Fundamentals of using Python in Health Related Research

Nicola Orsini and Peter Alping

Schedule

- **Morning:** 09:00-12:00 / **Afternoon:** 13:00-16:00
- **Monday-Wednesday:** Room 326 / **Thursday:** Room 218 / **Friday:** Parker (Widerströmska)

Day / Block	Topic
Monday	Python Basics `python`
Tuesday	Working with Data `polars`
Wednesday	Data Visualization `matplotlib`
Thursday	Data Modelling and Analysis `scipy` `statsmodels` `lifelines` `sklearn`
Friday	Questions and Complementary Tools `git` `just` `quarto` + **Take-Home Exam**

Learning Outcomes

This course aims to introduce the fundamental elements of the Python programming language, using motivating examples from health-related research.

After successfully completing this course, you should be able to:

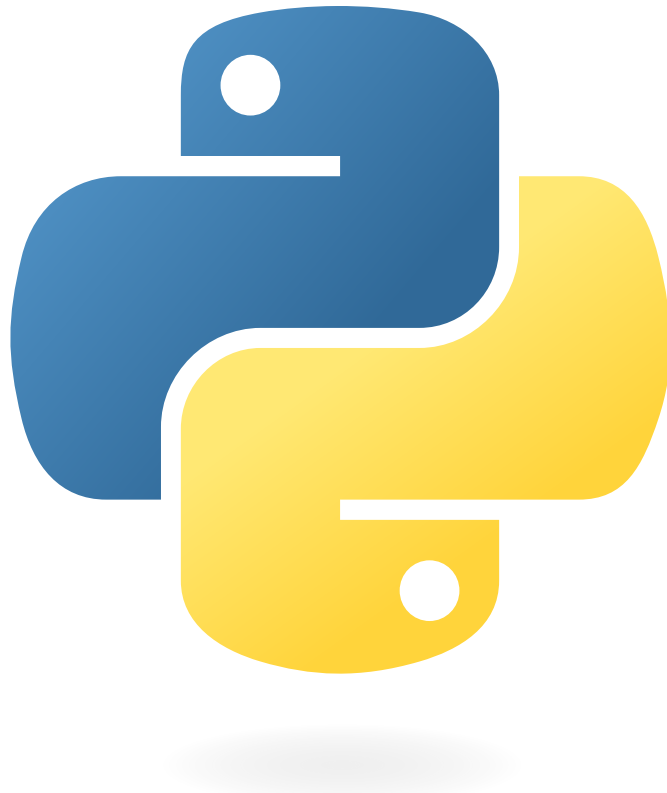
- Use the fundamentals of the Python programming language
- Work with data by importing, describing, joining, grouping, and aggregating it
- Create data visualisation for both data exploration and publication
- Fit your data to basic regression models (linear, logistic, Cox)
- Make a simple prediction model using machine learning techniques
- Have an idea about complimentary tools, such as `Git`, `Just`, and `Quarto`

Examination will be in the form of an individual written exam (pass/fail).

Course material available at: pythondatascience.dev

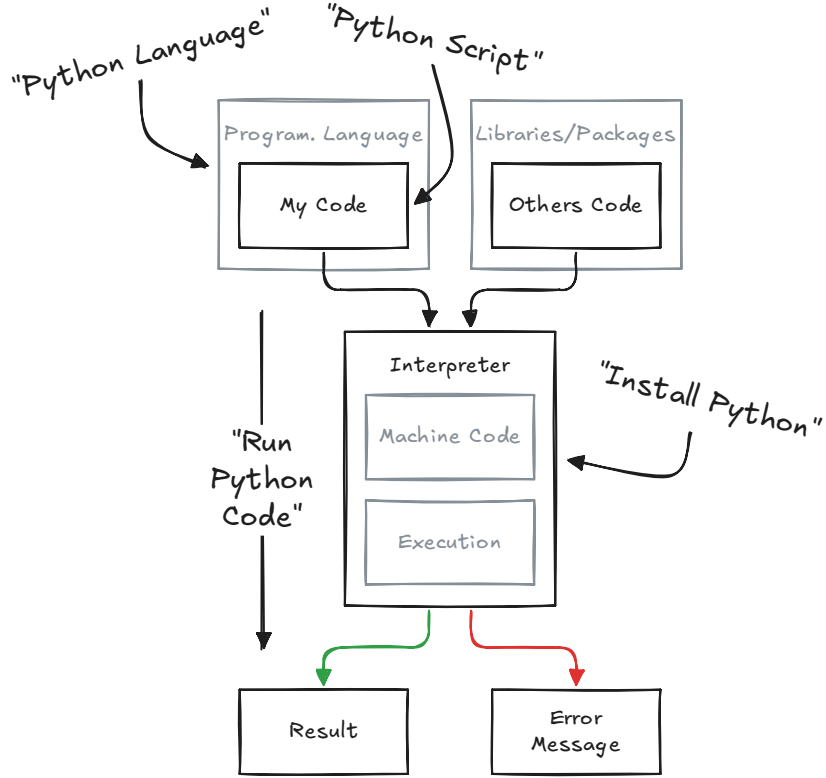
Introduction

- Python is a high-level, general-purpose programming language
- Increasing use in "data science"-type research, rivalling R, STATA, SAS
- Especially common in machine learning and bioinformatics
- Interpreted language (no compilation step)



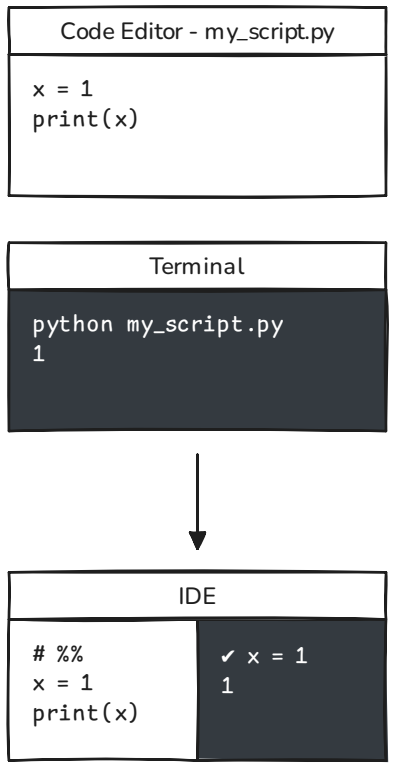
Language / Interpreter / Editor

- **Programming Language:** This is the human-readable language that we write our code in to instruct the computer to do things
- **Interpreter:** Software that translates the programming language into code that can be understood and used by the computer
- **Editor:** Software that we use to write our code and save them as files. Can be any text editor (eg. MS Notepad), but editors specifically designed for coding provide additional features



What does this look like in practice?

1. Create a new text file and name it `my_script.py`
2. Open the file with a text editor (eg. Notepad)
3. Write your code and save the file
4. Send it to the Python interpreter



Integrated Development Environment (IDE)

Software for working with code that has a lot of extra functionality apart from just a text editor, such sending code to the interpreter, debugging, and source control.

In this course, the recommended IDE is VSCode.

Why Programming Matters in Research

- Common notion that programming is just a means to an end
- Writing good code is not just about making things work
- It is about **making research reproducible, scalable, and trustworthy**
- The code is our experimental setup and as researchers we need to write clean, well-documented code
- **Reproducibility:** Others should be able to run our code and get the same results
- **Efficiency:** Knowing our tools allow us to iterate faster and generate more value
- The best data scientists are professionals who understand that **the code is the research**
- **We should treat our code with the same respect as we would any other scientific instrument.**

More on this topic: Richard McElreath - Science as Amateur Software Development (2023 edition) - YouTube

Why Python?

Pros

- Easy to read and learn
- Combines accessibility with great versatility
- Has a large ecosystem of libraries/packages
- Is free and open-source
- Handles everything from data management to statistical analysis to machine learning

Cons

- Not all statistical methods are implemented
- Might not be what your supervisors/colleagues use

As with all research tools, Python is really only as strong as what functionality has been implemented through different libraries

Python for Data Science

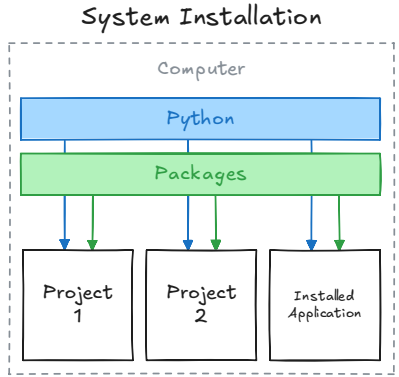
Since Python is a general-purpose programming language, we need to leverage the power of several Python libraries to efficiently use it for data-science/research tasks.

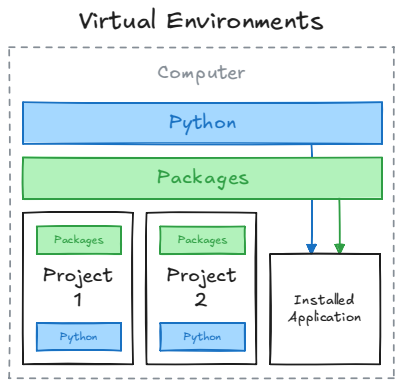
- NumPy - General scientific computing and linear algebra
- SciPy - Optimization, integration, interpolation, statistics, and more
- Pandas - Working with data (older)

- Polars - Working with data (modern)
- Matplotlib - Plotting and making figures
- Statsmodels - Estimation of statistical models and tests
- Lifelines - Time-to-event and survival analysis
- Scikit Learn - Predictive data analysis

Environment

- To make our research reproducible, it is important that we have a controlled research environment
- For our programming environment to be controlled and reproducible, we need to have a specification of the Python and package versions
- Packages in Python are pieces of code written by others and (often) published on the **Python Package Index (PyPI)**
- We do this by setting up a **virtual environment** (`venv`) and using a **package manager** (`pip`)
- Modern tools can deal with both environment and package management (`uv`)





`uv`

- `uv` is a relatively new and very fast virtual environment and package manager for Python
- It can be used as a drop-in replacement for both `venv` and `pip`, combining both tools into one while also providing better performance and dependency resolution

```
uv init  # Create a new virtual environment
uv add package-name  # Install a package
uv run python my_script.py  # Run a Python script in the environment
```

This will create a new virtual environment and install any packages in a .venv directory inside the project.

`Anaconda`

Another popular virtual environment and package manager, especially for data science tasks, is `Conda` (Anaconda). Read more about it in the FAQ.