

---

# "PY4BD"

## *Python for Big Data*

Georges Georgoulis – Alteractifs

Pour le Greta 92

Décembre 2022

---

---

j1

---





# Un cours amical pour l'environnement

## Supports

- Support de cours en ligne
  - PPT
  - PDF
  - exercices/corrigés/expériences
  - Sources \*.py
  - Classeurs Jupyter
- Documentation et références
  - URLs

## Prise de notes

- Invitation à utiliser Jupyter
- Espace de travail partagé

# Planning

		Semaines	1	2	3	4	5	6	7	8	9	10	11	12	13
Activités	Intervenants														
Administratif	Amandine, Estelle														
TRE	Hakim														
Python	Georges, Philippe														
Bibliothèques	Georges, Philippe														
Data Science	Benjamin, Lina														
Déploiement	Benjamin, Lina														
uProjet/Stage	Georges, Lina														

9/1/23

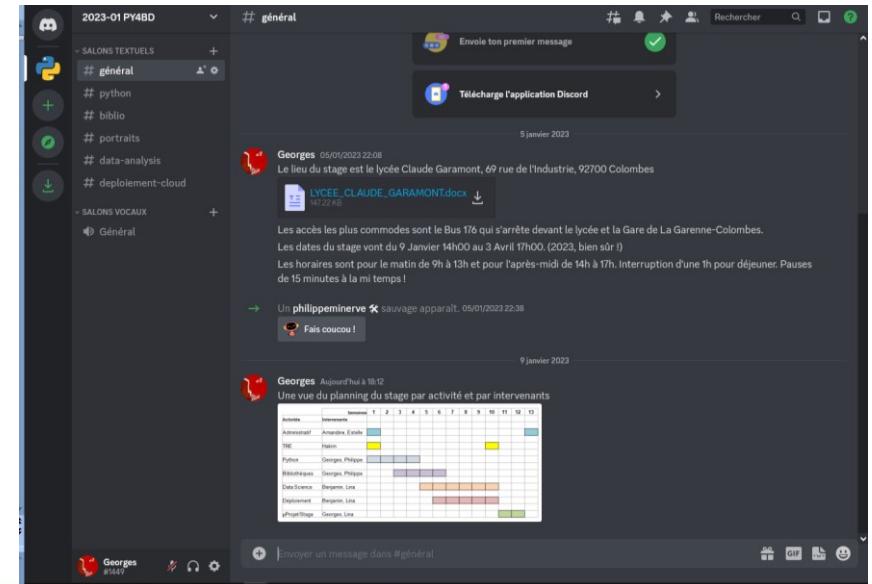
22/3/23



# Espace de travail partagé

<https://discord.com/>

- Cours
- Supports
- Exercices
- Corrigés
- Questions & Discussions
- Progression



# Discord

# J1 : Plan

---

- Importance de Python sur le marché du développement logiciel
- Histoire de Python
- Mettre en place l'environnement

# J1 : Objectifs

---

- Apprendre à se connaître
- Le développement logiciel
- Python
- Mettre en place l'environnement Python

# Python

## Pour quoi faire ?

- Traiter des masses
  - de fichiers texte
  - d'images
- Génie logiciel
- Automatiser des taches
- Taches de tests ou de développement
- Base de données
- Développement rapide
- Jeux
- Interfaces graphiques
- Bureau « Gnome » d'Ubuntu

## Sur quelles plateformes ?

- Windows,
- Unix,
- Linux,
- MacOS
- IOT
  - Arduino, Raspberry
- Cloud
  - Amazon AWS Python Boto
  - Big data /Data Science

---

# Développement logiciel PYTHON ET LE MARCHÉ



- Créé en 2008, [Stackoverflow](#) est une référence mondiale incontournable pour les développeurs.

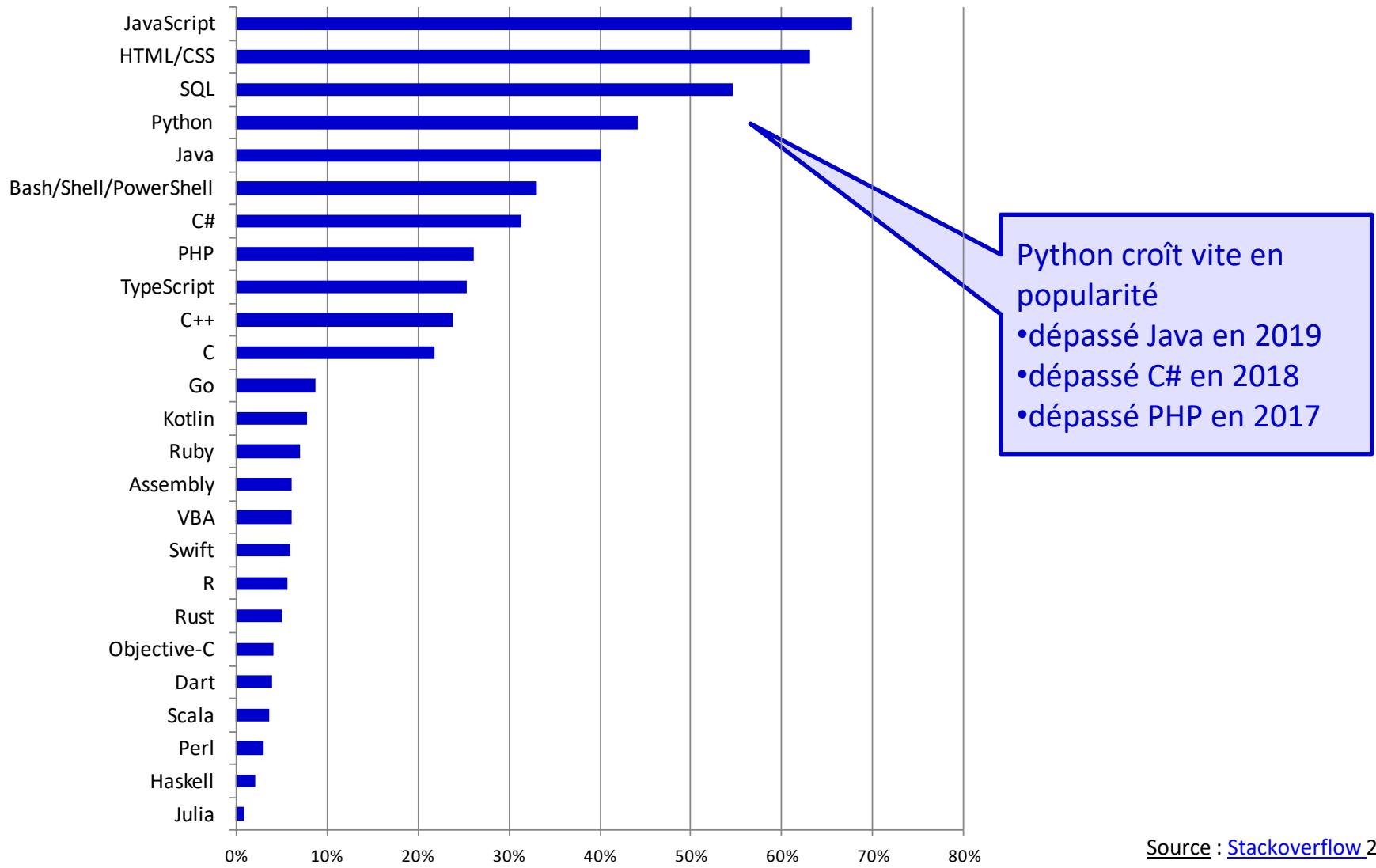
10 Millions de membres à fin 2018

16 Millions de questions traitées à mi 2018

- Beaucoup de développeurs disent apprendre en parcourant les forums
- Les questions les plus pointues y trouvent des réponses
- Stackoverflow dispose d'un corpus suffisamment significatif pour analyser les comportements des développeurs.
  - [Developer Survey Results](#)



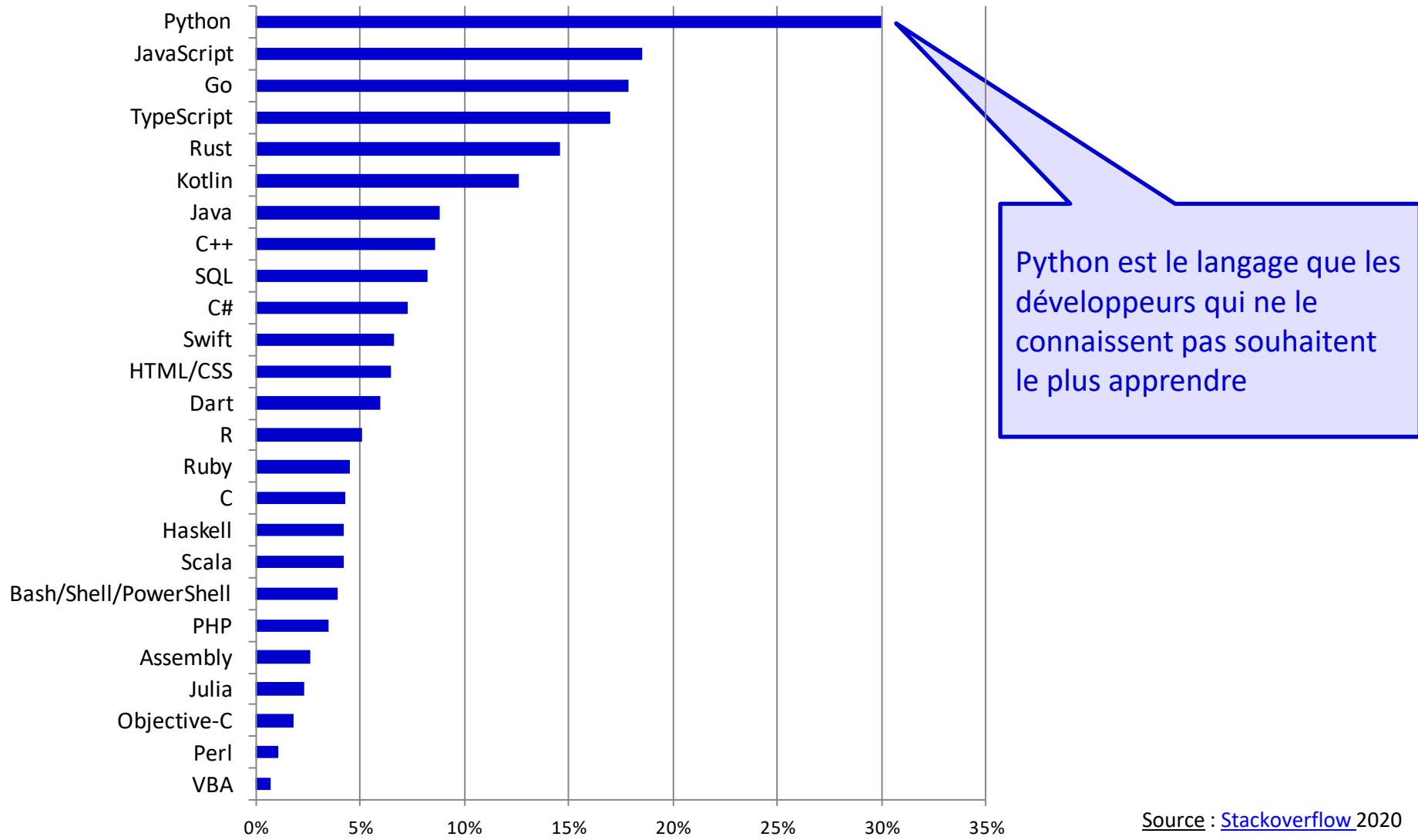
# Popularité des langages de programmation



Source : [Stackoverflow](#) 2020



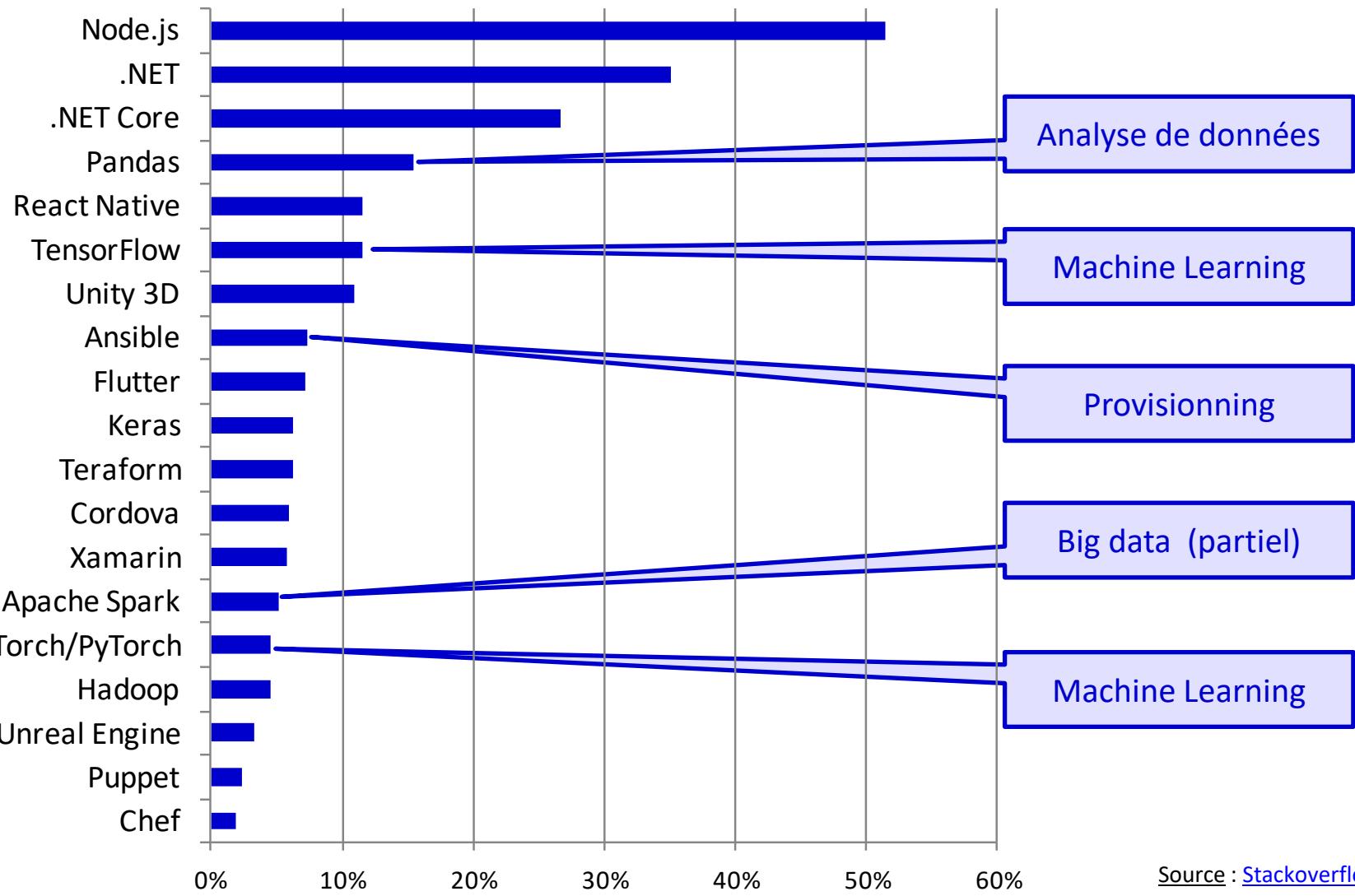
# Langages les plus désirés



Source : [Stackoverflow](#) 2020



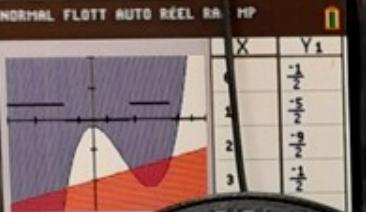
# Autres frameworks



Source : [Stackoverflow](#) 2020

# Texas Instruments

TI-83 Premium CE  
EDITION PYTHON



Voir la calculatrice  
en action et ses  
tutoriels gratuits  
[lestutosmaths.fr](http://lestutosmaths.fr)



TI-83 Premium CE  
Edition Python

Lycée et supérieur



MINISTÈRE  
DE L'ÉDUCATION NATIONALE,  
DE L'ENSEIGNEMENT SUPÉRIEUR  
ET DE LA RECHERCHE

Mode examen intégré

python™



Tout en  
français



Batterie  
rechargeable



Écran  
couleur



Mises à jour  
gratuites



Affichage naturel :  
MathPrint™



Câble USB  
inclus

Conforme aux nouveaux programmes

CASIO

fx-92

fx-92+  
Spéciale  
Collège

Stylo écrit  
Répéter 4  
Avancer de 20 p<sup>i</sup>  
Tourner de 90°

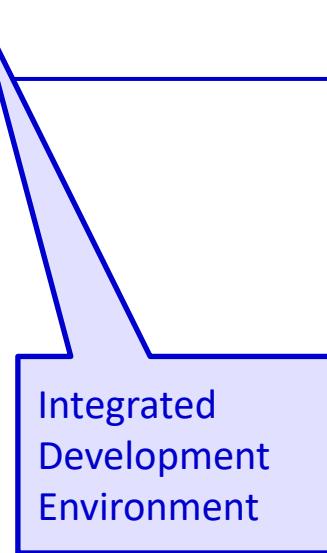
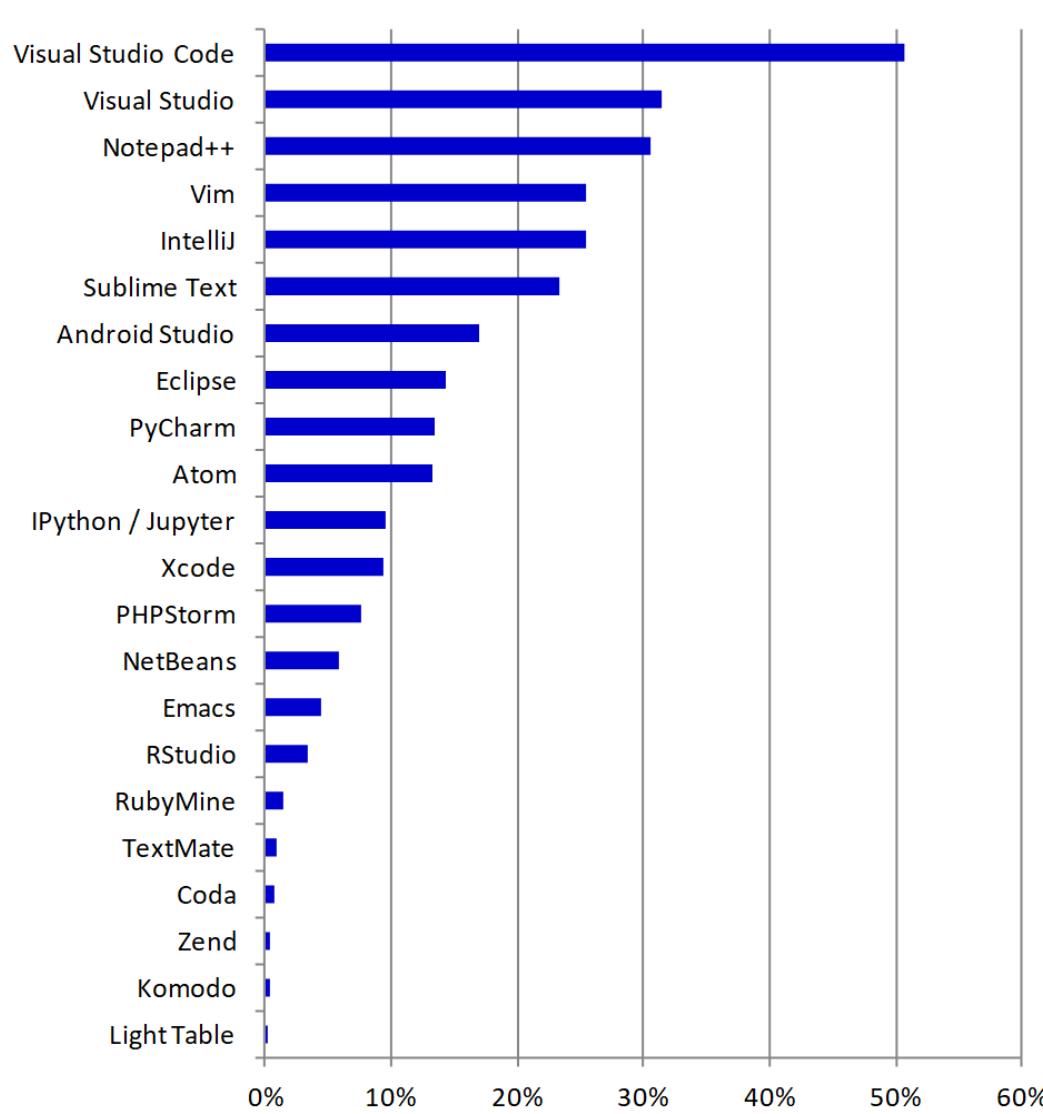


3€  
REMBOURSÉS

Pour l'achat d'une calculatrice  
fx-92+ Spéciale Collège  
du 13/06/2021 au 10/09/2021  
Max remboursement par élève de 100€



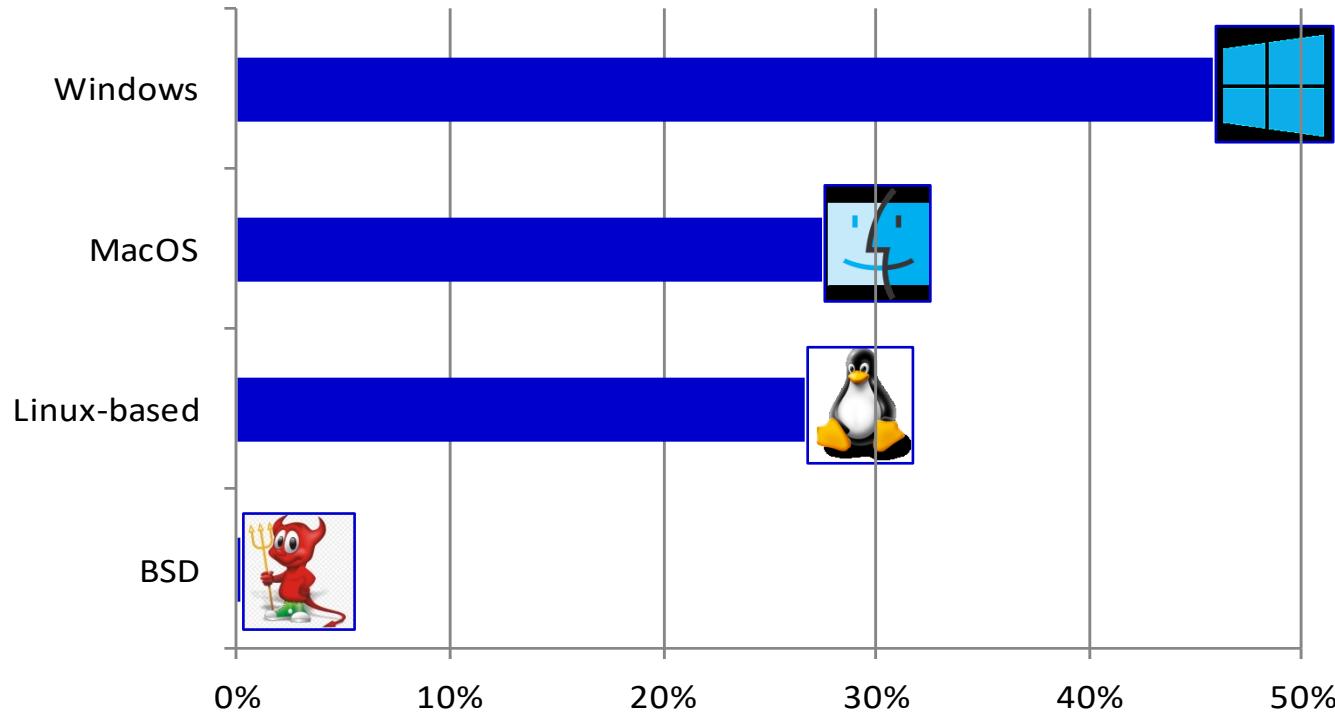
# Popularité des IDE



Source : [Stackoverflow 2019](#)



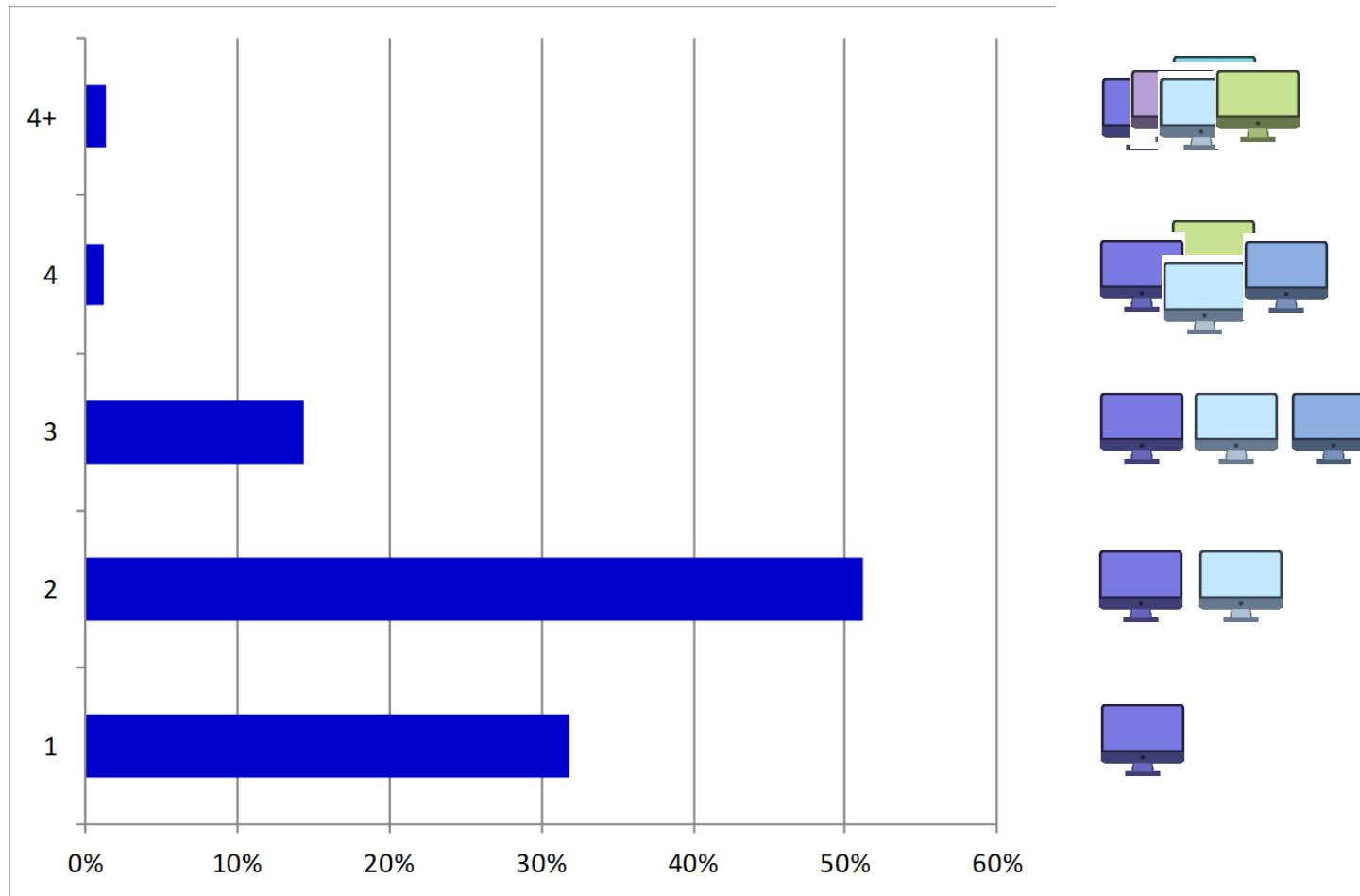
# Popularité des OS de développement



Source : [Stackoverflow](#) 2020



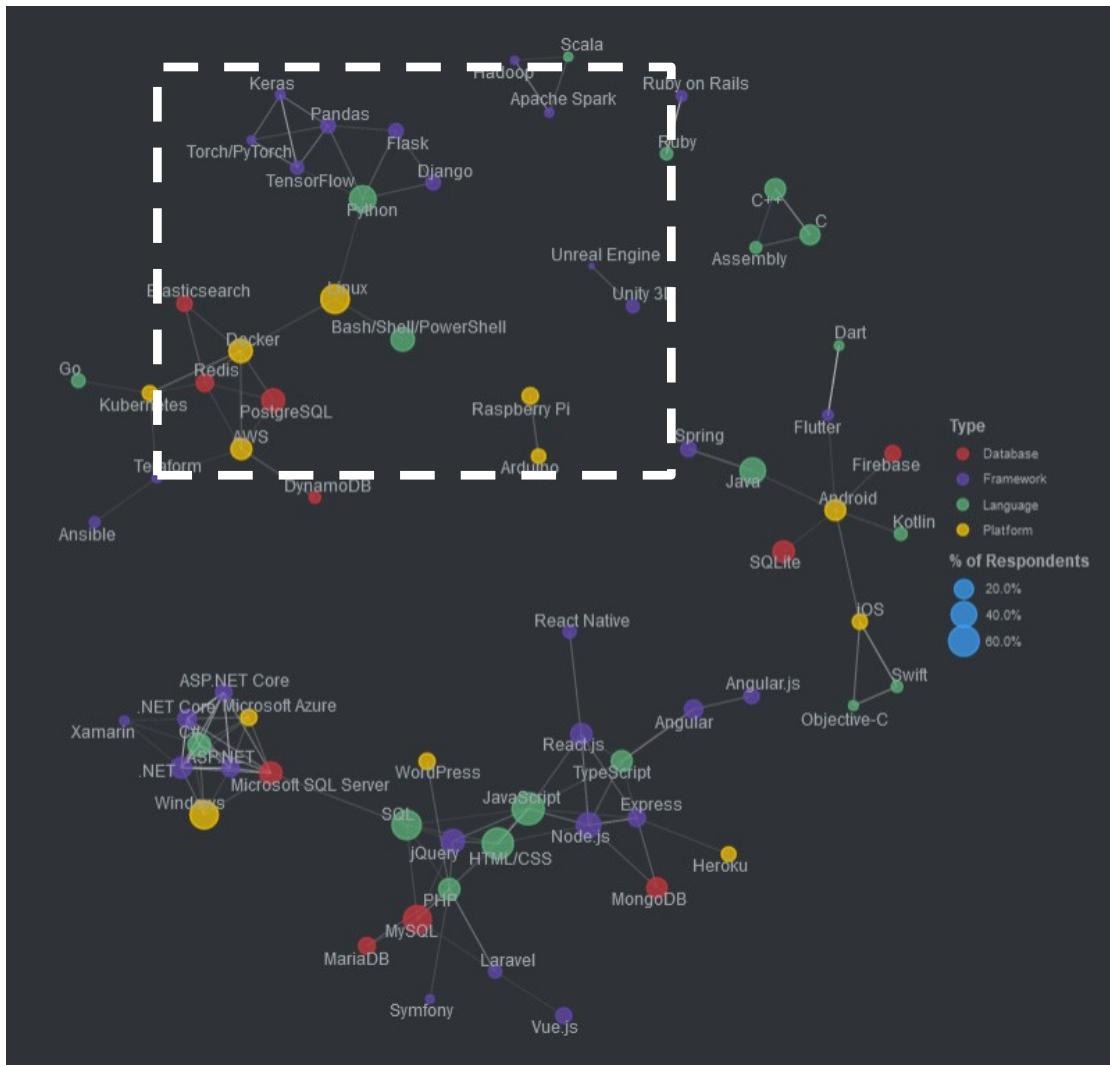
# Combien d'écrans dans une station de développement ?



Source : [Stackoverflow 2018](#)

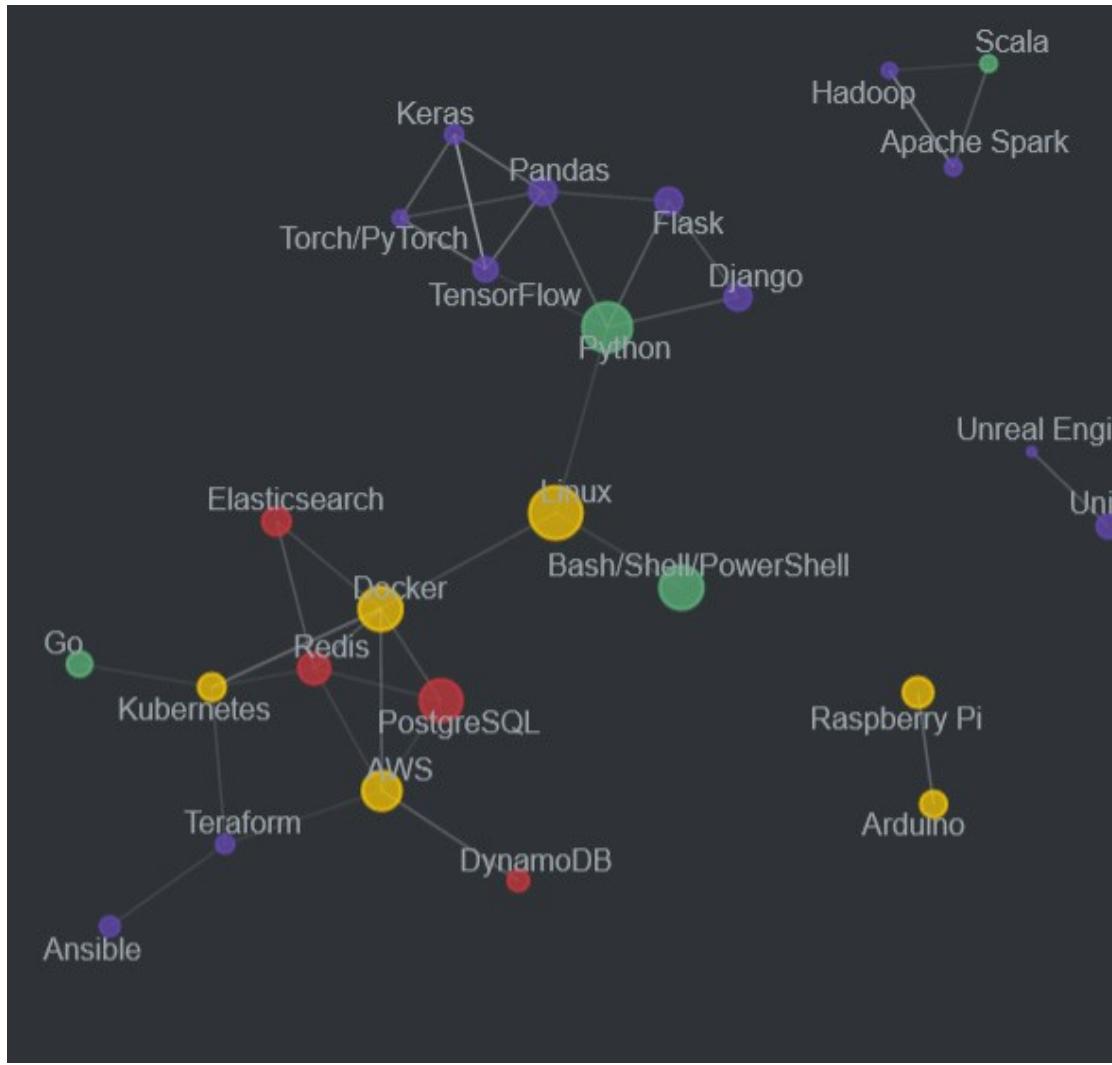


# Les technologies qui vont ensemble



Source : [Stackoverflow](#) 2020

# Les technologies qui vont ensemble (zoom)



Source : [Stackoverflow](#) 2020

---

# **PYTHON**

## **QUE PEUT ON EN ATTENDRE ?**

# Python

## Qualités du langage

- facile à apprendre, même pour les non-informaticiens
- notation légère, utilisation de l'indentation, lisible
- richesse d'expression
- fonctionnel
- orienté objet
- efficacité
- concision
- langage interprété
- scripting
- Open source
- Bibliothèques riches et nombreuses

## Bibliothèques

- Graphiques : wxPython, pyQT, pyGtk
- Traitement d'image : PIL, Pillow, OpenCV
- Base de données : SQLAlchemy
- Web : Requests, Scrapy, Django, Flask, BeautifulSoup
- Réseaux : twister, scrapy
- Analyse de données : Panda, Numpy, SciPy
- Visualisation de données : matplotlib
- Big data : Hadoop, PySpark
- Jeux : Pygame, pyglet
- Calcul formel : SymPy
- Traitement du langage naturel : nltk
- Interface à windows : pywin32

---

# **PYTHON**

# **HISTOIRE ET INSPIRATION**

# Guido van Rossum

- Hollandais
- né le 21 Janvier 1956
- crée Python
  - Février 1991, version 0.9.0
  - Juillet 2010, Python 2.7
  - Décembre 2008, **Python 3.0**
  - Janvier 2020, Python 2.7 n'est plus supporté
    - [Compte à rebours](#) expiré !
    - [Roadmap](#)
- Il était jusqu'à 2018 « *Benevolent Dictator For Life* »(BDFL) pour Python
  - dictateur bienveillant à vie

Celui que nous apprenons



By Doc Searls - 2006oscon\_203.JPG, CC BY-SA 2.0,  
<https://commons.wikimedia.org/w/index.php?curid=4974869>

# Un style marqué par l'humour des Monty Python



Graham Chapman, John Cleese, Eric Idle, Michael Palin, Terry Jones, Terry Gilliam  
Monty Python's Flying Circus

# PEP-0020 Le Zen de Python (par Tim Peters)

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!



Préfère :

la beauté à la laideur,  
l'explicite à l'implicite,  
le simple au complexe  
et le complexe au compliqué,  
le déroulé à l'imbriqué,  
l'aéré au compact.

Prends en compte la lisibilité.

Les cas particuliers ne le sont jamais assez pour violer les règles.

Mais, à la pureté, privilégie l'aspect pratique.

Ne passe pas les erreurs sous silence,  
... ou bâillonne-les explicitement.

Face à l'ambiguïté, à deviner ne te laisse pas aller.

Sache qu'il ne devrait avoir qu'une et une seule façon de procéder,  
même si, de prime abord, elle n'est pas évidente, à moins d'être Néerlandais.

Mieux vaut maintenant que jamais.

Cependant jamais est souvent mieux qu'immédiatement.

Si l'implémentation s'explique difficilement, c'est une mauvaise idée.

Si l'implémentation s'explique aisément, c'est peut-être une bonne idée.

Les espaces de nommage ! Sacrée bonne idée ! Faisons plus de trucs comme ça.

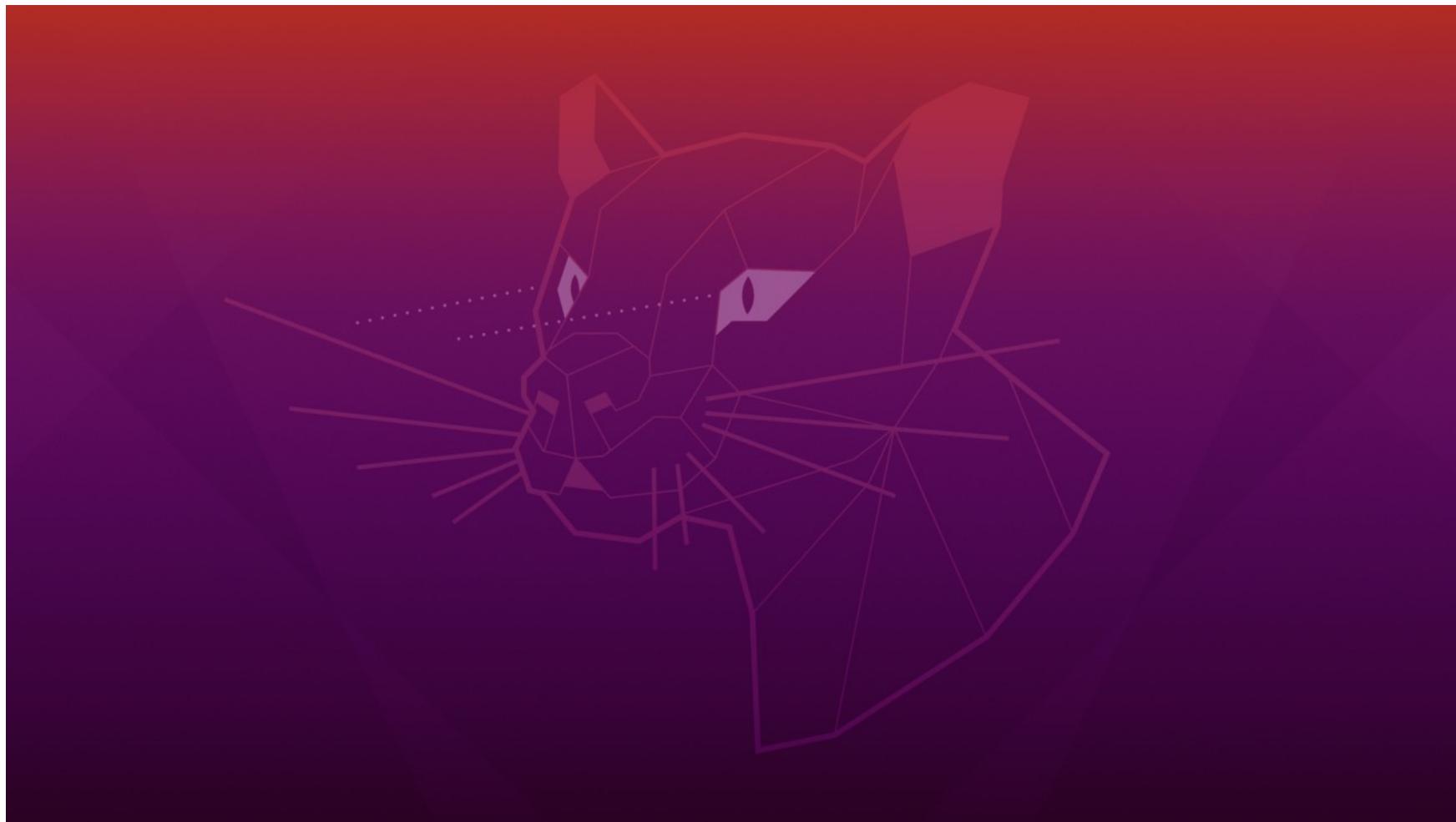


---

# **MISE EN PLACE DE L'ENVIRONNEMENT PYTHON3 & ANACONDA**



# Ubuntu, Focal Fossa, 20.04 LTS





# Environnement logiciel

- Vérifier la version d'OS

```
$ lsb_release -d  
$ lsb_release -a
```



# Fenêtre de commandes

- Pour ouvrir un terminal
  - CTRL- ALT -T
  - ou bien, rechercher l'application "Terminal"
- on est alors pris en charge par le "shell"
  - différentes variantes (Bash, Csh, Korn, ...)
- Interprète du langage de commande d'UNIX
  - Command Language Interface ou "CLI"
- Un véritable langage de programmation

A screenshot of a terminal window titled "georges@Houbountou: ~". The prompt "(base) georges@Houbountou:~\$" is visible at the top. The window has a dark background and a light-colored text area. The title bar includes standard window controls (minimize, maximize, close).

# Touches utiles en mode ligne

---

- Flèches
- Flèche en haut pour rappeler la dernière commande
- Tabulation pour compléter
- CTRL-C pour interrompre une commande
- CTRL-D pour simuler une fin de fichier/fermer le terminal
- CTRL-Z pour suspendre le processus en cours
- CTRL-U pour annuler la ligne en cours de frappe
- CTRL-S pour stopper un défilement
- CTRL-Q pour reprendre la lecture
- CTRL-O pour ignorer/reprendre la lecture des informations qui défilent
- Ces comportements sont pour la plupart des fonctions du shell



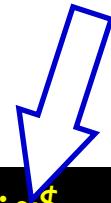
# Dossiers @ Linux (Ubuntu)

- L'utilisateur qui se connecte atterrit par défaut dans son "home directory"
- Pour répondre à la question "Où suis-je ?" ...  
...Regarder le prompt du shell
- Ou envoyer la commande "print working directory"
- Revenir à un endroit connu (par exemple, le home directory) par la commande "change directory"

```
(base) georges@Houbountou:~$
```

```
$ pwd
```

```
$ cd
```





# Fichiers @ Linux (Ubuntu)

- Pour répondre à la question : qu'est ce qu'il y a dans ce dossier ?...

...Lister les fichiers

```
$ ls
```

- Avec plus de détails

```
$ ls -l
```

- En incluant les fichiers dont le nom commence par ":"

```
$ ls -a
```

- En combinant les 2

```
$ ls -al
```

- Comme "dir" dans Windows



# Commande ls -al

A screenshot of a terminal window titled "georges@Houbountou: ~/ecconda". The window shows the command "ls -al" being run and its output. The output includes the following details:

```
(base) georges@Houbountou:~/ecconda$ ls -al
total 16
drwxrwxr-x  3 georges georges 4096 nov.  13 21:07 .
drwxr-xr-x 24 georges georges 4096 nov.  15 16:08 ..
-rw-rw-r--  1 georges georges 1148 nov.  13 21:06 ecjup1.ipynb
drwxrwxr-x  2 georges georges 4096 nov.  13 21:07 .ipynb_checkpoints
(base) georges@Houbountou:~/ecconda$
```

The terminal has a dark background with light-colored text. The window has standard Linux-style window controls at the top right.

- Le fichier "." désigne le dossier en cours.
- La fichier ".." désigne le dossier père du dossier en cours
- Les fichiers dont le nom commence par "." sont "cachés"
- Sont indiqués en particulier, les permissions, le groupe, le propriétaire, la taille, la date de création du fichier et son nom.



# Hiérarchie des fichiers

- Les fichiers sont rangés dans une arborescence dont la racine est "/"
- Les liens symboliques sont signalés ("raccourcis")
- Cet arrangement est généralement suivi par tous les systèmes UNIX.

```
.
├── bin    -> usr/bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib    -> usr/lib
├── lib32  -> usr/lib32
├── lib64  -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin  -> usr/sbin
├── snap
├── srv
├── sys
└── tmp
└── usr
└── var

24 directories
```

[https://fr.wikipedia.org/wiki/Filesystem\\_Hierarchy\\_Standard](https://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)



# Parcourir l'arborescence

- Pour visiter un sous-dossier du dossier en cours

```
$ cd ecconda  
$ ls
```

- Pour remonter d'un niveau

```
$ cd ..  
$ ls
```

- Pour se transporter ailleurs dans l'arborescence

```
$ cd /usr/bin  
$ ls
```

- Un chemin absolu commence par "/"
- Les autres chemins sont relatifs au dossier en cours

# Voir le contenu d'un fichier

- S'applique aux fichiers textes

- Faire défiler le contenu

```
$ cat toto.txt
```

- Faire défiler mais en contrôlant l'affichage (page d'écran par page d'écran ou moins)

```
$ more toto.txt  
$ less toto.txt
```

- Voir le début

```
$ head toto.txt
```

- Voir la fin

```
$ tail toto.txt
```

# Créer un fichier

- Créer un fichier vide

```
$ touch toto.txt
```

- Employer la commande **cat** et une redirection. On tape au clavier le contenu. Terminer par une fin de ligne "Enter" et une fin de fichier "CTRL+D"

```
$ cat >toto.txt
```

- Employer la commande **echo** et une redirection

```
$ echo "Voila le contenu" >toto.txt
```

- Employer un éditeur de texte

# Redirections

- Entrée standard – **stdin** – en général, le clavier

```
$ cat <toto.txt
```

- Sortie standard – **stdout** – en général, l'écran

```
$ cat >toto.txt
```

- Erreur standard – **stderr** – en général, la même chose que stdout

```
$ cat 2>toto.txt
```

- Nouveau fichier >
- Fichier existant que l'on rallonge >> (mode append)

```
$ cat >>toto.txt
```



# Editeurs de texte @ Linux (Ubuntu)

Mode fenêtre graphique :

- mode le plus ergonomique : gedit

```
$ gedit
```

Mode plein écran : (vieillot, répandu)

- nano

```
$ nano
```

- vi ou vim

```
$ vi
```

Mode ligne : (passepartout)

- ed – mode ligne

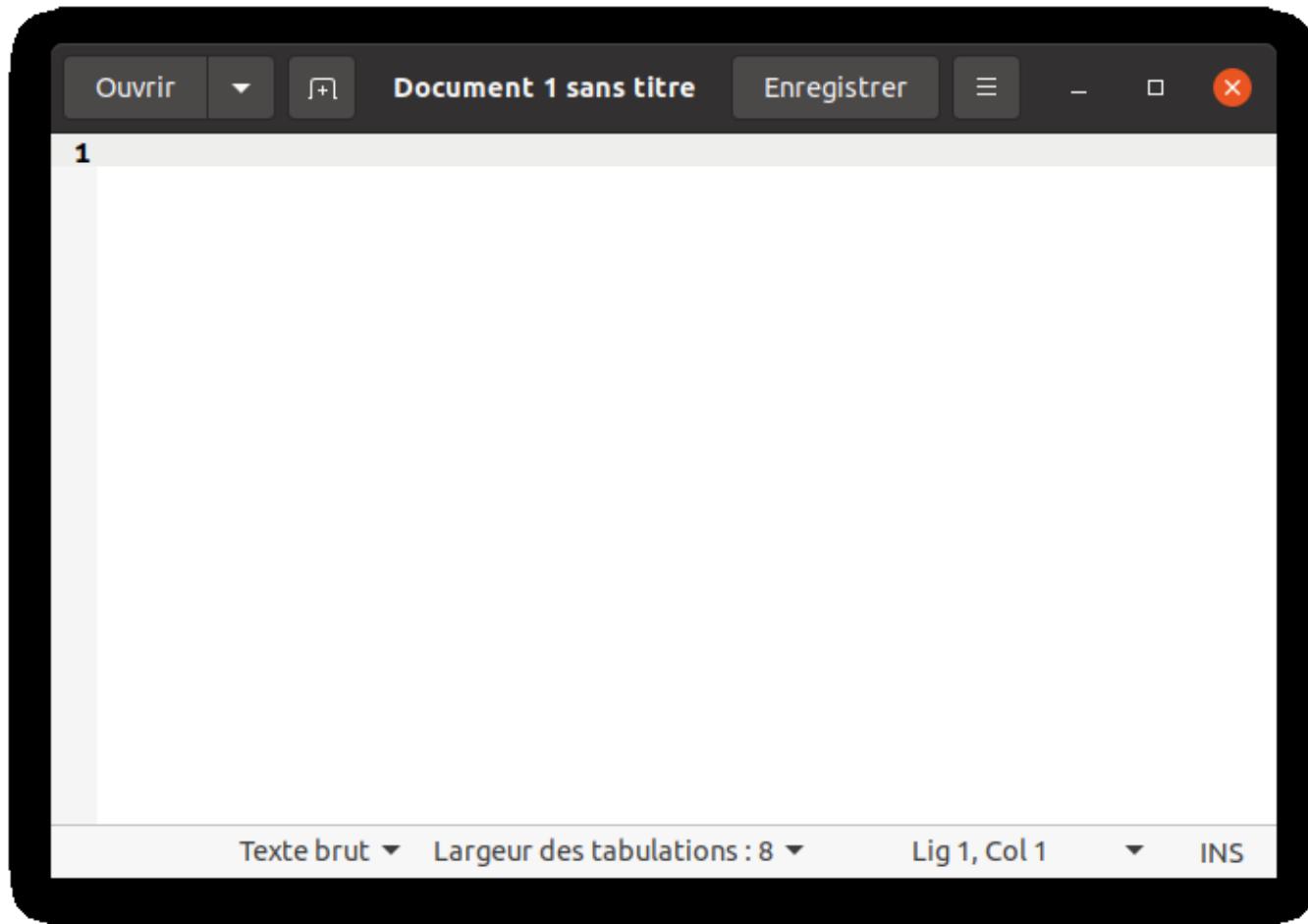
```
$ ed
```

- sed – stream editor

```
$ sed
```

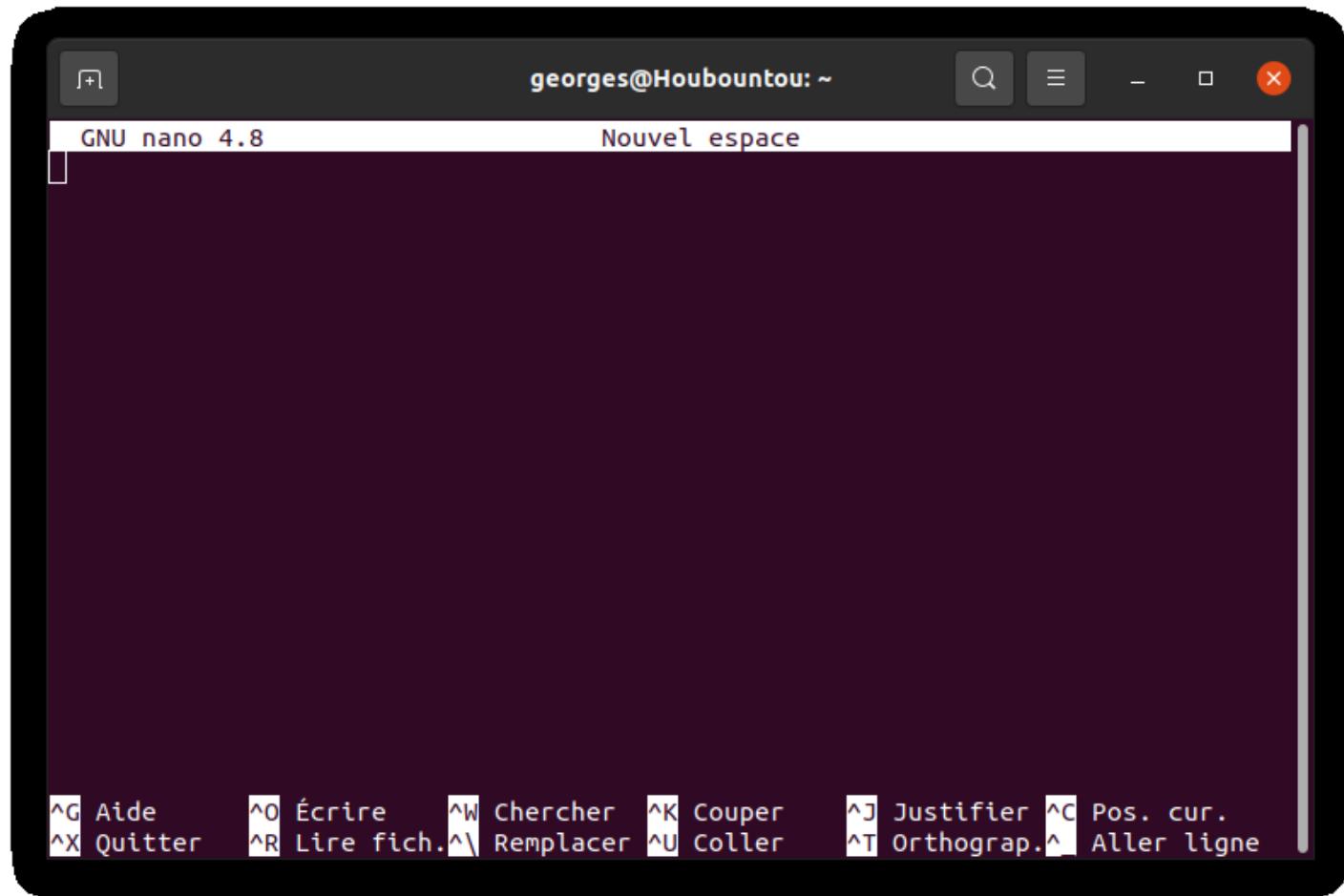


# gedit





# nano





vi

A screenshot of a terminal window titled "georges@Houbountou: ~". The window contains the Vim startup message:

```
VIM - Vi IMproved
version 8.1.3741
by Bram Moolenaar et al.
Modified by team+vim@tracker.debian.org
Vim is open source and freely distributable

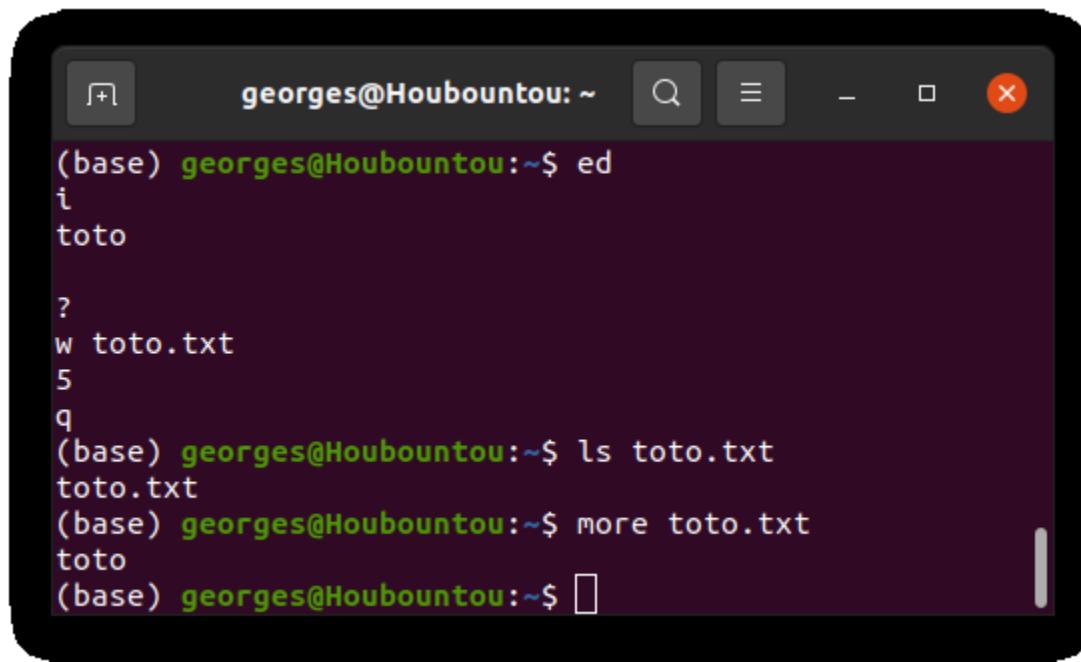
      Help poor children in Uganda!
type :help iccf<Enter>      for information

type :q<Enter>          to exit
type :help<Enter> or <F1> for on-line help
type :help version8<Enter> for version info
```

The terminal has a dark purple background and a light purple cursor.



# ed



A screenshot of a terminal window titled "georges@Houbountou: ~". The terminal shows the following session:

```
(base) georges@Houbountou:~$ ed
i
toto

?
w toto.txt
5
q
(base) georges@Houbountou:~$ ls toto.txt
toto.txt
(base) georges@Houbountou:~$ more toto.txt
toto
(base) georges@Houbountou:~$ 
```



# sed

```
georges@Houbountou: ~
(base) georges@Houbountou:~$ sed 's/toto/bar/' <toto.txt >bar.txt
(base) georges@Houbountou:~$ ls
anaconda3 Bureau ecconda Modèles Public          toto.txt      Vidéos
bar.txt    Documents Images Musique Téléchargements Untitled.ipynb
(base) georges@Houbountou:~$ more bar.txt
bar
(base) georges@Houbountou:~$ more toto.txt
toto
(base) georges@Houbountou:~$ 
```



# Gestion des fichiers

- Détruire un fichier
- Renommer/Déplacer un fichier (move)
- Copier un fichier
- Créer un dossier vide
- Détruire un dossier
- Changer la propriété d'un fichier utilisateur et groupe utilisateur groupe
- Changer la protection d'un fichier ("mode")

```
$ rm toto.tx
```

```
$ mv toto.txt tata.txt
```

```
$ cp toto.txt ~/essai
```

```
$ mkdir mon_dossier
```

```
$ rm -r mon_dossier
```

```
$ chown georges:georges toto.txt  
$ chown georges toto.txt  
$ chgrp georges toto.txt
```

```
$ chmod u+x toto.txt  
$ chmod 755 toto.txt
```



# Permissions

## Pour qui ?

- user
  - group
  - other

# Pour quoi ?

- read
  - write
  - execute

- Au lieu de rwx, on peut noter un chiffre octal 0-7
    - Exemple : 755 permission de tout faire pour user, mais lire et exécuter pour group et other

-rw-rw-r--



# Script shell

- Créer un fichier

```
$ vi script.sh
```

- Introduire la commande echo

```
echo "Bonjour"
```

- introduire le **shebang** à la 1<sup>ère</sup> ligne
  - désigne l'interprète de commandes qui sera invoqué pour exécuter le script

```
#! /bin/bash
```

- rendre le script exécutable

```
$ chmod u+x script.sh
```

- l'exécuter

```
$ ./script.sh
```



# Processus

- Que fait Ubuntu ?

- Lister mes processus

```
$ ps
```

- Lister les processus plus ceux associés à la session

```
$ ps -a
```

- Lister tous les processus en détail

```
$ ps aux
```

- le PID (process ID) est un numéro unique pour le système

- Utile pour tuer un processus

```
$ kill -9 1234
```





# Pipes | (1/2)

- Commande qui donne le nombre de lignes, mots, caractères d'un fichier : word count
  - pour avoir les lignes seulement
- Commande combinée pour compter les processus
- Commande qui cherche une chaîne de caractères dans un fichier
- Commande qui trouve les détails du processus mysql
  - et qui les affiche par écrans

```
$ wc toto.txt
```

```
$ wc -l toto.txt
```

```
$ ps | wc -l
```

```
$ grep -i Georges toto.txt
```

```
$ ps aux | grep -i mysql
```

```
$ ps aux | grep -i mysql | more
```



# Pipes | (2/2)

- Un programme qui lit stdin et écrit sur stdout s'appelle un filtre
- pipe, noté " | " enchaîne des filtres
  - la sortie standard du précédent est dirigée sur l'entrée standard du suivant
- La commande **tee** permet de capturer dans un fichier une étape intermédiaire

```
$ ps | tee proc.txt | wc -l
```



# & et &&

- Lancer simultanément deux commandes, la 1<sup>ère</sup> s'exécutant en arrière plan

```
$ ./script.sh & wc -l toto.txt
```

- Lancer deux commandes à la suite, à la condition que la 1<sup>ère</sup> se soit bien terminée

```
$ ./script.sh && wc -l toto.txt
```

- L'inverse, c'est à dire exécuter la 2<sup>ème</sup> seulement si la première s'est mal passée, s'écrit

```
$ ./script.sh || wc -l toto.txt
```



# Source

- Unix crée facilement des processus qui vivent, font leur travail et meurent.
- Lorsque shell exécute une commande, il crée un sous-processus et la commande est exécutée dans ce processus FILS
  - le processus père n'est pas affecté
- Si l'on veut que la commande soit exécutée dans le processus PERE, on emploie la commande **source**

```
$ source myenv/bin/activate
```

```
$ source .bashrc
```



# Variables d'environnement

- Une fois

```
$ export VAR=valeur
```

- De manière permanente

- en ajoutant cette ligne à un script exécuté au lancement d'un shell
  - à l'ouverture de session desktop GUI
  - en ajoutant une ligne pour tous les users, dans ou pour un seul user, dans

```
~/.bashrc
```

```
~/.profile
```

```
VAR="valeur"
```

```
/etc/environment
```

```
~/.pam_environment
```

- Contrôler la valeur par ou bien

```
$ echo $VAR
```

```
$ printenv
```

```
$ printenv VAR
```

<https://help.ubuntu.com/community/EnvironmentVariables>



# Divers

- Répéter la dernière commande

```
$ !!
```

- La voir avant de la répéter

```
$ !!:p
```

- Employer à nouveau l'argument de la commande précédente

```
$ rm !$
```

- Afficher cette commande

```
$ rm !$:p
```

- Voir les processus en attente

```
$ jobs
```

- Réactiver le processus 1

```
$ %1
```



# Bibliographie

---

- [https://doc.ubuntu-fr.org/projets/école/scripting/initiation au shell](https://doc.ubuntu-fr.org/projets/école/scripting/initiation_au_shell)

---

# **PRISE EN MAIN DE L'ENVIRONNEMENT PYTHON3 & ANACONDA**



# Anaconda

- Noter que la forme de l'invite (prompt) montre que l'environnement par défaut d'Anaconda est activé  
(base)

```
(base) georges@Houbountou:~$
```

- Vérifier la version installée

```
$ conda --version
```

- Lancer

```
$ anaconda-navigator
```



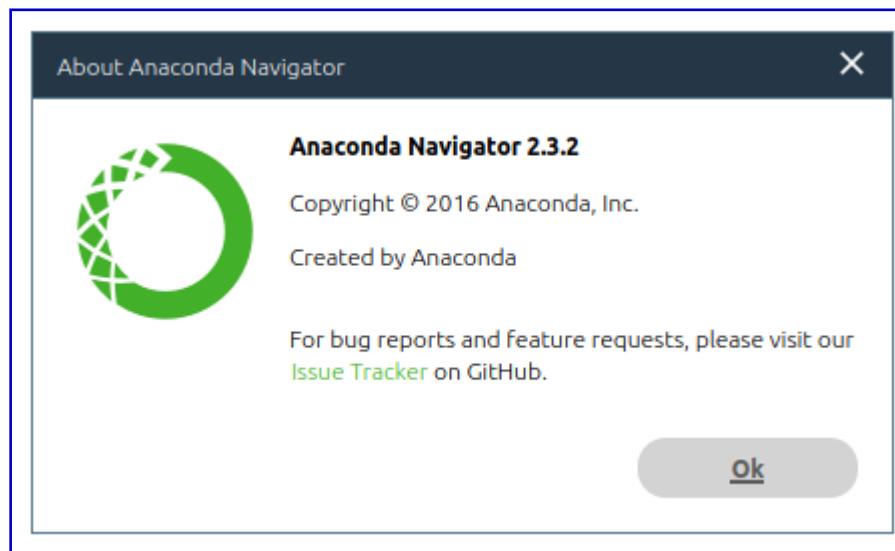
# Anaconda Navigator

The screenshot shows the Anaconda Navigator application window. The left sidebar has links for Home, Environments, Learning, Community, Anaconda Notebooks (with a 'New!' badge), Documentation, and Anaconda Blog. The main area displays a grid of data science tools:

- DataSpell**: An IDE for exploratory data analysis and prototyping machine learning models. Version 3.4.4. Description: "DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment." Buttons: [Install](#), [Launch](#).
- JupyterLab**: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Version 3.4.4. Description: "An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture." Button: [Launch](#).
- jupyter Notebook**: A web-based, interactive computing notebook environment. Version 6.4.12. Description: "Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis." Button: [Launch](#).
- Qt Console**: A PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Version 5.3.2. Description: "PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more." Button: [Install](#).
- Spyder**: A scientific Python Development Environment. Version 5.3.3. Description: "Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features." Button: [Launch](#).
- Datalore**: Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no.



# Version d'Anaconda Navigator





# Python en ligne de commande

- Ouvrir une fenêtre de commande
- Vérifier quelle version de Python est accessible
- Lancer l'interprète Python en ligne de commande
- Observer qu'Anaconda est mentionné dans la version de Python
- Observer la forme de l'invite de l'interpréteur Python

```
$ python -v  
$ python --version
```

```
$ python
```

```
>>>
```



# Python en ligne de commande

- Envoyer la commande

```
>>> print("Hello, World !")
```

- Puis quitter l'interpréteur par

```
>>> quit()
```

- A l'aide de votre éditeur de texte préféré, créer un fichier script.py avec le contenu ci-contre

```
print("Hello, World !")
```

- Revenir au Shell pour lancer votre script par

```
$ python script.py
```

- On peut faire aussi

```
$ python -m script
```



# \*.pyc

- Après avoir fait

```
$ python script.py
```

- Noter l'apparition d'un nouveau dossier
  - pas toujours
  - quand il y a des imports

```
$ ls  
...          __pycache__
```

- Se déplacer dans ce dossier et examiner son contenu

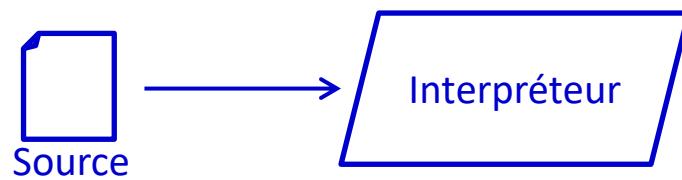
```
$ cd __pycache__  
$ ls
```



# Interprété ou Compilé

## Interpréteur

- Boucle
  - Lire
  - Evaluer
  - Imprimer
- REPL = Read – Eval – Print Loop



script.py

Python est essentiellement interprété

## Compilateur

- Code source
- Code objet
  - Exécuté par le processeur (machine)
  - ou par une machine virtuelle



script.py → script.pyc

Les .pyc sont interprétés plus rapidement



# Spyder

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar below the menu contains various icons for file operations like Open, Save, Print, and Run. The left pane displays two open files: 'untitled2.py' and 'script.py'. The code in 'script.py' is:

```
1 print("Bonjour")
2
```

The right pane features a 'Usage' help panel with instructions on how to get help for objects using **Ctrl+I**. It also includes a 'New to Spyder? Read our tutorial' link. Below the help panel is the IPython console, titled 'Console 1/A'. The console output shows:

```
Python 3.9.13 (main, Aug 25 2022, 23:26:10)
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:
```

At the bottom, status bars show 'conda: base (Python 3.9.13)', 'Completions: conda(base)', 'LSP: Python', 'Line 2, Col 1', 'ASCII', 'LF', 'RW', and 'Mem 66%'. The overall theme is dark.



# Spyder

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar below has various icons for file operations like Open, Save, Copy, Paste, and Run. The left pane displays two tabs: 'untitled2.py' and 'script.py'. The 'script.py' tab contains the following code:

```
1 print("Bonjour")
2
```

The right pane shows a file browser with the following directory structure:

Name	Date Modified
__pycache__	15/11/2022 22:55
script.py	15/11/2022 22:54

The bottom pane is the 'Console 1/A' tab, which outputs:

```
Python 3.9.13 (main, Aug 25 2022, 23:26:10)
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('/home/georges/ecpython/script.py', wdir='/home/georges/ecpython')
Bonjour

In [2]:
```

At the bottom of the interface, status bars show 'conda: base (Python 3.9.13)', 'Completions: conda(base)', 'LSP: Python', 'Line 2, Col 1', 'ASCII', 'LF', 'RW', and 'Mem 66%'. There are also tabs for 'IPython Console' and 'History'.



# Jupyter

The screenshot shows a web browser window with the URL `localhost:8888/notebooks/ecjupyter/bonjour.ipynb`. The browser title bar has three tabs: "ecjupyter/" (inactive), "bonjour - Jupyter Notebook" (active, highlighted in orange), and "ecpandas - Jupyter Notebook" (inactive). The main content area of the browser displays a Jupyter Notebook interface. At the top, there's a toolbar with various icons for file operations, cell execution, and kernel selection. Below the toolbar, the notebook header reads "jupyter bonjour Dernière Sauvegarde : il y a 6 minutes (auto-sauvegardé)". On the right side of the header, there's a Python logo icon and a "Se déconnecter" button. The main content area contains a section titled "Essai Jupyter". Below it, a code cell shows the command `Entrée [1]: print("Bonjour")` and its output "Bonjour".

- Anaconda Navigator prend en charge le démarrage du serveur jupyter et le démarrage du navigateur qui joue le rôle de client.
- On peut lancer jupyter depuis la ligne de commande en faisant :

```
$ jupyter notebook
```



# Scripts avec Jupyter

- Depuis le GUI de Jupyter
  - Menu File, Choix Download as
    - choisir Python (.py)

```
$ jupyter nbconvert hello.ipynb  
--to hello.py
```

- Le contenu inclut :

```
#!/usr/bin/env python  
# coding: utf-8  
# In[1]: ...
```

- Rendre le script exécutable

```
$ chmod u+x hello.py
```

- L'exécuter

```
$ ./hello.py
```

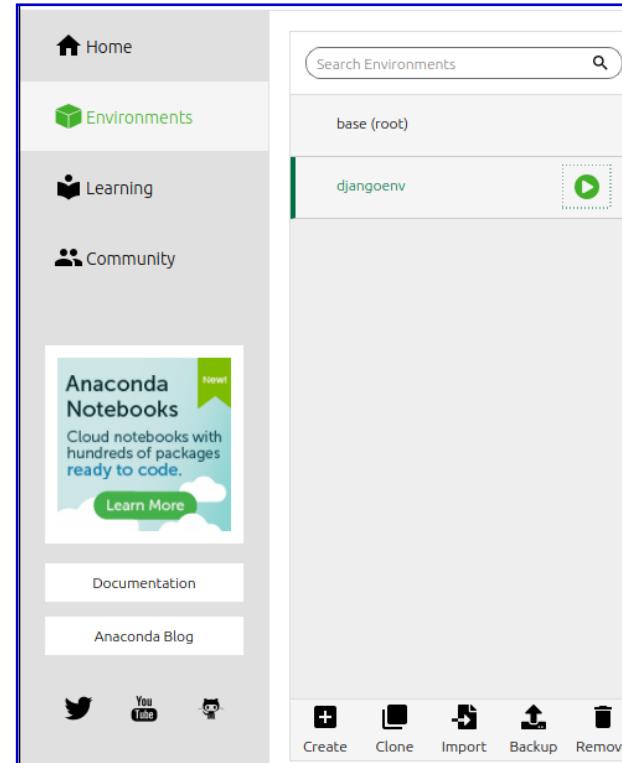


# Environnements

- Python arrive avec des bibliothèques très riches
- Anaconda réunit plus de 400 modules et la possibilité de travailler avec les différentes versions de Python
- Chaque projet a des exigences spécifiques en termes de versions de Python et de bibliothèques installées
- Pour chaque projet, un environnement isolé des autres est souhaitable

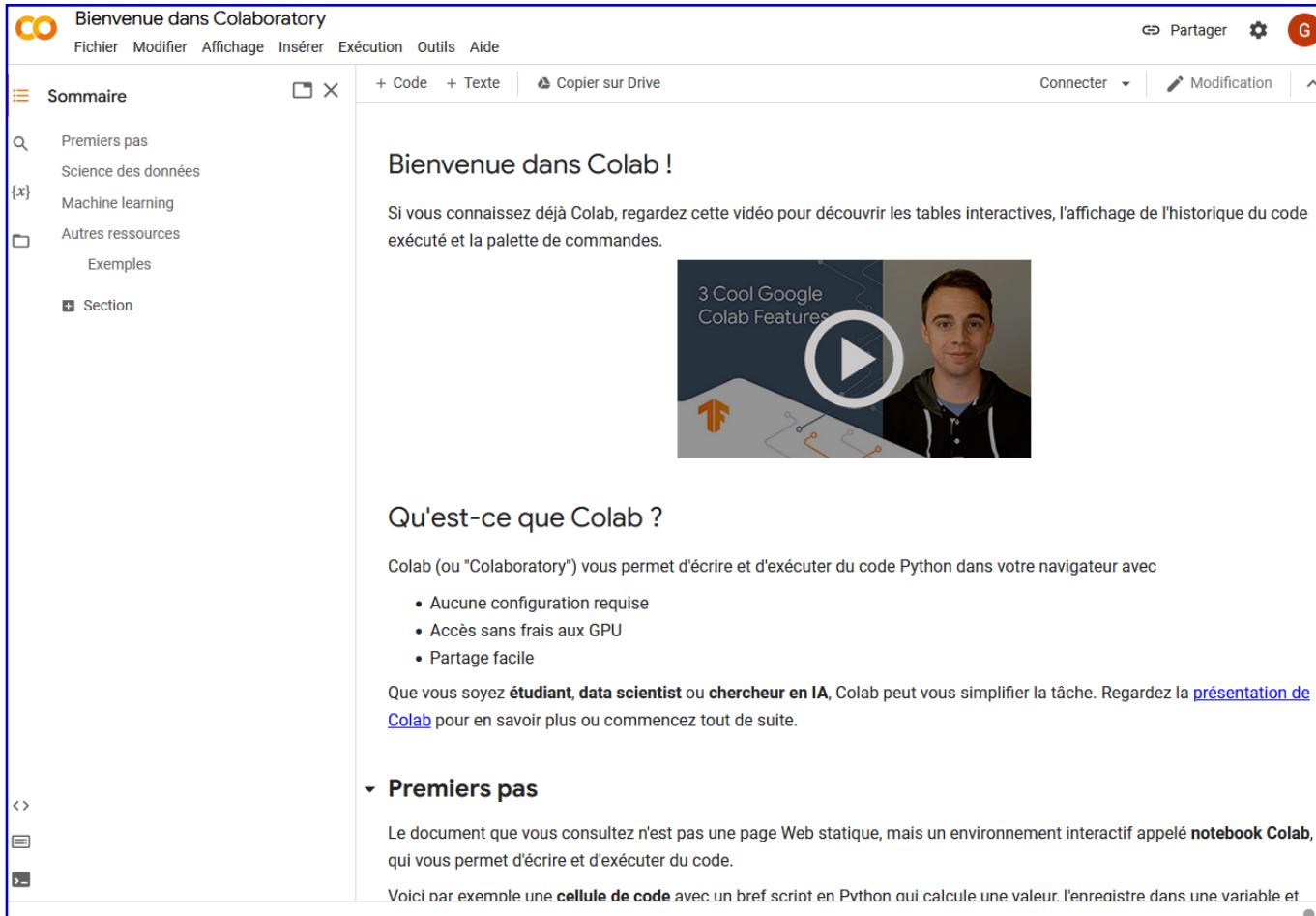
# Environnements avec Anaconda

- Choisir **Environments**, puis **Create**.
  - Accepter les modules qu'Anaconda introduit dans tout nouvel environnement.
  - Rechercher parmi les modules non installés, ceux que vous souhaitez.
  - Faire **Apply**.
- 
- Avec **Anaconda Navigator**, choisissez votre environnement depuis la page de garde
  - Avec le **shell**, activez/désactivez votre environnement



```
$ conda activate mon_envt  
(mon_envt) $ conda deactivate
```

# Jupyter en ligne



Bienvenue dans Colaboratory

Fichier Modifier Affichage Insérer Exécution Outils Aide

Partager G

Sommaire

- Premiers pas
- Science des données
- {x} Machine learning
- Autres ressources
- Exemples
- Section

+ Code + Texte Copier sur Drive

Connecter Modification

## Bienvenue dans Colab !

Si vous connaissez déjà Colab, regardez cette vidéo pour découvrir les tables interactives, l'affichage de l'historique du code exécuté et la palette de commandes.



## Qu'est-ce que Colab ?

Colab (ou "Colaboratory") vous permet d'écrire et d'exécuter du code Python dans votre navigateur avec

- Aucune configuration requise
- Accès sans frais aux GPU
- Partage facile

Que vous soyez **étudiant, data scientist ou chercheur en IA**, Colab peut vous simplifier la tâche. Regardez la [présentation de Colab](#) pour en savoir plus ou commencez tout de suite.

### Premiers pas

Le document que vous consultez n'est pas une page Web statique, mais un environnement interactif appelé **notebook Colab**, qui vous permet d'écrire et d'exécuter du code.

Voici par exemple une **cellule de code** avec un bref script en Python qui calcule une valeur, l'enregistre dans une variable et

<https://colab.research.google.com/>



# Pratique

---

C'est le moment de mettre en place votre dispositif de sauvegarde

- Clé USB
  - Dropbox
  - Google Drive
  - One Drive
- 
- Pour les supports de cours
  - Pour vos notes
  - Pour vos exercices, devoirs, projets

# Quizz (1/3)

---

- Qui est Guido van Rossum ?
- Que signifie PEP ?
- Python est-il interprété ou compilé ?
- Quelle est l'OS de développement utilisé dans ce cours ?
- Que signifie REPL ?
- Qu'est ce qu'un IDE ?
- Comment s'appelle l'IDE natif de Python ?
- Quel est l'IDE recommandé pour la suite de ce cours?
- Quelle est la version de Python que nous allons apprendre/pratiquer ?
- Pourquoi ?

# Quizz (2/3)

---

- Pourquoi apprendre Python ?
  - [ ] c'est un langage général
  - [ ] il fonctionne sur toutes les plateformes
  - [ ] il présente des concepts de programmation intéressants
  - [ ] les bibliothèques sont riches
- A quoi sert un deuxième écran quand on développe ?
  - [ ] à jouer pendant le travail (regarder des films)
  - [ ] à consulter Stackoverflow
  - [ ] à rechercher des templates de code sur internet
- Pourquoi utiliser Anaconda ?
  - [ ] parce que toutes les bibliothèques utiles au Data Analyst sont préinstallées
  - [ ] parce les opérations sur les environnements sont plus simples
  - [ ] parce que les dépendances sont traitées de manière cohérente
  - [ ] parce que l'accès aux outils Jupyter et Spyder est facile

# Quizz (3/3)

---

- En quoi Python est-il utile pour un administrateur système et réseau?
  - [ ] pour écrire des scripts plus puissants que les Shell Scripts
  - [ ] pour mieux communiquer avec les développeurs d'applications
  - [ ] pour supporter une démarche DEVOPS
- En quoi Python est-il utile pour l'analyse de données ?
  - [ ] pour la préparation des données
  - [ ] pour le nettoyage des données
  - [ ] pour la mise en forme, normalisation des données
  - [ ] à cause des fonctions et des bibliothèques statistiques disponibles
- Qu'est ce que le site StackOverflow ?

# Merci !

- Restons en contact :

- Georges Georgoulis – [ggeorgoulis@alteractifs.org](mailto:ggeorgoulis@alteractifs.org) – 06 12 68 40 06



- The Alteractifs logo consists of the word "Alteractifs" in a white, rounded, sans-serif font, set against a bright orange, rounded rectangular background.

- Coopérative d'activité et d'entrepreneurs [www.alteractifs.org](http://www.alteractifs.org)