Oubliez flake8, black et venv : place aux outils de nouvelle génération

1 - Qui suis-je ?

1.1 - vpoulailleau

Vincent Poulailleau

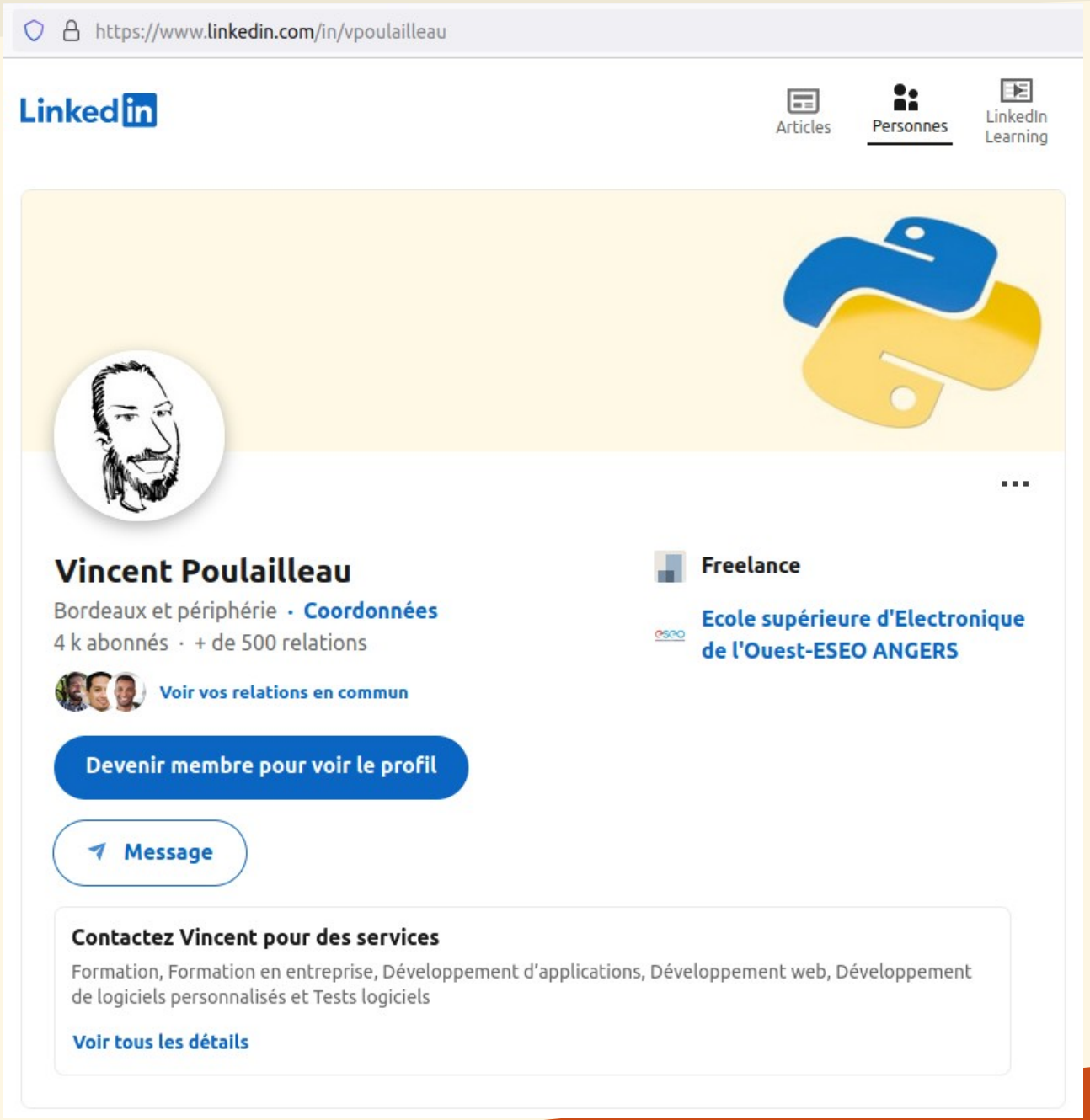
Passionné de Python

Formateur (Python, C, git, OpenGL…)

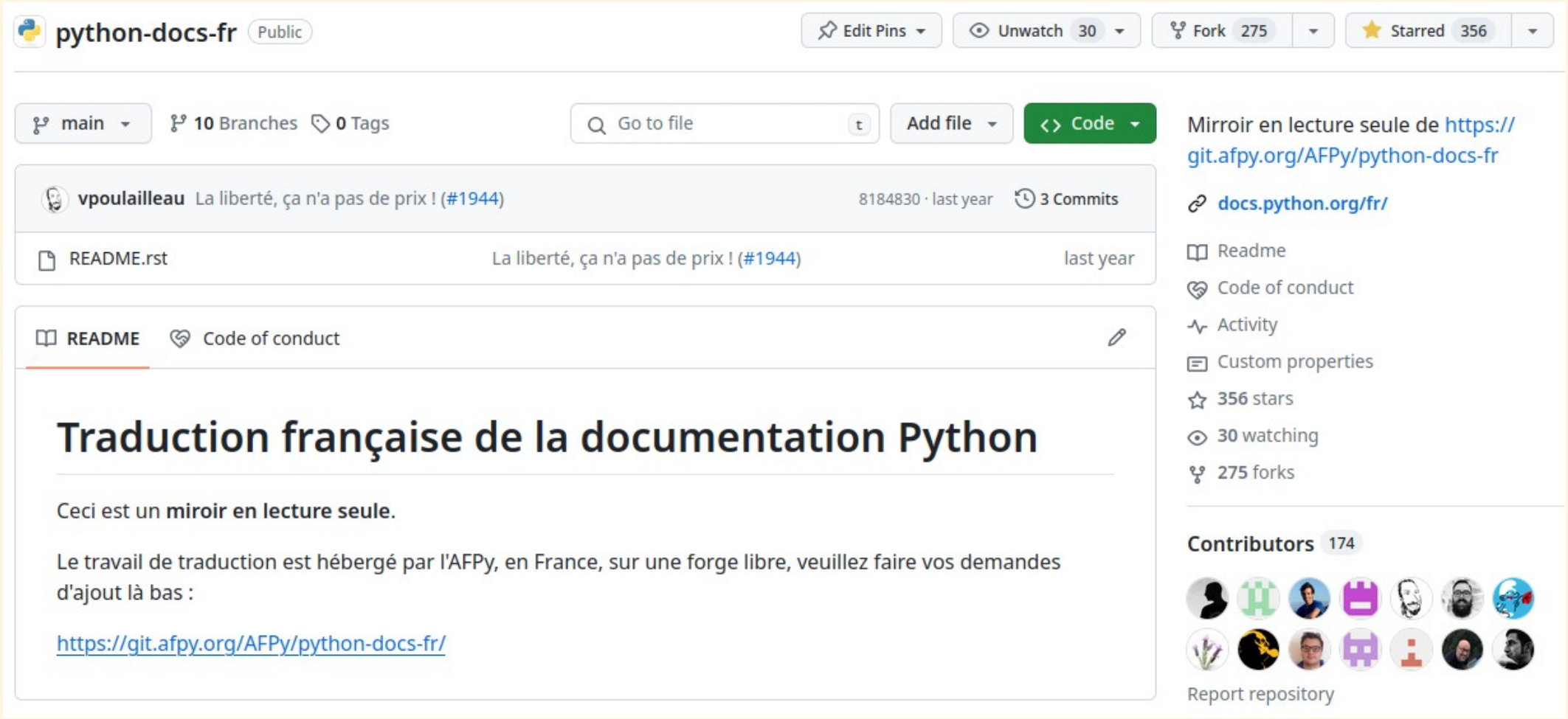
Freelance

`vpoulailleau` sur internet

1.2 - LinkedIn



1.3 - PyCon FR



L'essentiel : Qui suis-je ?

Vincent Poulailleau

vpoulailleau@gmail.com

https://www.linkedin.com/in/vpoulailleau/

https://www.pycon.fr

2 - À l'ancienne

2.1 - Environnement virtuel

2.1.1 - Environnement virtuel

Python peut installer un package

```
pip install ze_super_package
```

```
python -m pip install ze_super_package
```

Pas plus d'une version à la fois

Problème :

Mon projet 1 utilise `fastapi==0.110.2`

Mon projet 2 utilise `fastapi==0.95`

2.1.2 - Environnement virtuel

Solution : avoir plusieurs installations de Python

- Une par projet, chacun avec ses dépendances
- Installation légère ⇒ environnement virtuel

2.1.3 - Environnement virtuel

Création : `python3.12 -m venv nom_du_venv`

`nom_du_venv` est classiquement `venv`, `.venv`, `env` ou `.env`

Utilisation : dans un terminal

`. nom_du_venv/bin/activate` sous bash, zsh…

`nom_du_venv\Scripts\Activate.ps1` sous PowerShell

L'essentiel : Environnement virtuel

Une installation de Python par projet, avec ses dépendances

```
python3.12 -m venv nom_du_venv
```

```
. nom_du_venv/bin/activate
```
sous bash, zsh…

2.2 - Linter

2.2.1 - Linter

Petit bout de code

```
def UneFonction(UneValeur, UneListe = []):
    UneListe.append(UneValeur)
    print(sum(UneListe))


UneFonction(5, [1,2,3,4])   # affiche 15
UneFonction(3)              # affiche 3
UneFonction(10)             # affiche ?
```

2.2.2 - Linter

Dans MS Word ou LibreOffice : vaguelettes rouges si erreur

Un linter est un outil qui détecte les possibilités d'amélioration du code

Un linter couramment utilisé : `flake8`

```
python3.12 -m venv venv

source venv/bin/activate

pip install flake8
```

2.2.3 - Linter

Utilisation de flake8

```
$ flake8 toto.py
toto.py:1:36: E251 unexpected spaces around keyword / parameter equals
toto.py:1:38: E251 unexpected spaces around keyword / parameter equals
toto.py:5:1: E305 expected 2 blank lines after class or function definition, found 1
toto.py:5:18: E231 missing whitespace after ','
toto.py:5:20: E231 missing whitespace after ','
toto.py:5:22: E231 missing whitespace after ','
```

2.2.4 - Linter

Code corrigé

```
def UneFonction(UneValeur, UneListe=[]):
    UneListe.append(UneValeur)
    print(sum(UneListe))



UneFonction(5, [1, 2, 3, 4])   # affiche 15
UneFonction(3)                 # affiche 3
UneFonction(10)                # affiche ?
```

2.2.5 - Linter

`flake8` me dit tout va bien

Tout va bien ?

PEP 8, annotations de type, docstrings…

`flake8` a besoin de plugins

https://github.com/DmytroLitvinov/awesome-flake8-extensions

2.2.6 - Linter

```
pip install pep8-naming flake8-docstrings flake8-annotations
```

Utilisez-en plein d'autres en plus

Utilisation de flake8

```
$ flake8 toto.py
toto.py:1:1: D100 Missing docstring in public module
toto.py:1:1: D103 Missing docstring in public function
toto.py:1:6: N802 function name 'UneFonction' should be lowercase
toto.py:1:17: ANN001 Missing type annotation for function argument 'UneValeur'
toto.py:1:18: N803 argument name 'UneValeur' should be lowercase
toto.py:1:28: ANN001 Missing type annotation for function argument 'UneListe'
toto.py:1:40: ANN201 Missing return type annotation for public function
```

L'essentiel : Linter

Outil de détection des possibilités d'amélioration du code

```
pip install flake8

pip install pep8-naming flake8-
docstrings flake8-annotations...
```

2.3 - Formateur

2.3.1 - this

https://github.com/python/cpython/blob/main/Lib/this.py

this.py

```
s = """Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpvg vf orggre guna vzcyvpvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fcnefr vf orggre guna qrafr.
Ernqnovyvgl pbhagf.
Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire cnff fvyragyl.
Hayrff rkcyvpvgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)

print("".join([d.get(c, c) for c in s]))
```

2.3.2 - Lisibilité

La lisibilité ça compte

Respectez un standard de codage

En particulier la PEP 8

https://peps.python.org/pep-0008/

One of Guido's key insights is that ***code is read much more often than it is written***. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code.

2.3.3 - PEP 8

Une espace de chaque côté des opérateurs mathématiques

Deux lignes blanches après une fonction ou une classe de premier niveau

nom_de_variable, nom_de_fonction, NomDeClasse, NomDeType, NOM_DE_CONSTANTE

Une espace après les virgules

- Sauf les virgules de fin (trailing commas)

…

2.3.4 - Formateur

Un formateur de code peut améliorer le code

- Pour le respect de la PEP 8 : `black`, `autopep8`, `yapf`...
- Pour la maintenabilité : `removestar`, `pyupgrade`, `autoflake`...

2.3.5 - Avant utilisation de black

Mauvais code

```
x = {  'a':37,'b':42,

'c':927}

if very_long_variable_name is not None and \
 very_long_variable_name.field > 0 or \
 very_long_variable_name.is_debug:
 z = 'hello '+'world'
class Foo  (       ):
  def f    (self   ):
    return       37*-2
  def g(self, x,y=42):
      return y
def very_important_function(template: str,*variables,file: os.PathLike,debug:bool=False,):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
     ...
```

Utilisation : `black toto.py`

2.3.6 - Après passage de black

Meilleur code

```python
x = {"a": 37, "b": 42, "c": 927}

if (
    very_long_variable_name is not None
    and very_long_variable_name.field > 0
    or very_long_variable_name.is_debug
):
    z = "hello " + "world"


class Foo:
    def f(self):
        return 37 * -2

    def g(self, x, y=42):
        return y


def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```

L'essentiel : Formateur

Le code doit être lisible : PEP 20

Respectons le standard de codage : PEP 8

Un formateur de code peut améliorer le code en automatique

```
pip install black pyupgrade

pyupgrade --py312-plus toto.py

black toto.py
```

2.4 - Alias

2.4.1 - Alias

Beaucoup de commandes répétitives au quotidien

- Gestion d'environnements virtuels
- Linter / formateur
- `git`
- ...

2.4.2 - Alias

Un alias permet de donner un nom court à une autre commande

Sous Linux/Mac :
https://man7.org/linux/man-pages/man1/alias.1p.html

Sous Windows :
https://learn.microsoft.com/fr-fr/powershell/module/microsoft.powershell.core/about/about_aliases?view=powershell-7.4

2.4.3 - Alias

```
alias venv='python3.12 -m venv venv'

alias act='source venv/bin/activate'

alias lint='flake8'

alias format='black'
```

L'essentiel : Alias

Alias : nom court pour une autre commande

L'essentiel : À l'ancienne

§ Environnement virtuel

§ Linter

§ Formateur

§ Alias

3 - Version moderne

3.1 - Introduction

Ça avait l'air bien

- Environnements virtuels
- Linter
- Formateur

On oublie tout ça ?

- NON !!!

3.2 - Rust

3.2.1 - Rust

Langage de programmation compilé

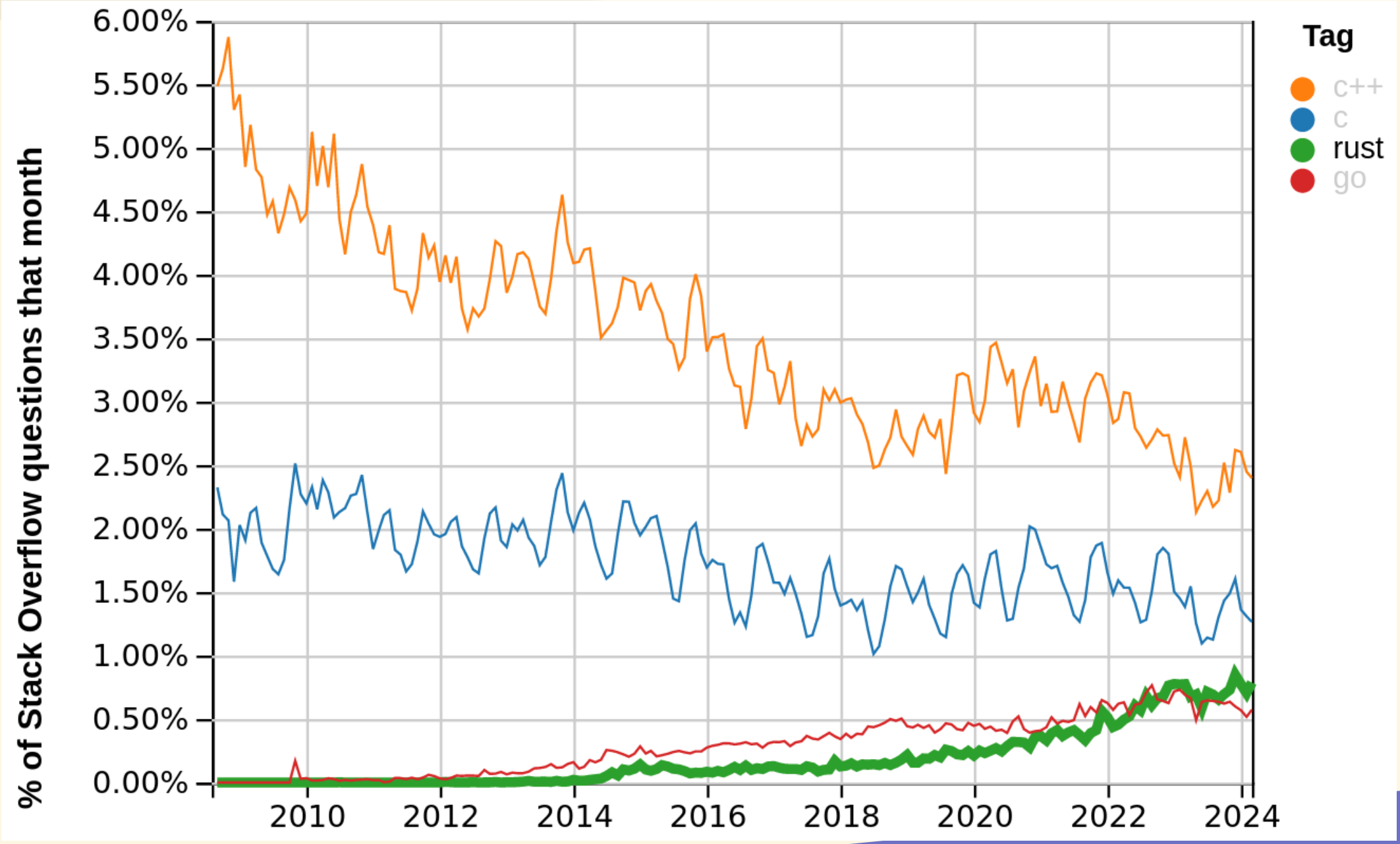
Axé performance, sûreté et concurrence

Créé en 2006

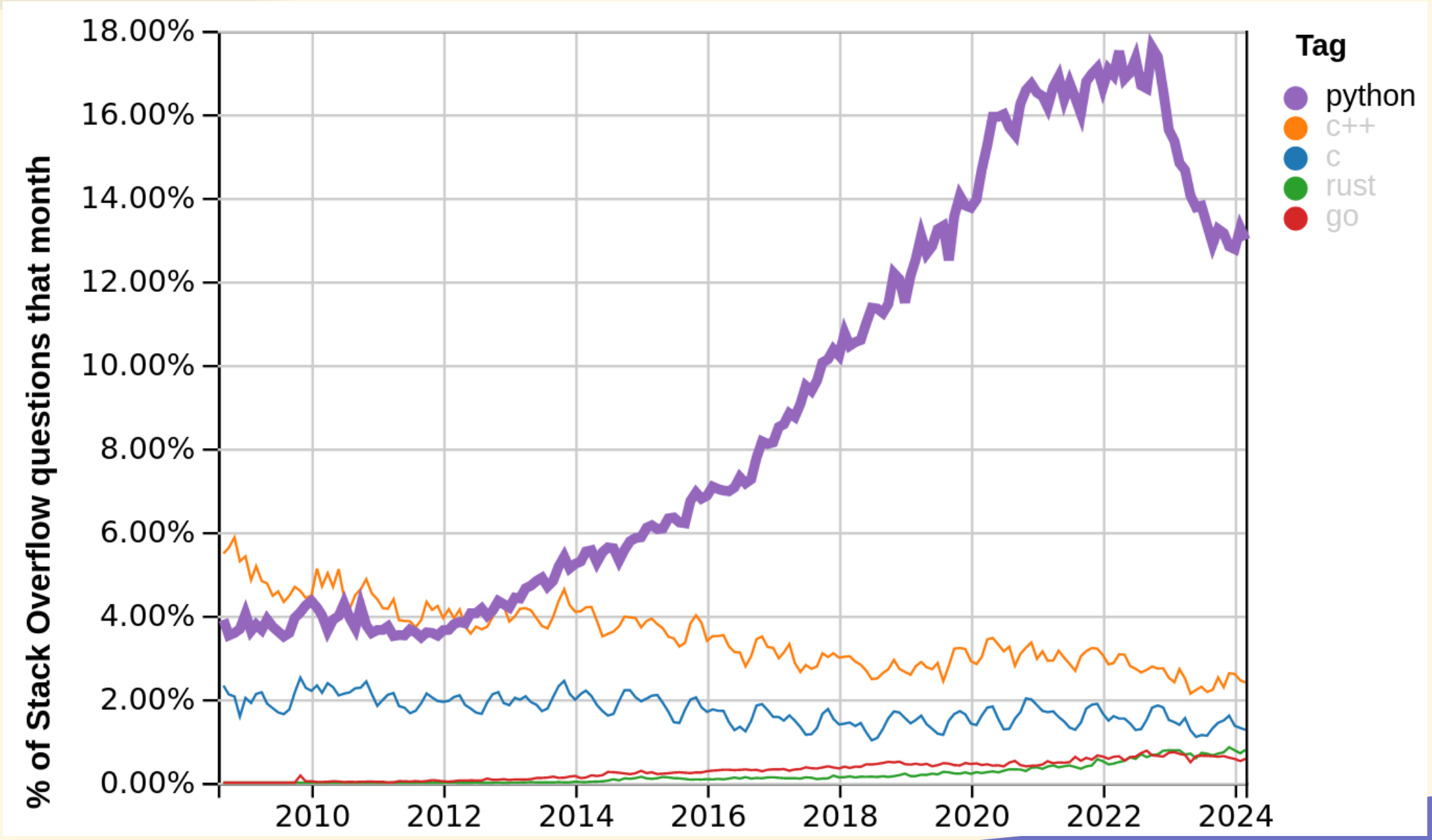
J'imagine qu'il va remplacer le C et le C++

https://survey.stackoverflow.co/2023/#section-admired-and-desired-programming-scripting-and-markup-languages

3.2.2 - Rust



3.2.3 - Rust



L'essentiel : Rust

Langage de programmation compilé, sûr, performant

J'imagine qu'il va remplacer le C et le C++

3.3 - Astral

3.3.1 - Astral

https://astral.sh/

« Next-gen Python tooling »

« High-performance developer tools for the Python ecosystem. »

Début public : août 2022

https://notes.crmarsh.com/python-tooling-could-be-much-much-faster

Utilisation de Rust pour faire des outils performants

L'essentiel : Astral

Entreprise américaine

« High-performance developer tools for the Python ecosystem. »

3.4 - uv

3.4.1 - venv + pip

Astral propose `uv` au lieu de `venv` et `pip`

3.4.2 - pip

Avant

```
pip install fastapi

pip install -r requirements.txt

pip install --editable .
```

Après

```
uv pip install fastapi

uv pip install -r requirements.txt

uv pip install --editable .
```

3.4.3 - uv pip

`uv pip` fonctionne dans les environnements virtuels classiques

Il fait les calculs de dépendances très vite

Il télécharge les packages en parallèle

Il ne télécharge pas les packages téléchargés précédemment

3.4.4 - uv venv

`uv venv` permet la création d'un environnement virtuel

Création d'environnements virtuels

```
uv venv  # crée .venv avec l'interpréteur actif

uv venv nom_du_venv  # crée nom_du_venv

uv venv -p 3.12  # utilise Python 3.12 installé sur le système
```

3.4.5 - uv venv

Activation de manière classique

- `. venv/bin/activate` par exemple

Ne fournit pas `pip` de base dans `venv/bin/`

- Utilisation de `uv pip`
- Ou `uv venv --seed`

L'essentiel : uv

`uv pip` remplace `pip`

`uv pip` fonctionne aussi dans les environnements virtuels classiques

`uv venv` permet la création d'un environnement virtuel

3.5 - ruff check

3.5.1 - ruff

Astral propose `ruff` pour remplacer `flake8` et de nombreux plugins

Plus de 800 règles vérifiées

https://docs.astral.sh/ruff/rules/

`ruff` contient un linter : `ruff check`

3.5.2 - ruff check

ruff check basique

```
$ ruff check toto.py  # le fichier toto.py de tout à l'heure

All checks passed!
```

3.5.3 - ruff check

ruff check --select "ALL"

```
$ ruff check --select "ALL" toto.py

warning: `one-blank-line-before-class` (D203) and `no-blank-line-before-
class` (D211) are incompatible. Ignoring `one-blank-line-before-class`.

warning: `multi-line-summary-first-line` (D212) and `multi-line-summary-second-
line` (D213) are incompatible. Ignoring `multi-line-summary-second-line`.

toto.py:1:1: D100 Missing docstring in public module

toto.py:1:5: N802 Function name `UneFonction` should be lowercase

toto.py:1:5: ANN201 Missing return type annotation for public function `UneFonction`

toto.py:1:5: D103 Missing docstring in public function

toto.py:1:17: N803 Argument name `UneValeur` should be lowercase

toto.py:1:17: ANN001 Missing type annotation for function argument `UneValeur`

toto.py:1:28: N803 Argument name `UneListe` should be lowercase

toto.py:1:28: ANN001 Missing type annotation for function argument `UneListe`

toto.py:1:39: B006 Do not use mutable data structures for argument defaults

toto.py:3:5: T201 `print` found

Found 10 errors.

No fixes available (3 hidden fixes can be enabled with the `--unsafe-fixes` option).
```

3.5.4 - ruff check

ruff check --preview

```
$ ruff check --select "ALL" --preview toto.py
warning: `one-blank-line-before-class` (D203) and `no-blank-line-before-
class` (D211) are incompatible. Ignoring `one-blank-line-before-class`.
warning: `multi-line-summary-first-line` (D212) and `multi-line-summary-second-
line` (D213) are incompatible. Ignoring `multi-line-summary-second-line`.
toto.py:1:1: D100 Missing docstring in public module
  |
1 | def UneFonction(UneValeur, UneListe = []):
  |  D100
2 |     UneListe.append(UneValeur)
3 |     print(sum(UneListe))
  |

toto.py:1:1: CPY001 Missing copyright notice at top of file
  |
1 | def UneFonction(UneValeur, UneListe = []):
  |  CPY001
2 |     UneListe.append(UneValeur)
3 |     print(sum(UneListe))
  |

toto.py:1:5: N802 Function name `UneFonction` should be lowercase
  |
1 | def UneFonction(UneValeur, UneListe = []):
  |     ^^^^^^^^^^ N802
2 |     UneListe.append(UneValeur)
3 |     print(sum(UneListe))
  |

# et ça continue encore et encore
```

3.5.5 - ruff check

ruff check --output-format concise

```
$ ruff check --select "ALL" --preview --output-format concise toto.py
warning: `one-blank-line-before-class` (D203) and `no-blank-line-before-
class` (D211) are incompatible. Ignoring `one-blank-line-before-class`.
warning: `multi-line-summary-first-line` (D212) and `multi-line-summary-second-
line` (D213) are incompatible. Ignoring `multi-line-summary-second-line`.
toto.py:1:1: D100 Missing docstring in public module
toto.py:1:1: CPY001 Missing copyright notice at top of file
toto.py:1:5: N802 Function name `UneFonction` should be lowercase
toto.py:1:5: ANN201 Missing return type annotation for public function `UneFonction`
toto.py:1:5: D103 Missing docstring in public function
toto.py:1:17: N803 Argument name `UneValeur` should be lowercase
toto.py:1:17: ANN001 Missing type annotation for function argument `UneValeur`
toto.py:1:28: N803 Argument name `UneListe` should be lowercase
toto.py:1:28: ANN001 Missing type annotation for function argument `UneListe`
toto.py:1:36: E251 [*] Unexpected spaces around keyword / parameter equals
toto.py:1:38: E251 [*] Unexpected spaces around keyword / parameter equals
toto.py:1:39: B006 Do not use mutable data structures for argument defaults
toto.py:3:5: T201 `print` found
toto.py:5:1: E305 [*] Expected 2 blank lines after class or function definition, found (1)
toto.py:5:18: E231 [*] Missing whitespace after ','
toto.py:5:20: E231 [*] Missing whitespace after ','
toto.py:5:22: E231 [*] Missing whitespace after ','
Found 17 errors.
[*] 6 fixable with the `--fix` option (3 hidden fixes can be enabled with the `--unsafe-fixes` option).
```

3.5.6 - ruff check

ruff check --fix

```
ruff check --select "ALL" --preview --output-format concise --fix toto.py

warning: `one-blank-line-before-class` (D203) and `no-blank-line-before-
class` (D211) are incompatible. Ignoring `one-blank-line-before-class`.

warning: `multi-line-summary-first-line` (D212) and `multi-line-summary-second-
line` (D213) are incompatible. Ignoring `multi-line-summary-second-line`.

toto.py:1:1: D100 Missing docstring in public module

toto.py:1:1: CPY001 Missing copyright notice at top of file

toto.py:1:5: N802 Function name `UneFonction` should be lowercase

toto.py:1:5: ANN201 Missing return type annotation for public function `UneFonction`

toto.py:1:5: D103 Missing docstring in public function

toto.py:1:17: N803 Argument name `UneValeur` should be lowercase

toto.py:1:17: ANN001 Missing type annotation for function argument `UneValeur`

toto.py:1:28: N803 Argument name `UneListe` should be lowercase

toto.py:1:28: ANN001 Missing type annotation for function argument `UneListe`

toto.py:1:37: B006 Do not use mutable data structures for argument defaults

toto.py:3:5: T201 `print` found

Found 17 errors (6 fixed, 11 remaining).

No fixes available (3 hidden fixes can be enabled with the `--unsafe-fixes` option).
```

3.5.7 - ruff check

Configurable avec un fichier `pyproject.toml`, `ruff.toml`, `.ruff.toml`

pyproject.toml

```toml
[tool.ruff]
line-length = 88
indent-width = 4
target-version = "py312"  # Python 3.12

[tool.ruff.lint]
select = ["ALL"]
ignore = ["E501", "B"]  # ignore specific rules

# Ignore `E402` (import violations) in all `__init__.py`
# files, and in select subdirectories.
[tool.ruff.lint.per-file-ignores]
"__init__.py" = ["E402"]
"**/{tests,docs,tools}/*" = ["E402"]
```

L'essentiel : ruff check

`ruff` contient un linter : `ruff check`

```
ruff check --select "ALL"
```

`ruff check --preview` pour les règles en bêta

```
ruff check --output-format concise
```

`ruff check --fix` pour les corrections automatiques

3.6 - ruff format

3.6.1 - ruff format

`ruff format` est un formateur de code

- Formate le code de manière déterministe
- Homogénéise le code au sein d'une équipe
- Surtout pour les juniors, les étourdis, les fainéants ou ceux qui délèguent à l'ordinateur
- Très compatible de `black`

3.6.2 - ruff format

```
ruff format toto.py

ruff format dossier/

ruff format # le dossier courant
```

Respecte `.gitignore`

3.6.3 - ruff format

Même fichier de configuration que `ruff check`

pyproject.toml

```
[tool.ruff]
line-length = 88


[tool.ruff.format]
quote-style = "double"
indent-style = "space"
docstring-code-format = true
```

L'essentiel : ruff format

`ruff format` est un formateur de code

Très compatible de `black`

3.7 - alias

3.7.1 - alias

Il suffit de mettre à jour ses alias

```
alias venv='uv venv -p 3.12 venv'

alias act='source venv/bin/activate'

alias pip='uv pip'

alias lint='ruff check --select "ALL" --
preview --output-format concise'

alias format='ruff format'
```

3.8 - Démo

3.8.1 - Démo

Règle n°1 : ne jamais faire de démo

Loi de Murphy

Règle n°2 : dans le doute, se référer à la règle n°1

Mais soyons fous !

L'essentiel : Version moderne

§ Rust

§ Astral

§ uv

§ ruff check

§ ruff format

§ alias

§ Démo

4 - Conclusion

4.1 - Conclusion

Utilisez des outils pour vous aider

- `uv`, `ruff check`, `ruff format`
- `alias`
- `git` ou équivalent
- `pre-commit`
- Intégration continue, déploiement continu
- Un bon IDE
- ...

Testez les outils d'Astral !

4.2 - Questions ? Slides :

https://www.lecalamar.fr/posts/2024-06-06-oubliez-flake8-black-ou-venv-place-aux-outils-de-nouvelle-generation/