Programming in Python[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2023/24, I semester

Lecture XV: Using Third-party libraries

Third-party libraries

Python is "sold" *batteries included* (with many useful built-in libraries). Moreover, like many modern programming environments, it has standard **online package directories** that list libraries produced by independent developers.
https://pypi.org/
The Python package index currently lists almost 300K libraries!

Installing a library

The details are explained here: https://packaging.python.org/tutorials/installing-packages/

- In most cases it is very easy, the pip program does all the magic
- It is **very** important to understand the difference between a system-wide and a project-specific installation.

System-wide vs. Project-specific

If you don't take special precautions, a package is installed in a way that makes it available to your Python system: every Python interpreter you launch sees them.

- In many cases, this is **not** what you want
- **Different projects/programs might depend on different versions of the libraries**
- Libraries themselves depend on other libraries, you want to understand exactly which packages your program is using in order to **reproduce** the settings on other machines

83

Virtual environments

Python provides the idea of virtual development environments (`venv`)

- You can create one with: `python -m venv CHOOSE_A_NAME`
- You must activate it (syntax depends on your OS): `CHOOSE_A_NAME\Scripts\activate.bat`
- In an active virtual environment all the installation are **confined** to it
- You can get the list of installed packages with `pip freeze`

84

Simplified venv administration

Virtual environments are key to avoid messing up your system. Many tools simplify their administration.

- `pipenv` (my preferred one, we will use this)
- `poetry` (similar to `pipenv`, currently less popular, but it has a better dependency control, a bit more complex)
- `conda` (uses its own package index, great flexibility and complexity, manage different python versions)

85

Virtual environments caveats

When you are working in a Python virtual environment, remember to launch **all** your development tools "inside" the virtual space.
For example, to use Thonny you have to activate the proper virtual environment each time you launch the application.

86