Python Course

Teacher: Fabrizio Damicelli (fabridamicelli.github.io)

A practical introductory course with a bias towards scientific computing and data analysis. No previous Python or programming knowledge is assumed. Both beginner and intermediate-level students should find useful content in this course, either with tasks that will get them quickly off the ground writing useful code or by improving their understanding of the language.
Disclaimer: This is *will not be* a whirlwind tour of Python libraries. The goal is rather to grasp a fundamental understanding of the language that will hopefully enable students to pursue their own interests and to dive deeper into any library of their interest.

Workload and Course Organization: 2 days, 7 hours per day. There will be explanations interleaved with exercises along the way for each topic plus some integrative mini-projects (see below).

Curriculum

Tentative plan – Changes may occur based on time constraints or specific interests of the students.

Getting Started, Toolchain & Workflow: Why Python? Installing Python. Setting up a project. Third-party libraries and Virtual Environments. Runtime, IPython REPL & Jupyter Notebook.

Basic Syntax: Core data types and data structures. Flow control. Variables; Mutability; Scopes. Functions. Comprehensions.

Effective Coding, Debugging & Testing: Practical considerations to write cleaner code. Exceptions. Built-in debugger, breakpoints, notebook debugger. Basic testing in Python. Code linters. Code-style standards and formatters.

Working with data: Read/write data from/to files in common formats (csv, json, parquet). Writing scripts to process data. Simple Command-Line Applications (CLI). Processing tabular and numerical data (*pandas*, *numpy*). Notebook workflow. Data visualization (*matplotlib*, *seaborn*).

Intermediate Topics: Namespaces. Modules. Standard Library: Some useful built-in modules for scientific computing. Object Oriented Programming. Functional tools: Decorators, cache, partial, generators.

Hands-On Projects: Students can bring their own data/tasks from their that they could use to apply the knowledge to solve concrete problems. Alternatively, these are some potential projects to practice:
1) Scripting Task/s: Create/modify/move files of an experiment to organize the data analysis.
2) Data Visualization: Create or reproduce an interesting figure.
3) Hypothesis Testing: Run a simple test using a re-sampling method (bootstrap).
4) Streamlit Dashboard: Create an interactive dashboard to show results.