



Matthias COLIN

# Introduction



- Python project : [python.org](https://python.org)
- Language created by Guido van Rossum
  - 1989 : 1<sup>st</sup> version (0.9)
  - 1994 : version 1.0
  - 2000 : version 2.0
  - **2001** : version 2.1 (Python Software Foundation)
  - **2008** : version 3.0 (non compatible 2.x)
  - 2024 : **3.13** et 2.7 (eol 1/1/2020)



# Assets of the language

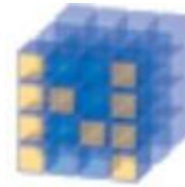
- Multi platforms
- Interpreted : `python [-i monscript.py]`
- Simple Syntax
- 3 paradigms of programming
  - **Functional** `map(sqrt, [1, 4, 9])`
  - **Imperative** `while delta < epsilon:  
                    delta = computeAgain()`
  - **Object** `valeurs = [3, 5, 7]  
            pos = valeurs.index(7)`
- Rich Integrated Library + External ones (PyPI)
- Big Community

# Programmation Web



# Calcul Scientifique, Data Science

- SciPy.org
  - NumPy : [numpy.org](http://numpy.org)
  - SciPy
  - Matplotlib : [matplotlib.org](http://matplotlib.org)
  - SymPy : [sympy.org](http://sympy.org)
  - Pandas : [pandas.pydata.org](http://pandas.pydata.org)
  - IPython : [ipython.org](http://ipython.org)
  - Notebook
- Travis Oliphant, Eric Jones, and Pearu Peterson



IP[y]:



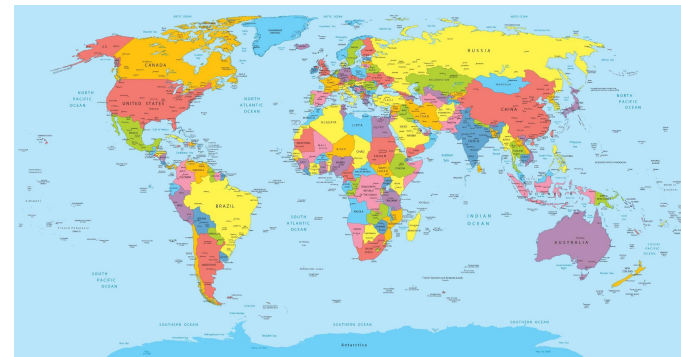
# IA



- <https://www.tensorflow.org/>
- <https://pytorch.org/>
- <https://scikit-learn.org>
- <https://keras.io/>
- <https://github.com/Microsoft/cntk>
- <https://github.com/Theano/Theano>
- <https://www.automl.org/>
- <https://shap.readthedocs.io/>
- <https://github.com/BobLd/YOLOv4MLNet>

# GIS

- Data : GeoPandas, GeoAlchemy, Xarray, Shapely, GeoPy, Geos, Fiona, GDAL, Rasterio, OGR, RSGISlib, PySAL, TorchGeo, Rasterstats, WhiteboxTools, scikit-mobility, EarthPy, Geocoder, PyCRS, RasterVision, osmnx, Overpy, geospatial-learn, GeoMesa, RasterFrames, laspy, PDAL, h3-py, Rtree,
- Spatial : PySAR, SarPy, snappy, PyRAT
- Cartes/Visualisation : ipyleaflet, geoplot, cartopy, folium, GeoViews, geoplot, Pydeck, PyVista, Open3D, geemap, reportlab
- Web : GeoDjango
- Liaison logiciels : ArcGIS, qgis



# Resources

- Documentation :
  - [docs.python.org/3/](https://docs.python.org/3/)
    - Tutorial
    - Library Reference
    - Language Reference
    - Python Module Index
  - Help from interpreter : dir, help, ?
- Python Package Index : PyPI
  - [pypi.python.org/pypi](https://pypi.python.org/pypi)
  - 500k projects
  - outil pip, conda
- Real Python tutorials: <https://realpython.com/>





# Environnements Distributions

- Python idle
- IPython : projet SciPy `IP[y]:`
- IDE: PyCharm / VS Code / Spyder
- Jupyter Lab|Notebook



- Distribution Anaconda, Miniconda, Miniforge

# Syntaxe du Langage

- Variable
- Base Types
- Block
- Condition
- Loop
- Comprehension
- With

# Types de données de base

<b>bool</b>	<b>True, False</b>	<b>None, 0, 0., [], (), {}, ...</b>
int	3, -3, 0b1001, 0o675, 0x3F1_000_000_000	int32/int64 (python 2) infini (python3)
long	9223372036854775808, 4L	la suite des int (python 2)
float	4., 1.5, -7.6E-123 float('nan'), float('inf')	IEEE 754 simple/double
complex	3+4j	
str	'Toto', "Titi"	
datetime.time, datetime.date, datetime.datetime	date(2017,11,20)	
decimal.Decimal	Decimal('1')/Decimal('3')	virgule fixe
fractions.Fraction	Fraction('1/3')	
NoneType	None	

# Opérations

- Booleans : or, and, not
- Comparisons : ==, !=, <, <=, >=, is, is not, in, not in  
a is None  
3 in [1, 2, 3]
- Numbers : +, -, \*, /, //, %, \*\*
- Matrix : @
- Bitwise : |, &, ^, ~, <<, >>
- Acces (index, key, slice) : [ ]  
s[0], s[-1]  
s[3:12]  
s[3:12:2]

# Operators and functions

- Logical :
  - or, and, not
  - <, <=, >, >=, ==, !=, is, is not
- Mathematical :
  - + , - , \* , / , // , % , \*\* , += , -= , \*= , /= , %=
  - functions Built-In : float, int, long, abs, cmp, min, max, sum
  - module math (floor, sqrt, cos, pi, e, ...)
- Strings :
  - +
  - functions Built-In : len, str, repr, cmp
  - methods : join, upper, lower, index, ...
  - slices

# Structures de contrôle

- if elif else
  - pas de case (jusqu'à 3.9)
- for in
  - « foreach » over all iterable object
  - for i « old school » : range, enumerate
- while
  - no do while
- comprehension : list, dict, generator
- with
  - open/close resource
- match case (python 3.10): <https://peps.python.org/pep-0636/>

# Functions

- Definition
  - `def f(x):`  
    `return x + 1`
  - `lambda x: x+1`
  - 2<sup>e</sup> order : `map`, `iter`, `all`, `any`, `filter`
- Argument
  - position or keyword
  - `var argos: tuple (*) / dict (**)`
- Return value / `None`
- Scope of variables
- Built-in functions

# Objets Standards

- Strings
- Lists
- Dictionnaires
- Tuples
- Generators



# Sequences et Dictionnaires

- Listes : list  
[1,2,3], [3], [], [[1,2,3], [4,5,6]]
- Tuples : tuple  
1,2,3, (1,2,3), (1,), ()
- Sets : set  
{1,2}
- Dictionnaires : dict  
{'Pau':64, 'Toulouse':31}
- Operators : + et [] (acces or slice)

# Iterable/Iterator/Generator

- Un itérateur permet de parcourir une donnée complexe
  - Built-In fonction `next()`
- Un objet itérable renvoie un itérateur sur lui-même
  - Built-In fonction `iter()`
- Permet un parcours avec une boucle, une comprehension list

```
spam = ['eggs1', 'eggs2', 'eggs3']
for item in spam:
    print item
```
- Un générateur fournit des valeurs à la demande
  - Faible coût mémoire
  - Un générateur est itérable
  - Implémentation avec `yield` et `yield from (*)`
  - Exemple : `range(10)`

(\*) python 3 uniquement

# Package/Module

- Déclaration et structure
  - module = fichier python (.py)
  - package = répertoire avec un fichier `__init__.py`
- Convention de nommage
- Opérations sur les modules

# Programmation Orientée Objets

- Concepts de la POO
- Membres d'instances et de classes
- Méthodes spéciales
- Encapsulation

# Librairies Communes

- Système / processus : sys
- Système de fichiers : os.path, pathlib, glob
- Expressions régulières : re
- Base de données : PEP249

# Gestion des Fichiers

- Ouverture/fermeture de fichiers
- Lecture/Ecriture
- Informations sur les fichiers
- Gestion des répertoires

# Environnement IPython

IP[y]:

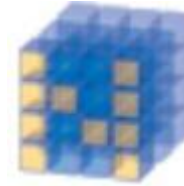
- <http://ipython.org>
- Shell python
- Interactivité ++
- Aide
- Complétion automatique

# Environnement Jupyter

- Environnement Web
  - notebook
  - lab
- Conserver code et résultats
- Graphiques
- Article scientifique



# NumPy



- <http://www.numpy.org/>
- Types NumPy :
- N-dimensionnal array + matrix
  - Broadcasting
- Algèbre Linéaire
- FFT
- Finances
- Input/Output
- Polynomes
- Tris
- Statistiques

# Types Numpy

- <https://docs.scipy.org/doc/numpy-1.13.0/user/basics.types.html>
- Taille + Signe
- Exemple : int8, uint32
- Entiers
- Flottants
- Complexes

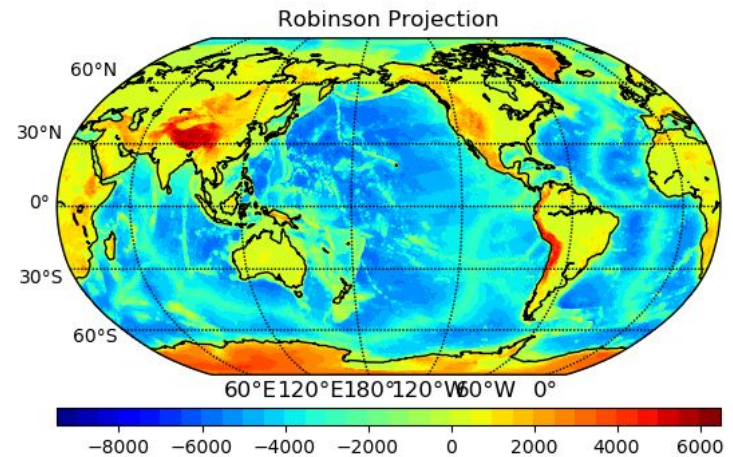
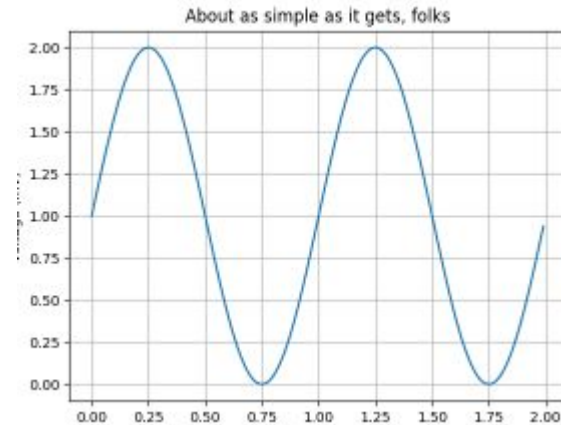
Matrices : C ou Fortran contiguous

Exemple: mode C, phénomène cache

12	23	45	...
33	55	77	...
...			
77	89	11	...

# Matplotlib & Basemap

- <https://matplotlib.org/>
- `matplotlib.pyplot`
- `mpl_toolkits.basemap`



# Matplotlib & co

- <https://plotly.com/>
- <https://docs.bokeh.org/en/latest/>
- <https://panel.holoviz.org/>
- <https://seaborn.pydata.org/>
- <https://dash.gallery/Portal/>
- 



# Pandas

- <https://pandas.pydata.org/>
- Entrées/sorties multi-format
- Nettoyage, conversion
- Transformation
- Passerelles vers numpy et matplotlib

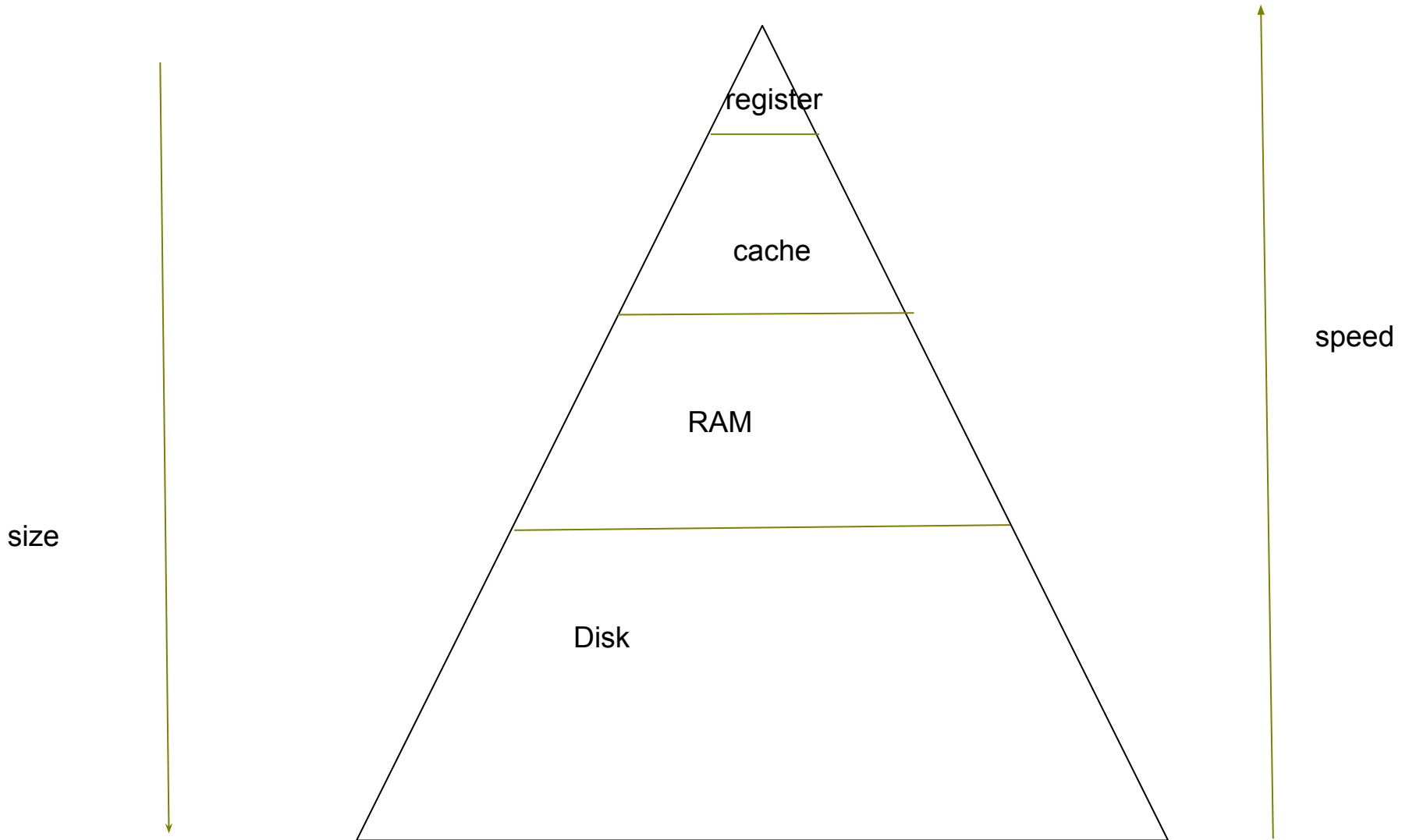
# Encodage

- ascii : 128 caractères (1 bit de contrôle)
- 1 caractère = 1 octet (char du langage C)
  - latin-1 (ISO8859-1) : europe occidentale
  - latin-5 (ISO8859-5) : cyrillique
  - ...
- 1 caractère = 1 octet avec l'€ (europe occid.)
  - latin-15 (ISO8859-15)
  - CP1252/ANSI : Microsoft
- Unicode : 3 encodages
  - UTF-8
  - UTF-16
  - UTF-32

# Décorateur

- <https://realpython.com/primer-on-python-decorators/>
- Quoi décorer
  - fonction
  - classe
- Principe : wrapper ce qu'on décore
- Exemples:
  - `@total_ordering`
  - `@dataclass`
  - `@property`
  - `@lru_cache`





# Virtual Environments

- venv (included in python)
- virtualenv
- conda (anaconda, miniconda)
- poetry

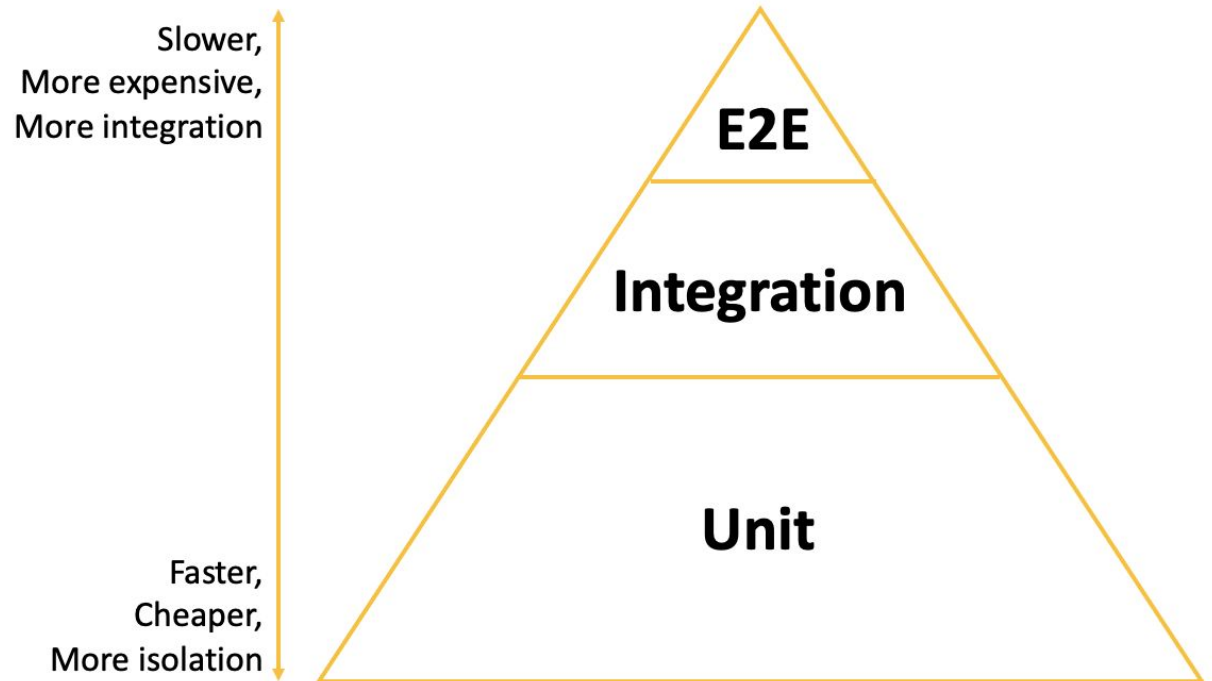
# Build

<https://packaging.python.org/en/latest/overview/>

- A lot of possibilities
- Formats
  - tar, zip, git (python only)
  - wheel (for pip)
  - binary
- Quid project
  - mixte: python/C++
  - python => C/C++
  - JIT with numba (<https://numba.pydata.org/>)
- Dependencies: requirements.txt
- Organisation
  - setup.py (old) + setuptools
  - pyproject.toml (modern)
    - PEP 517/518, 621(<https://peps.python.org/pep-0000/>),<sup>35</sup>
    - Tuto : <https://packaging.python.org/en/latest/tutorials/packaging-projects/>

# Tests en Python

- python: unittest
- tiers:
  - pytest
  - nose
  - ...



# Pytest

- `pytest.org`
  - `pip install pytest pytest-mock`
- Run
  - Scan all project
    - `pytest`
  - Run one test file:
    - `pytest test_magic_square.py`
  - Run one package test:
    - `pytest test_somepackage`
  - Run all tests with name containing pattern
    - `pytest -k is_magic_square_all_present`

# Test links

- <https://realpython.com/tutorials/testing/>
- <https://realpython.com/pytest-python-testing/>
-

# Files

- builtin function open
- libraries
  - pathlib (object mode)
  - os.path (text mode)
  - a lot more
    - csv
    - json
    - xml.etree
      - lxml (<https://lxml.de/>)
    - BeautifulSoup (html/xml)
      - <https://www.crummy.com/software/BeautifulSoup/>
    - pandas: <https://pandas.pydata.org/>

# ORM

- Object Relational Mapper
  - class Movie  $\leftrightarrow$  table movies
  - attribute title  $\leftrightarrow$  column title
  - associations
  - object Movie  $\leftrightarrow$  row in table movies
- Queries
  - insert/update/delete: object
  - select with object vocabulary  $\Rightarrow$  objects Movie
- Python:
  - ORMs: SQLAlchemy, Django ORM
  - Pandas: use sqlalchemy



# IHM / GUI

- for tcl/tk: tkinter (inside python)
- for Qt:
  - Qt for Python aka PySide2 (official)
  - PyQt

# Type checking

- <https://realpython.com/python-type-checking/>
- Hints introduced by python 3.6
  - type annotation
  - module typing, `numpy.typing`
- Advantages:
  - documentation
  - code auto completion
- checker: linter, mypy, ...

# Concurrent Programming

1. Multi Processing
  - a. multiprocessing, shared\_memory
2. Multi Threading
  - a. threading (attention si trop de threads)
3. Executor/pool (thread or process)
4. Asynchronous programming
  - a. async keyword (python 3.6)
    - i. Ex: fastapi framework
  - b. module asyncio
  - c. module celery (with flask/django)

# Online Resources

**Python for Data Analysis, 3E**

<https://wesmckinney.com/book/>

# Nouveautés Python

- 3.6:
  - formatted strings
  - hints
    - `x: int = 3`
  - `async`
- 3.10
  - `match ... case`
- 3.12
  - improve generics hints, f-strings
  - `@override`