QLSC 612: Fundamentals of Neuro Data Science

Introduction to Python and virtual environments

Michelle Wang
14 May 2024

Outline

- The Python programming language and some important features
- Python environments
- Running Python code
- Python basics:
  - Variables, types and operators
  - Important data structures: lists, dictionaries, tuples and sets
  - Conditionals and loops
  - Errors and exception handling
  - Functions and variable scope
  - Objects and classes

Resources to get started

Textbooks/tutorials:

- Official Python tutorial (free)
- Think Python, 3rd edition (textbook) (free)
- Python distilled, Python cookbook, Python essential reference (David Beazley)
- Fluent Python (Luciano Ramalho)

Video recording of the 2021 lecture (given by Jacob Sanz-Robinson)

If you already know R or MATLAB:

- Primer on Python for R Users
- An introduction to Python for R Users
- Python for Matlab Users — VOLTTRON 9.0 documentation

The Python programming language

- Created in 1991 by Guido van Rossum
- Free, open, and multiplatform
- Versatile: used in many domains
- Large community and great ecosystem for data science and machine learning
- Simple, easy-to-learn syntax that emphasizes readability

What does this snippet do?

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
    print(number)
```

https://pythoninstitute.org/about-python

Features of Python

- **High-level language**: strong abstraction from computer hardware
- **Interpreted language**: does not need to be compiled
- **Dynamically typed**: variables can change types
- **Object-oriented** (more on that later)

The zen of Python

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Getting started with your Python environment

`conda` is a Python package manager that comes with the Miniconda/Anaconda Python distribution

- Activate the environment created during the installation lab:
  - `conda activate qlsc612` (in a Terminal window)
- You should see your prompt change to have `(qlsc612)` at the beginning

Useful resource: conda cheatsheet

See also: venv (Python built-in) as an alternative to `conda`

- Some systems (e.g., Compute Canada high-performance computers) do not support `conda`

Adding packages to your environment

- `conda install <PACKAGE_NAMES>`
- There are some packages that cannot be installed with `conda`
  - Typically smaller/less popular packages
  - You can use `pip` to install them instead: `pip install <PACKAGE_NAMES>`
- Be careful when mixing `conda install` and `pip install`!
  - In general, install all your `conda`-installable packages first, then `pip install` the rest
  - See this article for details

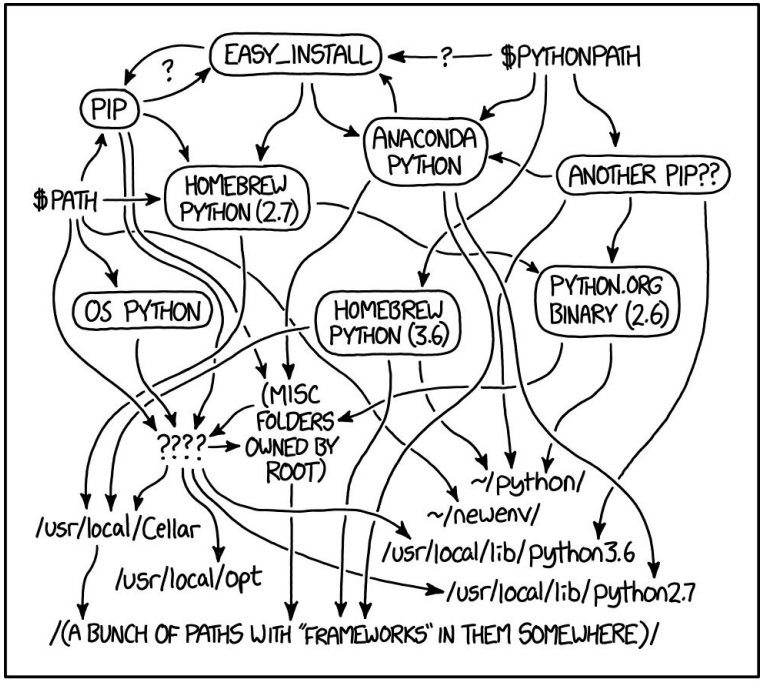
Why use Python environments?

Python Project A requirements

```
python 3.9
numpy 1.20.1
matplotlib 3.3.4
```

Python Project B requirements

```
python 3.7
old_package
numpy ??.??.??
```



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

Why use Python environments?

It can be difficult to manage different versions of Python and Python packages

- Solution: create a different environment for each project
  - Each environment has its own dependencies
  - Updating one environment does not after the other ones
- **Bonus: increase the reproducibility of your work!**
  - There are ways to help others create a Python environment similar to yours
  - `requirements.txt` file (example)
    - `pip install -r requirements.txt`
  - For `conda` only: `environment.yml` (example)
    - `conda env create -f environment.yml`

Running Python code: from the shell

- A Python file is a text file. By convention it should have the `.py` extension
- Let's create and run simple Python script called `hello.py`
- In a Terminal window, type in the command `python` (which points to the Python interpreter) followed by the path to your script
  - E.g., `python hello.py` in this case
  - If you have multiple versions of Python installed and in your `PATH`, you may need to explicitly specify `python3` or `python3.9`
    - Tip: you can use the `which` shell command to check the path of your Python executable
- Alternatively:
  - Add the shebang line `#!/usr/bin/env python` to the top of the file
  - Make the script executable (e.g., `chmod u+x hello.py`) (more on file permissions)
  - Run it as you would run a shell script (e.g., `./hello.py`)

Under the hood: the Python interpreter

- Python is an interpreted language (as are R and MATLAB)
- When you run Python code:
  - The code is translated into **byte code** (`.pyc` files)
    - Low-level set of instructions that can be executed by an interpreter
  - The byte code is executed on a virtual machine (VM) and not a CPU
  - The interpreter checks the validity of variable types and operations (as opposed to having to declare them and having them checked on compilation)
- **Advantage**: given the byte code and the VM are the same version, the byte code can be executed on any platform
- **Disadvantage**: typically slower than compiled languages
  - Compiled languages include C, C++, Java
- More information here

Running Python code: in a Jupyter Notebook

To open the notebook in VSCode, type the following in a **Terminal window**:

- `conda activate qlsc612` # activate the qlsc612 environment
- `code <PATH_TO_NOTEBOOK.ipynb>` # open the notebook in VS Code

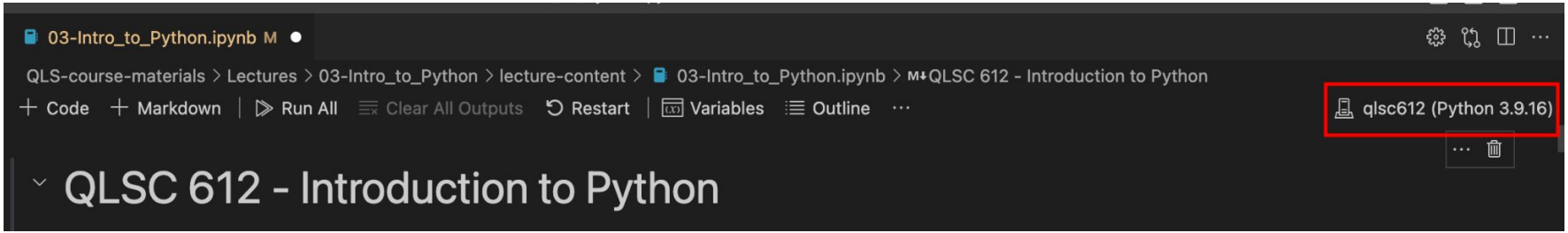
The notebook should now appear in a new VS Code tab or window.

Click on a cell to select it, and press `Ctrl` + `Enter` to execute the code.

If your shell complains that the `code` command cannot be found, you can install it by going to the VS Code Command Palette (`Ctrl`/`Cmd`+`Shift`+`P`) and typing/selecting ***Shell Command: Install 'code' command in PATH****.*

Running Python code: in a Jupyter Notebook

- When running a Jupyter notebook in VS Code, you may also need to specify the Python environment (kernel)
- There will be a **Select Kernel** button in the top right corner of the Jupyter notebook, click it and select the one that looks like **qlsc612 (Python 3.9.X) miniconda3/envs/qlsc612/bin/python**
- The button should be updated to read **qlsc612 (Python 3.9.X)**
    - This is the Python environment we have just created for this course: make sure it is the one you are using for later modules.



Some notes on Jupyter Notebooks

- Jupyter notebooks are interactive documents that can combine text elements and code (see also: Project Jupyter and Jupyterlab)
- They can be handy for exploring a dataset or presenting a report, but they are **not necessarily the best choice for running a full-fledged analysis**
- Running a notebook also requires some external dependencies (e.g., `jupyter`, or `ipykernel` if using VS Code with the Jupyter Notebook extension).
- Notebooks are also more complicated than simple Python scripts and modules:
  - Under the hood, they are JSON files with metadata for code snippets (try running `head <PATH_TO_A_NOTEBOOK>` in the Terminal)
  - Refer to the documentation if you are curious about how they work

The IPython shell

- Interactive Python shell that can be useful for quick checks/debugging
  - Alternative to the regular Python shell (run `python` without any arguments)
- Invoked with the `ipython` command (and requires `ipykernel`/`jupyter`)

```
(qlsc612) Michelles-MBP:04_intro_to_python michellewang$ ipython
Python 3.9.19 | packaged by conda-forge | (main, Mar 20 2024, 12:53:33)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print('This is the iPython console')
This is the iPython console

In [2]:
```

- If you use `ipython -i <PYTHON_FILE_PATH>` to run a script, at the end of the script it will open an IPython shell will all of your variables/functions there

Questions?

Next: Python basics

- Follow along in the `QLS-course-materials/Lectures/2024/04_intro_to_python/lecture/python_basics.ipynb` notebook