Python 101

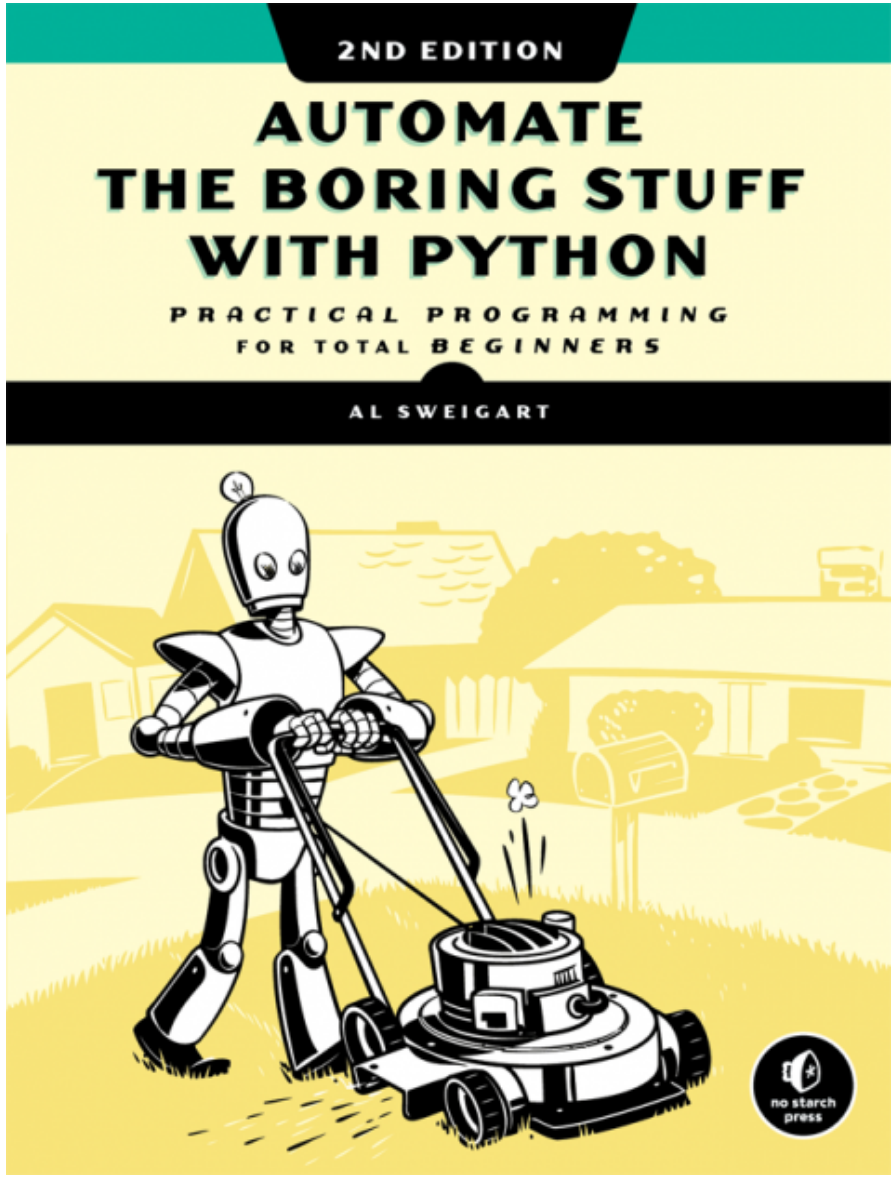

Introduction

Cédric Joly

- Chief Technology Officer @Rentacloud + Consultant @Praeludo
- Executive MBA @École de Guerre Économique
- Chargé de mission > UX Team Leader > Responsable Relations Écoles > Responsable Innovation & Expérience @Outscale
- Chef de projet @Pilot Systems
- { BTS, Licence Pro, Cycle ingénieur } Admin sys. et réseaux

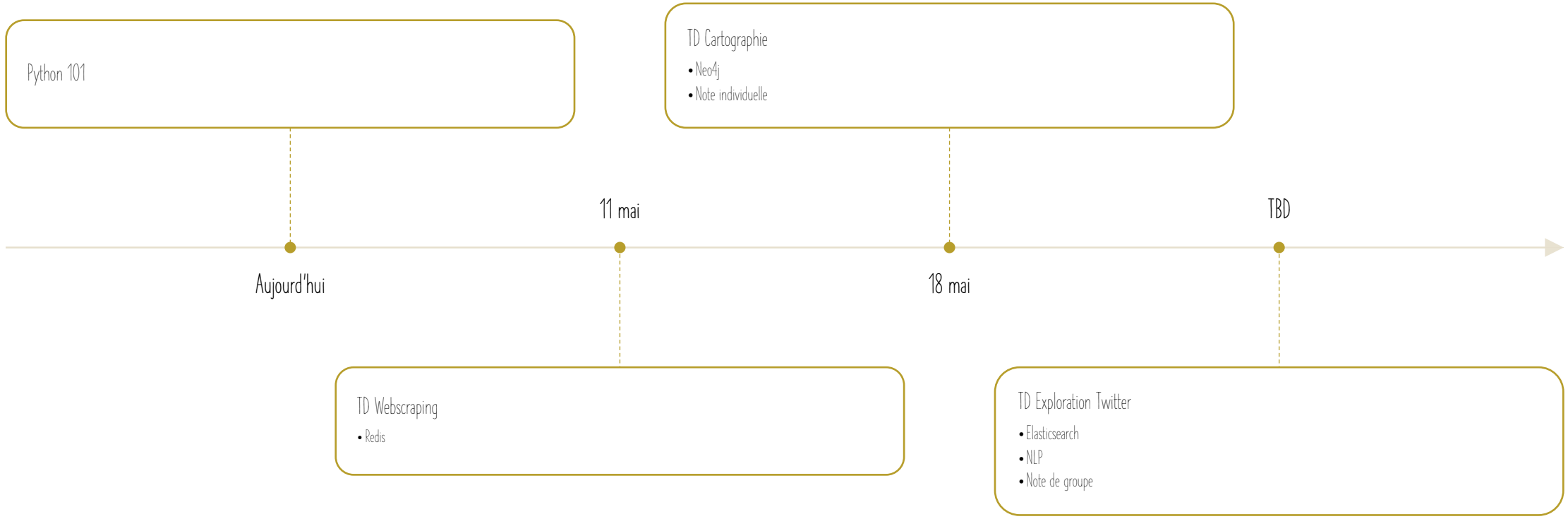
Objectif du module

- Savoir programmer est essentiel
  - programmer =! développer
- Python est votre ami
  - en cas de problème*, invoquez-le



* en cas de problème *qui peut être résolu par un ordinateur* ; feu, inondation ou accident nucléaire non couverts.

Programme du module



Programme de la session

Bases et concepts

Gestion des paquets, environnements virtuels et containers

Culture Pythonique

Pourquoi Python ?

Zen of Python

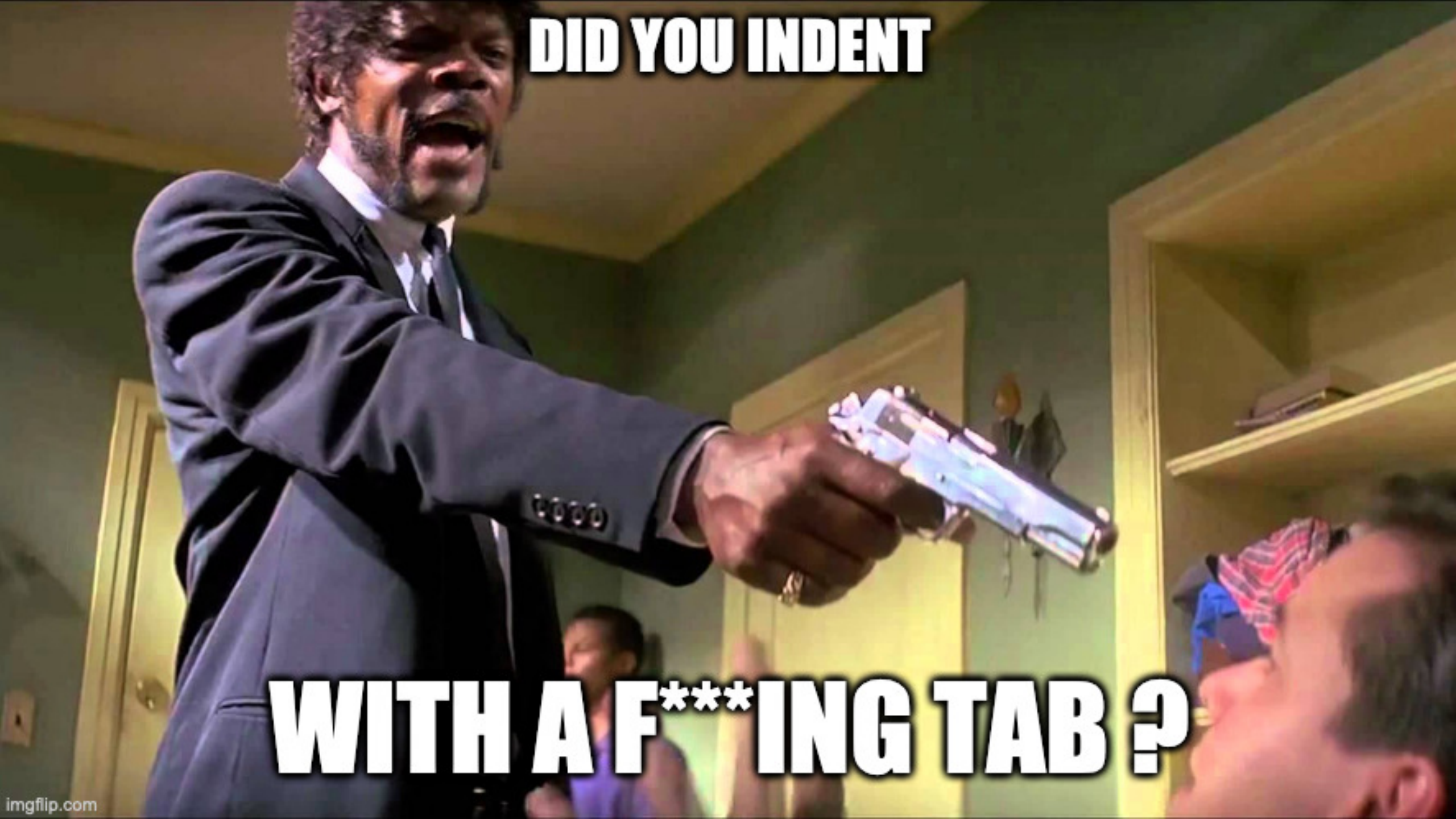
```
python -c "import this"
```

Popularité

- https://www.jetbrains.com/lp/devecosystem-2020/
- https://madnight.github.io/githut/#/pull_requests/2021/1
- https://insights.stackoverflow.com/trends?tags=c%23%2Cjava%2Cjavascript%2Cpython%2Cc%2B%2B%2Clua%2Cc%2Cruby%2Clisp%2Crust%2Cphp%2Cgo
- https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages
- https://www.tiobe.com/tiobe-index/

Polyvalent

- Scripting
- Développement Web
- Data Science
- Intelligence Artificielle
- Computer Vision & Natural Language Processing
- IoT & robotique
- GUI
- Développement de jeux


DID YOU INDENT
WITH A F***ING TAB ?
imgflip.com

Historique Python

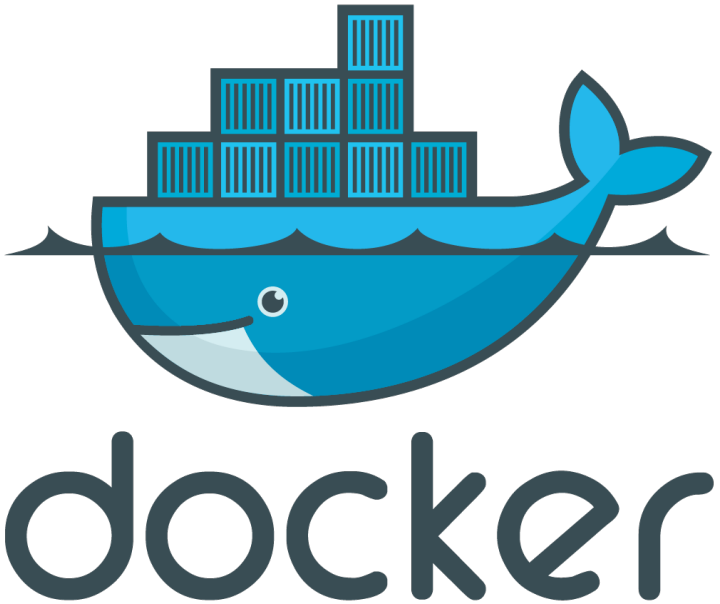
- 1991 : 0.9
- 1994 : 1.0
- 2000 : 2.0
- 2008 : 2.6, 3.0
- 2010 : 2.7
- 2016 : 3.6
- 01/01/2020 : 2.7 EOL (2.7.18)
- 10/2020 : 3.9 (stable actuelle)
- 10/2021 : 3.10 (version en développement)
- https://docs.python.org/3/whatsnew/index.html

Évolutions et nouveautés

- 2.X => 3.X
  - print => print()
  - str = ASCII => str = Unicode
  - / => //
  - xrange() => range()
- https://docs.python.org/release/3.9.0/whatsnew/3.9.html
  - 3.9 : dict union |
  - 3.8 : Walrus operator :=
  - 3.7 : Dataclasses, asyncio
  - 3.6 : f-strings
  - 3.5 : Type hints

Interlude Docker

- `docker run <options> <image>:<tag> <command>`
- https://hub.docker.com/_/python



Retour vers le futur

- https://docs.python.org/3/library/__future__.html
- `python3 -c "print(2/3)"`
- `docker run --rm python:2-slim python -c "print(2/3)"`
- `docker run --rm -ti python:2-slim python`
  - `>>> print 2/3.0`
  - `>>> from __future__ import division`
  - `>>> print 2/3`

PEPs

- Python Enhancement Proposals
  - https://www.python.org/dev/peps/
- PEP 20 : Zen of Python
- PEP 8 : Style Guide for Python Code
  - https://www.python.org/dev/peps/pep-0008/
- PEP 257 : Docstring Conventions
  - https://www.python.org/dev/peps/pep-0257/

Variables

- typage dynamique
- `type()`
- assignation avec `=`
- conventions de nommage (cf. PEP8)
  - minuscules_et_underscores_et_chiffres
    - mais pas au début les chiffres
  - explicite > implicite

Chaînes de caractères (aka Strings)

- https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str
- délimité par " " ou ' ' , """ """ ou ''' '''
- len()
- +
- slices : [début (inclus):fin (exclus):pas]
- lower(), upper(), capitalize(), join()
- startswith(), endswith()
- *
- import unidecode

Format de chaîne de caractères

```
from random import randint

acronym = "Greatest Innovation Team, Motherf******"

gitm_count = lambda x: randint(39, 43)

message = "GITM are %s and they are %d" % (acronym, gitm_count(_))

message = "GITM are the %d members of the %s" % (acronym, gitm_count(_))

message = "GITM are the {1} members of the {0}".format(acronym, gitm_count(_))

message = f"GITM are the {gitm_count(_)} members of the {acronym}"
```

À vos claviers

- Passez d' « ING1 timide » à « GITM » (avec des slices)
- À l'EPITA il y a des ING1 timides => À l'EPITA il y a des GITM

environnement

- https://docs.python.org/3/library/venv.html?highlight=venv#module-venv
- python -m venv .env
- source .env/bin/activate
- pypi : https://pypi.org/
- pip : https://pip.pypa.io/en/stable/reference/pip_install/
- python -m pip install unidecode
- pip install -r requirements.txt
- pipenv : https://pipenv.pypa.io/en/latest/
- poetry : https://python-poetry.org/

Interlude Docker (2)

- Dockerfile - https://docs.docker.com/engine/reference/builder/

FROM python:3-slim

RUN pip install unidecode

CMD [« python »]

- `docker build -t python_unidecode .`
- `docker run --rm -ti --name unidec python_unidecode python`
- `docker images`
- `docker ps`

~~Mecs Digitaux~~ Types Numériques

- https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex
- int, float, complex
- +, -, *, /, //, **
- 0.1 + 0.2
  - round(0.1 + 0.2, 1)
- import math
  - https://docs.python.org/3.9/library/math.html

Liste des modules de base

- https://docs.python.org/3/py-modindex.html

list, tuple, range

- https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range
- https://docs.python.org/3/tutorial/datastructures.html
- list() => [ ]
- tuple => ( )
- range()

Générateurs

- yield
- https://realpython.com/introduction-to-python-generators/

Structure de contrôle

- https://docs.python.org/3/reference/compound_stmts.html
- if <condition> :
  - elif / else :
- while <condition>:
- for <element> in <elements>:
- comprehension
  - [i**2 for i in range(-1, 5)]
  - (i**2 for i in range(-1, 5))
  - { i : i**2 for i in range(-1, 5) }
  - { i**2 for i in range(-1, 5) }
- https://docs.python.org/3/library/itertools.html

fonctions

- len
- type(len)
- len = « Don't do this »
- len
- del len
- len

fonctions

- def
- annotations https://realpython.com/python-type-checking/#annotations
- lambda
- builtins https://docs.python.org/3.9/library/functions.html

Ave Cæsar, pythonuri te salutant !

- Traduisez ce texte
- "Svyzxl.hyypjh'sh'mpu3'plsz'z.ðjypïylu{'lu'jovlly'A')W\x80{ovu3'NP[T'♨'()5'Wlpz'plsz'shwpkïylu{'s.hl{lly'kl'jl'qll'kl'tv{'ovu{ll\x7f5"
- Pro tips:
  - pas de 7
  - builtins

dict, set

- dict() => { }
- set() => { }

dict, set

dict() => {'key': 'value'}

set() => {'unique_value', 'a_different_unique_value'}

Plus loin sur les Data structures

- *args, **kwargs
- from collections import Counter
    - https://docs.python.org/3.9/library/collections.html
- https://realpython.com/python-data-structures/

fichier exécutable

- .py
- __init__.py
- if __name__ == "__main__"
- sys.argv
- ArgParse
  - https://docs.python.org/3/library/argparse.html

À vos claviers

- Créez une générateur de mots de passe aléatoire :
  - prenant en paramètres :
    - un nombre total de caractères (obligatoire)
    - un nombre minimum de caractères spéciaux (facultatif)
- Créez une fonction calculant la « force » d'un mot de passe
  - https://www.ssi.gouv.fr/administration/precautions-elementaires/calculer-la-force-dun-mot-de-passe/
  - doit déterminer automatiquement la taille de l'alphabet utilisé
- Appelez depuis la ligne de commande le script contenant vos fonctions
  - paramètres :
    - nombre de mots de passe à générer
    - afficher la force ou non
  - affiche la liste des mots de passe et leur force le cas échéant
- Utilisez la bibliothèque string : https://docs.python.org/fr/3/library/string.html

```
# python gen_pwd.py --n 3 --show-force true
azerty 23.50
aZ05qe8DSsqF 71.45
a(azA87ddaz2-.dzaF/V:rL.d<= 176.97
```

exceptions

- https://docs.python.org/3/library/exceptions.html
- try
- except
- raise
- finally

objet

- class
  - __init__()
  - __repr__()
  - self
- @property
- @x.setter
- héritage
  - __super__()
- dataclass
  - https://docs.python.org/3/library/dataclasses.html
  - https://realpython.com/python-data-classes/

fichiers

- open(), close()
- read(), readline(), readlines()
- with

Matière

- nom
- competences
- duree_en_heures
- moyenne_a_obtenir

Majeure

- nom
- matieres

- Etudiant
  - nom, prenom
  - adresse_mail (généré automatiquement)
  - majeure
  - save()
    - sauvegarde dans un fichier les informations sur l'étudiants
  - matieres_validees()
    - non implementé

EtudiantGITM

triade
matieres_validees()
valider_compétence(competence)
competences_a_valider

EtudiantSCIA

groupe
matieres_validees()
noter_matiere(matiere, [notes])
moyenne

```
python = Matiere(« Python », [« programmer»,
« indenter », « sauver le monde »], 12, 10)
maths = Matiere(« Mathématiques », 36, 14)
planete_solidaire = Matiere(« Planete Solidaire »,
[« gérer des sherpas », « faire des cookies »,
« sauver le monde », 800]
gitm = Majeure(« GITM », [python,
planete_solidaire])
scia = Majeure(« SCIA », [python, maths])
```

```
marcel = Etudiant(« Marcel », « Cesson », gitm, « Dai Huen Jai »)
marcel.valider_competences([« sauver le monde », « programme »,
 « indenter »])
marcel.competences_a_valider
=> « gérer des sherpas », « faire des cookies »
marcel.save()
=> marcel, cesson, GITM, Python
```

Scripting & Scraping

- https://docs.python.org/3/library/os.html
- https://docs.python.org/3/library/os.path.html
- Requests : https://2.python-requests.org/en/master/
- urllib.parse : https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse
- BeautifulSoup : https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Selenium : https://selenium-python.readthedocs.io/
- Scrapy : https://scrapy.org/

Développement Web

- Django : https://www.djangoproject.com/
- Flask : https://palletsprojects.com/p/flask
- FastAPI : https://fastapi.tiangolo.com/
- Tornado : https://www.tornadoweb.org/en/stable/
- Pyramid : https://trypyramid.com/
- Bottle : https://bottlepy.org/docs/dev/
- CherryPy : https://cherrypy.org/
- Sanic : https://sanicframework.org/en/

Data Science

- Jupyter : https://jupyter.org/
- NumPy : https://numpy.org/
- Matplotlib : https://matplotlib.org/
- Pandas : https://pandas.pydata.org/
- Anaconda : https://www.anaconda.com/products/individual
- Dash : https://dash.plotly.com/
- Dask : https://dask.org/
- Bokeh : https://bokeh.org/
- RAPIDS : https://rapids.ai/

Intelligence Artificielle

- PyTorch : https://pytorch.org/
- TensorFlow : https://www.tensorflow.org/
  - Keras : https://keras.io/
- SciKit Learn : https://scikit-learn.org/stable/
- Shogun : https://shogun-toolbox.org/

Computer Vision & Natural Language Processing

- OpenCV : https://opencv.org/
  - Tutoriel : https://www.pyimagesearch.com/
- SpaCy : https://spacy.io/
- NLTK : https://www.nltk.org/
- Gensim : https://radimrehurek.com/gensim/

IoT & robotique

- GPIOZero : https://gpiozero.readthedocs.io/en/stable/
- PyFirmata : https://github.com/tino/pyFirmata
- mraa : http://iotdk.intel.com/docs/master/mraa/python/

Développement de jeux & GUI

- PyGames : https://www.pygame.org/news
- Panda3D : https://www.panda3d.org/
- Pyglet : http://pyglet.org/
- Cocos2d : http://www.cocos2d.org/
- Arcade : https://arcade.academy/
- Kivy : https://kivy.org/#home
- wxPython : https://wxpython.org/
- tkinter : https://docs.python.org/fr/3/library/tkinter.html
- Qt for Python : https://doc.qt.io/qtforpython/
- Blender : https://docs.blender.org/api/current/index.html

Jeu + IA (sinon vous pouvez faire du LUA)

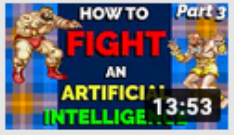
- OpenAI Gym Retro : https://retro.readthedocs.io/en/latest/
- https://youtu.be/kAis2mjr8hQ?list=PLTWFMbPFsvz23E5x70c0fcWQLtcRmNmuB&t=198

1 

How to FIGHT an AI - P1 - Rendering Open-Ai Retro in PyGame
Lucas Thompson

2 

How to FIGHT an AI - P2 - Keyboard and Joysticks
Lucas Thompson

3 

How to FIGHT an AI - P3 - Loading and fighting an AI
Lucas Thompson

4 

How to FIGHT an AI - P4 - Two AI's Fight To The Death
Lucas Thompson

Ressources complémentaires

- Documentation Python : https://docs.python.org/3/
- Real Python : https://realpython.com/
- PyCoder's Weekly : https://pycoders.com/
- PowerfulPython : https://powerfulpython.com/newsletter/
- PyBites : https://pybit.es/