McGill initiative in Computational Medicine

Intermediate Python (Part 2)

Workshop Lead: Benjamin Rudski

Facilitator: N/A

Registration link: https://involvement.mcgill.ca/event/264463

Approximate duration: 4 hours

Prerequisites:

- Basic knowledge of Python is required.
- Attendees must be comfortable using variables for simple data types, as well as collections. Attendees should also know how to import modules and understand how to call imported functions.
- To be able to participate in the exercises, participants must either:
    - Have a local installation of Python and Jupyter notebooks. Microsoft Visual Studio Code with the Python extension installed can also be used to run the Notebook.
    - Have a Google Account (to run in-browser as a Colab notebook)

Summary: (2-3 sentences summarizing the workshop)

In this workshop, students will learn new skills to process data using Python. Attendees will learn how to write and document their own functions to encapsulate repeated operations. We will also explore more advanced Python features, such as list comprehensions and enumerated types. Finally, attendees will also gain experience using popular data analysis packages, including NumPy and Matplotlib.

Learning Objectives: (List 3-5 learning objectives participants will learn upon completion of this workshop)

1. Write and document custom functions to perform complex operations.
2. Use advanced Python features, such as list comprehensions, to simplify code.
3. Use NumPy to easily process multidimensional data.
4. Use Matplotlib to generate different types of plots to visualise data.
5. Approach a new package and explore its documentation and examples.

Content:

1. **Module 0 – Welcome (10 minutes)**
    a. Introduction & overview
    b. Motivational problem context: Protein analysis.
2. **Module 1 – Introduction to Functions (45 minutes)**

a. Function Overview
    i. What is a function?
b. Writing Custom Functions
    i. Basic function definitions
    ii. Passing inputs: Defining parameters
    iii. Producing outputs: Return values
c. Documenting Functions
    i. Defining function docstrings
    ii. How to get help from your IDE: Type annotations
d. **Exercise:** Computing amino acid properties for a protein in our samples.

3. **Module 2 – Advanced Python Features (45 minutes)**
    a. One-liners: Some Python Syntactic Sugar
        i. List and dictionary comprehensions
        ii. Ternary operators **(Optional)**
    b. A Brief Overview of Classes
        i. Description of classes, attributes and methods
    c. Enumerations
        i. What are enumerations?
        ii. Defining enumerations
        iii. Using enumerations
    d. Introduction to Data Classes
        i. What is a data class?
        ii. Defining data classes
    e. **Exercise:** Representing experimental samples.
4. **Module 3 – Introduction to NumPy Arrays (1 hour)**
    a. Introduction to Arrays
        i. What is an array?
        ii. Creating arrays
        iii. Indexing arrays
        iv. Simple assignments
    b. Basic Array Operations
        i. Element-wise arithmetic
        ii. Matrix multiplication
    c. Advanced Array Operations
        i. Arrays as objects
        ii. Statistics on arrays
        iii. Reshaping and combining arrays

iv. Exploring the Documentation

d. **Exercise:** Amino acid frequencies.

5. **Module 4 – Visualising Data with Matplotlib (50 minutes)**
   a. Creating Plots with Matplotlib
      i. Introduction to pyplot
      ii. Creating simple plots, bar plots, histograms, violin plots, polar plots, …
      iii. Customising plots: Labelling axes, titles, and more!
      iv. Advanced plots: scatter plots, image plots, 3D plots
   b. Advanced Plotting Techniques
      i. Exporting plots as images
      ii. Working with figures and axes
      iii. Generating subplots
   c. Exploring the Matplotlib Documentation
      i. API reference
      ii. Examples
   d. **Exercise:** Generating plots of amino acid frequency.
6. **Module 5 – Wrap-up (10 minutes)**
   a. What to learn next? How?
   b. How to get help and how not to get help
      i. Your code editor
      ii. Documentation
      iii. Books
      iv. Tutorials
      v. Stack Overflow (and pitfalls)
      vi. ChatGPT (and pitfalls)
   c. Other cool programming topics (to mention, not to cover)
      i. Writing packages
      ii. Object-Oriented Programming
      iii. Developing Graphical User Interfaces
      iv. Hosting projects on GitHub