Python for the Muggle-born developer

Python Rennes - 24 Novembre 2021 - Lucius Malforel

PRESENTATION & DISCLAIMER

- @Zenigwarts 2016 (green advocate 2020+)
- OWL in **Care of magical creatures -** 🐍 +3 years
- `@lucsorel`

PRESENTATION & DISCLAIMER

- @Zenigwarts 2016 (green advocate 2020+)
- OWL in **Care of magical creatures -** +3 years
- `@lucsorel`

- This is NOT an introductory talk to the language
- This is about how to ~~tame~~ *play with* the beast

Lesson 1 – Tame the Right Snake

Choose Your Dialect

First tamer: **Guido van Rossum** released Python 1 in 1991

"Code is read much more often than it is written"

- Python 3 released in 2008
- Python 2 ended in 2020

Why do I have an obsolete beast alive in my syssSStem?

```
~ python2 --version
Python 2.7.18
```

Why do I have an obsolete beast alive in my syssSStem?

```
~ python2 --version
Python 2.7.18
```

```
~ sudo apt-get purge python2
Construction de l'arbre des dépendances...
```

Why do I have an obsolete beast alive in my syssSStem?

```
~ python2 --version
Python 2.7.18
```

```
~ sudo apt-get purge python2
Construction de l'arbre des dépendances...
```

```
...
Les paquets suivants seront ENLEVÉS :
  gimp-plugin-registry* inkscape* jackd2* jackd2-firewire*
  libjack-jackd2-0* python-backports.functools-lru-cache* python-bs4*
  python-cffi-backend* python-chardet* python-cryptography*
  python-dbus* python-enum34* python-gi* python-html5lib*
  python-ipaddress* python-lxml* python-numpy* python-openssl*
  python-pkg-resources* python-six* python-soupsieve* python-tk*
  python-webencodings* python2* scribus* scribus-data*
  ubuntustudio-controls* ubuntustudio-default-settings*
  ubuntustudio-desktop* ubuntustudio-desktop-core*
  ubuntustudio-installer* ubuntustudio-menu* ubuntustudio-menu-add*
Souhaitez-vous continuer ? [O/n]
```

My Python 3 beast is also rather old...

```
~ python3 --version
Python 3.8.6
```

My Python 3 beast is also rather old...

```
~ python3 --version
Python 3.8.6
```

```
~ sudo apt-get purge python3
Les paquets suivants seront ENLEVÉS (sélection):
  apparmor* chromium-browser* firefox*
  gnome-software-plugin-snap* language-selector-common*
  printer-driver-foo2zjs-common* printer-driver-m2300w*
  [...]
  python3* [...]
  software-properties-common* software-properties-gtk*
  system-config-printer* system-config-printer-common*
  system-config-printer-udev* ubuntu-advantage-tools*
  ubuntu-drivers-common* ubuntu-minimal*
  ubuntu-release-upgrader-core* ubuntu-release-upgrader-gtk*
  ubuntu-standard* ubuntustudio-controls*
  ubuntustudio-default-settings* ubuntustudio-desktop*
  ubuntustudio-desktop-core* ubuntustudio-installer*
  ubuntustudio-menu* ubuntustudio-menu-add*
  [...]
  update-manager* update-manager-core* update-notifier*
  xfce4-panel-profiles* xfpanel-switch* xorg* xserver-xorg*

Souhaitez-vous continuer ? [O/n]
```

📜✅ Don't mess around with your python system installation

Define a 🐍 version at the project level

pyenv (🐧, 🍏), pyenv-win (🪟)

- downloads specific 🐍 versions (including `anaconda`'s)

```
~ pyenv install 3.9.3
```

- specifies the version to use at the folder scale

```
~ cd Documents
~ pyenv local 3.9.3

~ cat .python-version
3.9.3
```

- adds *shims* to your `$PATH` that:
  - intercept calls to 🐍 binaries (`python`, `pip`, etc.)
  - search for a `.python-version` file (iteratively towards `/`)

Example: `cat ~/.pyenv/shims/python`:

```
#!/usr/bin/env bash
set -e
[ -n "$PYENV_DEBUG" ] && set -x

program="${0##*/}"

export PYENV_ROOT="~/.pyenv"
exec "~/.pyenv/libexec/pyenv" exec "$program" "$@"
```

Lesson 2 - Isolate Your Snake's Environment

Room of Requirement

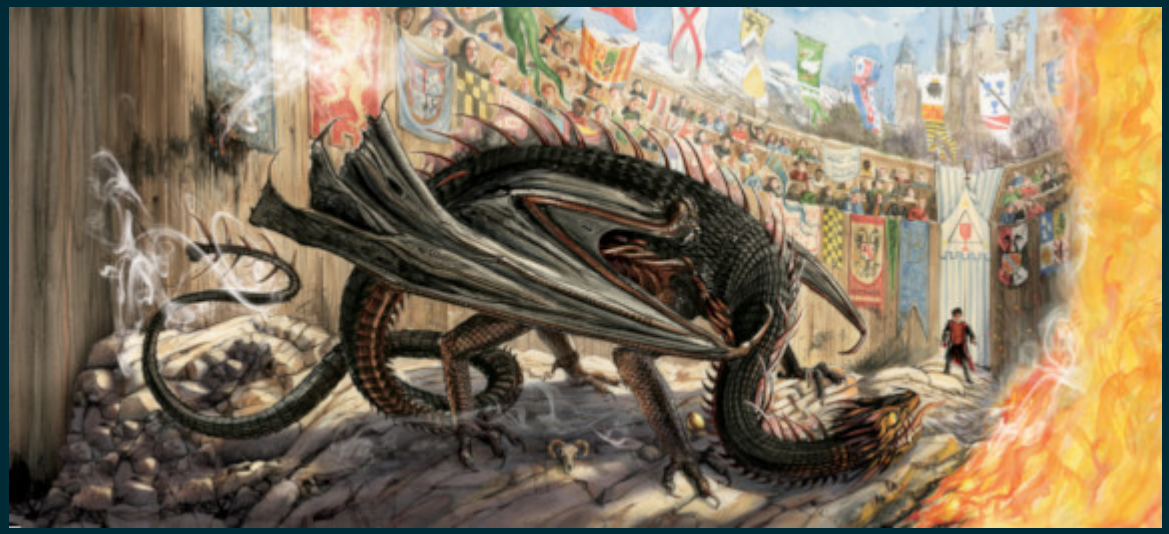
~150 "Python 3" sSSpells are already installed in my sysSStem... 😱

```
sudo apt list --installed | grep python3  | wc -l

151
```

- versions set by your (local) distribution
- your projects may involve different ones

📜✅✅ Don't mess around with your python system installation



Use a virtual environment

```
~ python -m venv my-project # .python-version thanks to pyenv, python version is now 3.x
~ cd my-project
```

Use a virtual environment

```
~ python -m venv my-project # 📜 thanks to pyenv, python version is now 3.x
~ cd my-project
```

```
my-project/
  ├ bin/       # available binaries
  │ ├ activate
  │ ├ pip
  │ └ python
  ├ lib/       # project dependencies
  └ pyvenv.cfg # env settings & paths to python binaries

~ source bin/activate
~ (my-project) ➜  pip install matplotlib
```

Use a virtual environment

```
~ python -m venv my-project # 📜 thanks to pyenv, python version is now 3.x
~ cd my-project
```

```
my-project/
  ├ bin/       # available binaries
  │ ├ activate
  │ ├ pip
  │ └ python
  ├ lib/       # project dependencies
  └ pyvenv.cfg # env settings & paths to python binaries

~ source bin/activate
~ (my-project) ➜  pip install matplotlib
```

```
~ (my-project) ➜  pip freeze > requirements.txt
```

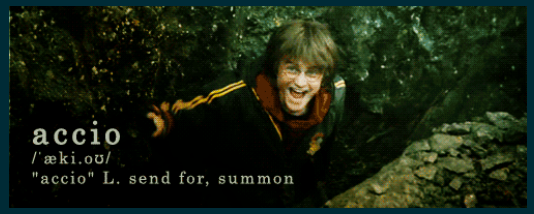
```
cycler==0.11.0
fonttools==4.28.2
kiwisolver==1.3.2
matplotlib==3.5.0
numpy==1.21.4
[...]
six==1.16.0
tomli==1.2.2
```

📜 pip (Python Package Manager)

- 🙂 huge `Python Package Index` (pypi.org)
- 🙂 `pip freeze` versions transitive dependencies too
- 😕 `pip un·install` does not update `requirements.txt`
- ☹️ does not distinguish `production` from `development` dependencies

Associate an image to a concept

A **B** **C**



1. `pip install`
2. `python -m venv`
3. `pip freeze`

LESSON 3 - BREEDING

DAILY CARE & INDUSTRIALIZATION

How should I sStructure & sSpecify my project?

Think about your code as a library:

- production code is expected in a sub-folder
- it is the package name that will be installed via PyPI
- other folders (eg `tests/`) and root contents will be discarded when installed from PyPI

How should I sSStructure & sSSpecify my project?

Think about your code as a library:

- production code is expected in a sub-folder
- it is the package name that will be installed via PyPI
- other folders (eg `tests/`) and root contents will be discarded when installed from PyPI

Project configuration root file:

- obliviate(~~`setup.py`~~): obsolete, imperative
- 💙💛 `pyproject.toml`: new standard (PEP-518, 2016), declarative

Poetry: spells for dependency management and packaging

- abstraction layer over `venv` & `pip`
- provides life-cycle commands for projects

```
~ poetry new expylliarmus
~ cd expylliarmus

# virtual environment in a `.venv` sub-folder
~ poetry config virtualenvs.create true --local
~ poetry config virtualenvs.in-project true --local

~ poetry install
```

Poetry: spells for dependency management and packaging

- abstraction layer over `venv` & `pip`
- provides life-cycle commands for projects

```
~ poetry new expylliarmus
~ cd expylliarmus

# virtual environment in a `.venv` sub-folder
~ poetry config virtualenvs.create true --local
~ poetry config virtualenvs.in-project true --local

~ poetry install
```

```
expylliarmus/          # root folder of the codebase (version its contents)
  ├ .venv/             # virtual environment folder (bin/activate, etc.)
  ├ expylliarmus/      # package production code
  │ └ __init__.py      # exposes the package main contents
  ├ tests/             # testing resources
  │ ├ __init__.py      # defines the tests package
  │ └ expylliarmus.py  # contains a basic unit test
  ├ poetry.lock        # dependencies lock file
  ├ poetry.toml        # poetry configuration
  ├ pyproject.toml     # package configuration
  └ README.rst         # empty root documentation file
```

Install some production & developement dependencies:

```
poetry add matplotlib
poetry add -D pylint
```

Install some production & developement dependencies:

```
poetry add matplotlib
poetry add -D pylint
```

Inside the `pyproject.toml` file:

```
[tool.poetry]
name = "expylliarmus"
version = "0.1.0"
description = "Put a spell on you"
# populated from your git config
authors = ["Lucius Malforel <l**.****l@z***a.c*m>"]

[tool.poetry.dependencies]
python = "^3.8"
matplotlib = "^3.5.0"

[tool.poetry.dev-dependencies]
pytest = "^5.2"
pylint = "^2.11.1"
```

See other optional keys (repo, bug-tracker, licence, etc.)

Lesson 4 - Spell-Book and Wands

Intently Dangerous Equipment

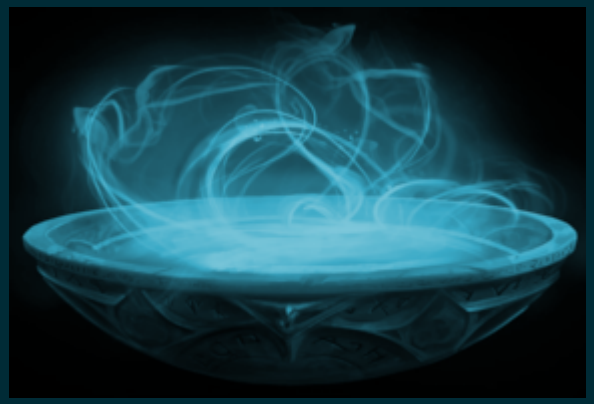
- Codium (vsCode without MS telemetry) & extensions
  - ms-python.python: Python tooling
  - bungcip.better-toml: `toml` file syntax highlighting
  - coenraads.bracket-pair-colorizer-2: brackets, braces & parentheses highlighting

- Codium (vsCode without MS telemetry) & extensions
  - ms-python.python: Python tooling
  - bungcip.better-toml: `toml` file syntax highlighting
  - coenraads.bracket-pair-colorizer-2: brackets, braces & parentheses highlighting
- Pycharm

- Codium (vsCode without MS telemetry) & extensions
  - ms-python.python: Python tooling
  - bungcip.better-toml: `toml` file syntax highlighting
  - coenraads.bracket-pair-colorizer-2: brackets, braces & parentheses highlighting
- Pycharm
- Spyder
- Sublime Text, Atom
- Vim, Emacs

Use your favorite !

Pensieve Takeaways



1. Don’t mess around with your Python system installation
2. Isolate project in a virtual environment
3. Production code inside a dedicated folder
4. Project configuration in `pyproject.toml`
5. Codium / vsCode + 3 extensions is a good start

💙💛 pyenv + poetry + pytest

THANKSSSS !

EXTRA 1 - ACCIO APPLICATION!

PORTKEYS

pyinstaller (~~installer~~ -> packager)

- distributable file containing Python binaries, imported libraries & application code

```
# package the application as a single file
poetry run pyinstaller --onefile --name="expylliarmus" \
  --specpath="$(pwd)/pyinstaller" --distpath="$(pwd)/dist"  \
  "$(pwd)/expylliarmus/__main__.py"
```

pyinstaller (~~installer~~ -> packager)

- distributable file containing Python binaries, imported libraries & application code

```
# package the application as a single file
poetry run pyinstaller --onefile --name="expylliarmus" \
  --specpath="$(pwd)/pyinstaller" --distpath="$(pwd)/dist"  \
  "$(pwd)/expylliarmus/__main__.py"
```

🐳 fully-configurable image

- install `pyenv`
- install the python version (`cat .python-version`)
- install `poetry` and app dependencies
- copy app production folder

Extra 2 - Conceal Shameful Experiments

Jupyter Horcruxes

Install & Customize Jupyter in a Image

```
docker run -p 8888:8888 -v "$(pwd)/shared:/home/jovyan/work" \
    -e JUPYTER_TOKEN=horcrux -e JUPYTER_ENABLE_LAB=yes jupyter/scipy-notebook
```

Install & Customize Jupyter in a 🐳 Image

```
docker run -p 8888:8888 -v "$(pwd)/shared:/home/jovyan/work" \
    -e JUPYTER_TOKEN=horcrux -e JUPYTER_ENABLE_LAB=yes jupyter/scipy-notebook
```

```
# Dockerfile
FROM jupyter/scipy-notebook
ENV JUPYTER_ENABLE_LAB=yes
# install custom libraries
RUN pip install bokeh==2.2.2 holoviews
```

```
# docker-compose.yml
version: "3"
services:
  jupyterlab:
    build: .
    environment:
      - JUPYTER_TOKEN=horcrux
    volumes:
      - "./shared:/home/jovyan/work"
    ports:
      - "8888:8888"
```

http://localhost:8888?token=horcrux