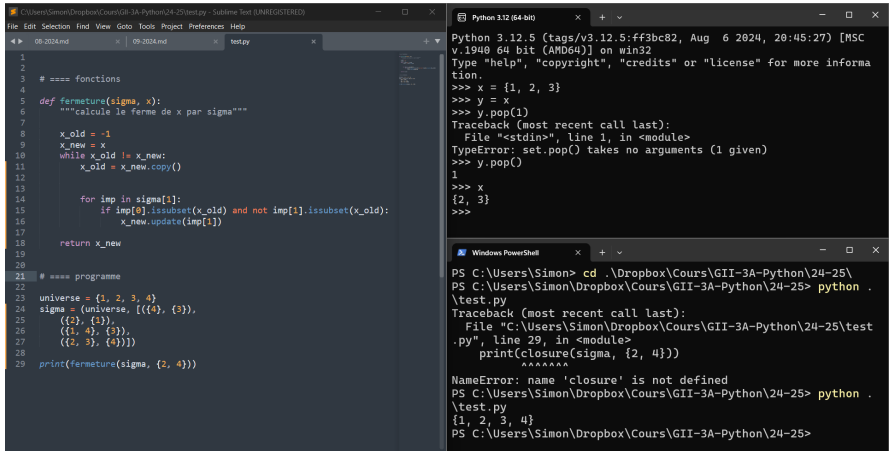
Exemples d'outils logiciels :

- éditeurs de texte : bloc-notes, **Notepad++**, **Sublime Text**, **Atom**, ...
- environnements de développement intégré : **Spyder**, **PyCharm**, **VS Code**, ...



**Remarque** : en Cours/TD, *éditeur de texte* + *interpréteur*

# Comment faire du Python ?



The screenshot displays a Python development environment with three windows:

- Code Editor (Sublime Text):** Shows a Python script named `test.py` with the following code:

```
1
2
3 # ==== fonctions
4
5 def fermeture(sigma, x):
6     """calcul le ferme de x par sigma"""
7
8     x_old = -1
9     x_new = x
10    while x_old != x_new:
11        x_old = x_new.copy()
12
13
14    for imp in sigma[1]:
15        if imp[0].issubset(x_old) and not imp[1].issubset(x_old):
16            x_new.update(imp[1])
17
18    return x_new
19
20
21 # ==== programme
22
23 universe = {1, 2, 3, 4}
24 sigma = (universe, [{4}, {3}],
25          [{2}, {1}],
26          [{1, 4}, {3}],
27          [{2, 3}, {4}]))
28
29 print(fermeture(sigma, {2, 4}))
```
- Python 3.12 (64 bit) Interpreter:** Shows the execution of the script, resulting in a `TypeError: set.pop() takes no arguments (1 given)` at line 1 of the module.

```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug 6 2024, 20:45:27) [MSC
v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informa
tion.
>>> x = {1, 2, 3}
>>> y = x
>>> y.pop(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: set.pop() takes no arguments (1 given)
>>> y.pop()
1
>>> x
{2, 3}
>>>
```
- Windows PowerShell:** Shows the command to run the script, resulting in a `NameError: name 'closure' is not defined` at line 29 of the module.

```
PS C:\Users\Simon> cd .\Dropbox\Cours\GII-3A-Python\24-25\
PS C:\Users\Simon\Dropbox\Cours\GII-3A-Python\24-25> python .
\test.py
Traceback (most recent call last):
  File "C:\Users\Simon\Dropbox\Cours\GII-3A-Python\24-25\test
.py", line 29, in <module>
    print(closure(sigma, {2, 4}))
          ^^^^^^^
NameError: name 'closure' is not defined
PS C:\Users\Simon\Dropbox\Cours\GII-3A-Python\24-25> python .
\test.py
{1, 2, 3, 4}
PS C:\Users\Simon\Dropbox\Cours\GII-3A-Python\24-25>
```

## Comment faire du Python ? (bis)

Sur une plateforme en ligne :

1. créer des *Jupyter notebooks*
2. les exécuter sur une plateforme distante (qui elle dispose de l'interpréteur Python) : [Google Colab](#), [Jupyterlab](#), ...

**i Remarque** : en TP, *jupyter notebooks* + *Jupyterlab*

**i Remarque** : il y a d'autres manière de faire du Python : notebooks en local, programmes en ligne, ...

# Comment faire du Python ? (bis)

The screenshot shows a JupyterLab environment with a file browser on the left and a notebook editor on the right. The file browser shows a directory named 'notebooks' containing several files: 'Intro.ipynb', 'Lorenz.ipynb', 'sqlite.ipynb', and 'TP1.ipynb'. The 'TP1.ipynb' file is selected. The notebook editor displays the title 'Python TP1 - Familiarisation avec l'environnement et premiers programmes' and the author 'Polytech GII - 3A - Année 2024/2025'. The notebook content includes a section 'Partie 1 - Premiers pas' with instructions and code cells. The first code cell contains a print statement: `print(f'C'est mon premier programme de 3A GII')`. The second code cell is empty. The third code cell contains a print statement: `print(f'La valeur de c est {c}')`. The fourth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The fifth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The sixth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The seventh code cell contains a print statement: `print(f'La valeur de c est {c}')`. The eighth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The ninth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The tenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The eleventh code cell contains a print statement: `print(f'La valeur de c est {c}')`. The twelfth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The thirteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The fourteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The fifteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The sixteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The seventeenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The eighteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The nineteenth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The twentieth code cell contains a print statement: `print(f'La valeur de c est {c}')`. The status bar at the bottom indicates 'Simple' mode, 'Python (Pyodide)', and 'Ln 1, Col 17'.

Python TP1 - Familiarisation avec l'environnement et premiers programmes

Polytech GII - 3A - Année 2024/2025

## Partie 1 - Premiers pas

Vous allez exécuter les cellules comme demandé. Certaines fonctions vous seront inconnues, essayez de faire marcher votre intuition pour les comprendre !

**Question 1.** Exécutez la cellule suivante en cliquant sur la flèche noire, ou en appuyant sur `Ctrl + Entrée`.

```
[1]: print(f'C'est mon premier programme de 3A GII')
```

C'est mon premier programme de 3A GII

**Question 2.** Reprenez le code suivant pour afficher la phrase de votre choix.

```
[ ]:
```

**Question 3.** Exécutez la cellule suivante. Assurez-vous d'avoir bien compris la différence entre `c` et `{c}` dans le `print` à l'affichage.

```
[ ]: a = 2 # création d'une variable a à laquelle est associée la valeur 2
b = 3 # création d'une variable b à laquelle est associée la valeur 3
c = a * b - a # création d'une variable c
print(f'La valeur de c est {c}') # affichage de texte et de la valeur de c
```

La valeur de c est 4

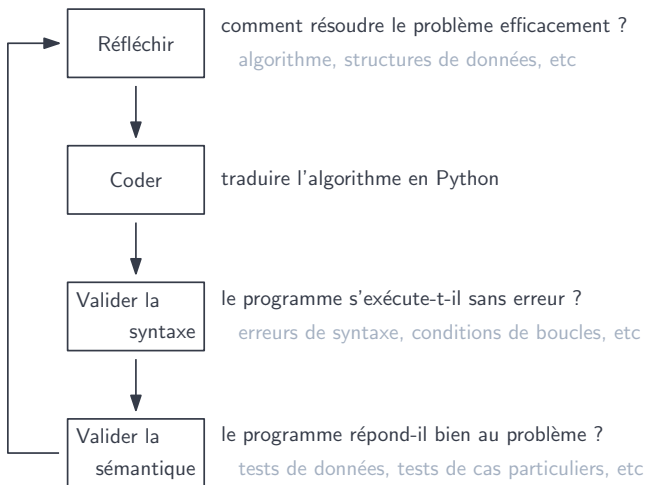
**Question 4.** Reprenez le code de la cellule précédente et modifiez le contenu des variables `a`, `b` et `c` pour qu'il s'affiche `8` (il y a plusieurs manières de faire).

```
[ ]:
```

**Question 5.** Pour travailler, on importe quasiment systématiquement les bibliothèques suivantes avec des alias pour simplifier leur usage :

# Réfléchir et coder

**! Attention :** programmer ça n'est pas écrire du code au hasard !



# Plan prévisionnel

- Rappels
- Structures conditionnelles et itératives
- Fonctions
- Collections
- Fichiers