Programming I

Learn the basics of programming, and start working with Jupyter notebooks.

Instructors: Any notes to add here?

Materials Needed

Jupyter Notebooks:

- 1 - Intro to Jupyter and Output
- 2 - Variables and Operations
- 3 - Lists and Loops
- 4 - Control Flow and Functions
- 5 - Vectorization (optional)
- 6 - Functions and Lambda Functions (optional)

Learning Objectives

1 - Intro to Jupyter and Output

- Learn how to open and run code from a Jupyter notebook.
- Learn to edit code and text.
- Learn how to use the "print" function for output.
- Learn to recognize and correct minor errors.

2 - Variables and Operations

- Understand variable assignment.
- Use variables to store, change, and access information.
- Perform operations on variables using basic arithmetic operations.
- Identify and correct a logic error.

3 - Lists and Loops

- Understand the advantages of using lists.
- Use lists to store data.
- Use loops to perform operations on data in lists.
- Select items from a list by index.

4 - Control Flow and Functions

- Understand the structure of code including "if" and "else" statements.
- Understand basic function calls and definitions.
- Might add to this?

5 - Vectorization

- Understand a vectorized arithmetic operation in contrast to a for loop

6 - Functions and Lambdas

- Revisit functions with a few more examples.
- Introduce lambdas, which are no-name one-line functions.

Programming I

Introduction

This session is intended to provide students with a basic understanding of programming. We cover a lot of material in a short amount of time, so the goal isn't for students to be experts on all of this material. We don't expect them to be ready to write their own code from scratch after this session. Instead, we hope that students will start to get comfortable reading and editing code, and be able to recognize some of the structures that they will see later.

The material is presented through Jupyter notebooks. The instructions are included in the notebooks, so students can work at their own pace. Each notebook includes exercises at the end. It is not necessary that all students complete every exercise; in fact, manny will not. Rather, these exercises are intended as checks on their knowledge and opportunities for additional practice for students with prior programming experience, or who pick up the material very quickly. Some students may move on to the next notebook before the rest of the class, and this is fine.

For each notebook, we start with a short introduction given by the instructor. This isn't strictly necessary, since all of the material is included in the notebook. Rather, this provides an opportunity to move all students on to the next notebook, so that the class progresses at approximately the same pace. It also provides the instructor the opportunity to emphasize the important concepts, and remind students that we aren't aiming for perfect mastery.

1 - Intro to Jupyter and Output

Make sure that every student is able to open this notebook on their computer or iPad. Talk through the first section, "Introduction to Jupyter," as a class, demonstrating how to run code, edit code and text, stop code, and restart the kernel. Go through this slowly; students should be following the same steps on their own computer.

Students can then work through the sections "Output" and "Exercises" independently. As they work on these, instructors will walk around the room providing help as needed. All students should get through the first exercise, which asks them to write code to produce specific output. Once all students have reached that point, the class can move on to the next notebook.

2 - Variables and Operations

Have students open up this notebook, then talk briefly about variable assignment. Draw a diagram with a box corresponding to a block of memory, filled in with a value. This explanation

should be very brief. After students have had some time to work through the first section, "Variables", talk through the last example as a class, using a diagram to show how the stored value is changed.

Students can then work through the rest of the sections independently, with instructors helping as needed. All students should finish the section "Operations." Once all students have started on the exercises, the class can move on to the next notebook.

3 - Lists and Loops

Have students work through the "Lists" section, then work through the examples as a class, using diagrams to show how memory is manipulated.

All students should get through at least the first exercise, then the class can move on to the next notebook.

Do we want to add to this? While loops? Different syntax? More exercises? KT: My opinion is that we should add some selection for lists material, but not while loops. We could potentially add dicts.... but maybe that's too much!

4 - Control Flow and Functions

Talk through the sections "Control Flow" and "Functions" as a class. Emphasize the structure of the code, and talk through it line by line.

It's not essential that students get through any of these exercises, but it's good if they have some time to at least attempt them.

Do we want to add to this? More examples? Exercises where they edit code? KT: I would like examples where they edit code, and also a quick one about multiple outputs

5 - Vectorization

This is an optional bonus notebook for students who get through the rest quickly. It covers vectorization, which is a useful tool in working with data to perform an operation on all entries in an array.

6 - Functions and Lambdas

This optional bonus notebook revisits functions with a few more examples and then introduces lambdas, no-name one-line functions.