Setting up your Python environment

Day 1

Céline Lemarinier

Eviden

September 4, 2023

Outline

About me

About me

- Master in Computer Science (2002)
- Previously worked at :
  - University of Tennessee - Knoxville
  - PayPal - Dublin
  - Eloquant - Grenoble
- Currently working as a software engineer for Eviden in the Data Management team
- celine.lemarinier@eviden.com

What is Python ?

Programming languages

Programming languages

The goal of a **programming language** is to write instructions in a way that is **understandable** by the computer.

Programming languages

Programming languages

The goal of a **programming language** is to write instructions in a way that is **understandable** by the computer.

A programming language can be:

- **Compiled**: machine code is generated from source code;
- **Interpreted**: code can be read directly by the computer without the need of transforming it.

What is Python ?

Python

Python is a **high-level, general-purpose programming language**, which emphasizes code readability.

What is Python ?

Python

Python is a **high-level, general-purpose programming language**, which emphasizes code readability.

Python is **interpreted** and needs an **interpreter** to be executed on the target system.

Python

Python is widely used across the academic and industrial community:

Python

Python is widely used across the academic and industrial community:

- Leader in Data Science/Deep Learning;

Python

Python is widely used across the academic and industrial community:

- Leader in Data Science/Deep Learning;
- Web applications;

Python

Python is widely used across the academic and industrial community:

- Leader in Data Science/Deep Learning;
- Web applications;
- Client/Server applications

Python

Python is widely used across the academic and industrial community:

- Leader in Data Science/Deep Learning;
- Web applications;
- Client/Server applications
- And many more applications ...!

Why Python ?

This summer school focuses on Python (instead of R, which would be the closest concurrent) because:

Why Python ?

This summer school focuses on Python (instead of R, which would be the closest concurrent) because:

- Very easy to learn and understand;

Why Python ?

This summer school focuses on Python (instead of R, which would be the closest concurrent) because:

- Very easy to learn and understand;
- Can be scripted or properly packaged to be production ready;

Why Python ?

This summer school focuses on Python (instead of R, which would be the closest concurrent) because:

- Very easy to learn and understand;
- Can be scripted or properly packaged to be production ready;
- Many available libraries are available for Data Science;

Why Python ?

This summer school focuses on Python (instead of R, which would be the closest concurrent) because:

- Very easy to learn and understand;
- Can be scripted or properly packaged to be production ready;
- Many available libraries are available for Data Science;
- Can easily be compiled for better portability / performance

Installing Python

Downloading Python

- Visit the official Python website: `https://www.python.org`
- Click on the ”Downloads” tab to access the download options.
- Choose the appropriate version for your operating system (Windows, macOS, Linux).
- Make sure to select the latest stable version available.

CAREFUL: all libraries are not necessarily compatible with every Python version. Check carefully if you want a specific version.

Windows Installation

- For Windows, download the executable installer.
- Run the installer and make sure to check the box that says "Add Python x.x to PATH".
- Click "Install Now" to start the installation process.
- The installer will set up Python and the associated tools on your system.

macOS Installation

- For macOS, download the macOS installer package.
- Run the installer package and follow the prompts.
- Python is pre-installed on most macOS versions, but using the installer ensures you have the latest version.

Linux Installation

- For Linux, Python is often pre-installed. Open a terminal and check with the command `python3 --version`.
- If not installed, use your package manager to install Python.
- On Debian-based systems: `sudo apt-get install python3`
- On Red Hat-based systems: `sudo yum install python3`

Verifying Installation

- Open a terminal (or Powershell if you are on Windows) and type `python3` to start the Python interpreter.
- You should see the Python prompt (>>>), indicating that Python is installed and running.
- To exit the interpreter, type `exit()` or press `Ctrl` + `Z` followed by `Enter`.

Possible troubleshooting: Adding Python to your path

Windows sometimes behaves weirdly regarding its PATH and you need to manually add Python to be able to run it in the command line. You may also need to restart your computer for changes to take effect.

Package management using pip

pip

pip

pip is the default package manager for Python, used to install, manage, and distribute Python packages.

pip comes with Python installs ever since version 3.4.

pip

pip

pip is the default package manager for Python, used to install, manage, and distribute Python packages.

pip comes with Python installs ever since version 3.4.
To check if pip is properly installed on your system, run in your terminal:

```
>>> python3 -m pip list
```

Installing Packages

Python packages

A Python package is a collection of **related modules, organized in a directory hierarchy, providing sets of functions, classes, and variables**. Packages allow you to benefit from features developed by other computer scientists instead of **redeveloping yours from scratch**.

Installing Packages

Python packages

A Python package is a collection of **related modules, organized in a directory hierarchy, providing sets of functions, classes, and variables**. Packages allow you to benefit from features developed by other computer scientists instead of **redeveloping yours from scratch**.

- To install a package, open a terminal and use the command:
- `pip install <package_name>`
- `pip` will download and install the specified package and its dependencies.

Example

To check if you have a working install of pip, run `pip install numpy`.

Listing Installed Packages

- To see a list of installed packages, use the command:
- `pip list`
- This will show all installed packages along with their versions.

Listing Installed Packages

- To see a list of installed packages, use the command:
- `pip list`
- This will show all installed packages along with their versions.

Example

Check if `numpy` that you have previously installed is properly listed.

Uninstalling Packages

- To uninstall a package, use the command:
- `pip uninstall <package_name>`
- This will remove the specified package from your system.

Example

Remove `numpy` by running `pip uninstall numpy`.

Requirements Files

- A requirements file (`requirements.txt`) lists all the packages required for a project.
- To install all the packages listed in a requirements file, use the command:
- `pip install -r requirements.txt`

Starting tomorrow, for each lab session, you will be able to install the dependency for each lab using the command `pip install -r requirements.txt`.

Working with virtual environment

Virtual environments

Python virtual environments

A **Python virtual environment*** is a self-contained directory that contains a Python installation and its own set of packages.

Virtual environments

Python virtual environments

A **Python virtual environment*** is a self-contained directory that contains a Python installation and its own set of packages.

- Allows you to manage multiple isolated Python environments on the same system.

Virtual environments

Python virtual environments

A **Python virtual environment*** is a self-contained directory that contains a Python installation and its own set of packages.

- Allows you to manage multiple isolated Python environments on the same system.
- Useful for projects with different dependencies or versions, preventing conflicts.

Virtual environments

Python virtual environments

A **Python virtual environment*** is a self-contained directory that contains a Python installation and its own set of packages.

- Allows you to manage multiple isolated Python environments on the same system.
- Useful for projects with different dependencies or versions, preventing conflicts.
- Helps maintain a clean and organized development environment.

Creating a Virtual Environment

- Using the built-in ‘venv‘ module (Python 3.3+):
    - `python3 -m venv <path/to/venv>`
- Activating the virtual environment:
    - On Windows: `<path/to/venv>/Scripts/activate`
    - On macOS and Linux: `source <path/to/venv>/bin/activate`
- Deactivating the virtual environment:
    - `deactivate`

Using a Virtual Environment

Once activated, the virtual environment's Python interpreter and packages take precedence over the system-wide Python.

- Install packages using `pip`, which installs them into the virtual environment.
- To check which packages are installed in the virtual environment: `pip list`

Benefits

Working with Python environment is **A MUST DO** in Python.

Benefits

Working with Python environment is **A MUST DO** in Python.

Benefits:

- **Isolation**: Prevents conflicts between project dependencies.

Benefits

Working with Python environment is **A MUST DO** in Python.

Benefits:

- **Isolation**: Prevents conflicts between project dependencies.
- **Reproducibility**: Easily share environment specifications for consistent setups.

Benefits

Working with Python environment is **A MUST DO** in Python.

Benefits:

- **Isolation**: Prevents conflicts between project dependencies.
- **Reproducibility**: Easily share environment specifications for consistent setups.
- **Easy cleanup**: Delete the virtual environment folder to remove all associated packages.

Benefits

Working with Python environment is **A MUST DO** in Python.

Benefits:

- **Isolation**: Prevents conflicts between project dependencies.
- **Reproducibility**: Easily share environment specifications for consistent setups.
- **Easy cleanup**: Delete the virtual environment folder to remove all associated packages.
- **Version compatibility**: Maintain different Python versions for different projects.

Running Python

Python can be run:

Running Python

Python can be run:

- Interactively, through your terminal, iPython or using Jupyter Notebook;

Running Python

Python can be run:

- Interactively, through your terminal, iPython or using Jupyter Notebook;
- As a script;

Running Python

Python can be run:

- Interactively, through your terminal, iPython or using Jupyter Notebook;
- As a script;
- As a structured package, with an entry point.

Running python scripts

A python script is a file with the extension `.py`, which can be run using the interpreter.

Running python scripts

A python script is a file with the extension `.py`, which can be run using the interpreter.

Example script

Write the command `print("I love digital humanities!")` in a file, save it as `test.py` and run it using `python test.py`.

Writing Python package

More advanced Python software engineering consists in organizing the code into **modules** and providing **entrypoints** for user to use our code.

Writing Python package

More advanced Python software engineering consists in organizing the code into **modules** and providing **entrypoints** for user to use our code.

Over this week, we will focus on **interactive** Python using Jupyter Notebook.

Working with Jupyter

Jupyter notebooks

Jupyter notebooks

Jupyter Notebooks are **interactive, web-based environments** for creating and sharing documents containing live code, equations, visualizations, and narrative text.

Jupyter notebooks are widely used for **data analysis, scientific computing and machine learning**. It also provides easy sandboxing environment to learn Python !

Installation

Running this in your previously created virtual environment:

- Install jupyter using pip: `pip install jupyter`
- Once installed, you can start Jupyter Notebook from the command line in the desired folder: `jupyter notebook`
- Open the Jupyter Notebook interface in your web browser (or click on the provided URL depending on your version)

Notebook Interface

- The Jupyter Notebook interface consists of cells.
- Cells can contain code or text (Markdown).
- You can execute code cells individually.
- Keyboard shortcuts help navigate and interact with the interface efficiently.

Code Execution

- To execute a code cell, select it and press `Shift + Enter`.
- Output will be displayed below the cell.
- Variables and outputs persist across cells within the same notebook.
- You can also interrupt or restart the kernel, which clears all variables and resets the notebook's state.

Markdown Cells

- Markdown cells allow you to include formatted text, headings, lists, links, images, and more.
- They provide rich narrative capabilities to document your code and analysis.
- You can render equations using LaTeX syntax within Markdown cells.

Sharing and exporting your notebooks

- Jupyter Notebooks can be saved in the `.ipynb` file format.
- Notebooks can be easily shared with colleagues or published online via platforms like GitHub, Jupyter Notebook Viewer, or Jupyter Hub.
- Notebooks can also be exported to various formats, including HTML, PDF, LaTeX, and slides.

Practicalities for labs

Accessing the lab material

Please download for each day the zip folder on the Github `py4shs`:

Questions

Questions ?

We will practice using Jupyter and Python this afternoon !

To launch the lab

- download the prog_basics lab
- inside the folder, create a virtual environment
- activate the virtual environment