**Introduction to Python I**

**Introduction to Python I**

1. **Pre-work:** 
   1. We will use **Colab** through the Google Cloud Console. You need to have a Gmail account to do this. You will also need to have Python downloaded onto your laptop. There is a SharePoint link for software requests, including Python, [here](https://jacksonlaboratory.sharepoint.com/sites/IT/Lists/Software%20Catalog%202/AllItems.aspx?e=HvrPC6&CID=a8c5ecbc%2D26d5%2D407f%2D8180%2Dae2feaceda5f).
   2. **(optional, but highly recommended)** Acquire VS Code (or miniforge) on your own laptop.  VS Code is directly available by request on the SharePoint page for software requests, [here](https://jacksonlaboratory.sharepoint.com/sites/IT/Lists/Software%20Catalog%202/AllItems.aspx?e=HvrPC6&CID=a8c5ecbc%2D26d5%2D407f%2D8180%2Dae2feaceda5f)**. If you would like to have miniforge, you will need to contact IT to have them install it. They can be reached at their usual number: (207) 288 1414.**
   3. Survey
   4. (optional) Motivation: “We suck, until we don't" (excerpt below), P.D. Schloss on learning new skills in academia: <https://www.academichermit.com/2016/06/27/novy-installation.html>

"I am regularly asked what I suggest doing to learn how to write computer programs. It is a great honor for someone else to recognize a set of skills that I work hard to hone, but I honestly want to answer, “the hell if I know.” The honest truth is that I took a semester long class in a horrible programming language - Pascal - and promptly forgot most of it. Then I picked up other languages by looking at the horrendous code from others and modifying it to see what the code did. I would work chapter by chapter through a programming book and then write my own horrendous code. Over time, I like to think that it has gotten better. I hadn’t found a succinct way to describe this until I got a call from a colleague at another institution who was really interested in strengthening his group’s programming skills and perhaps converting some of their home-grown code into a package for others to use. For the first time I answered honestly, “You’re going to suck, but it will get better.”

We live in a world where “sucking” isn’t something we can own and be proud of. We either expect that we can be experts in 24 hours, or we can never be experts because intelligence or expertise is a gift. Both of these ideas are insulting to people that have busted their asses to get better. I know that whenever I’ve wanted to pick up a skill, it’s taken a lot of mistakes and repetition to get better at that skill. Similarly, when there’s a new thing I have learned, but can’t find the time to practice, I know I’m losing my skills. As some of you know too well, the 5 handicap I carried in high school would be a bit higher today because I only play golf once every couple of years. Learning new skills and technologies isn’t just about reading every book on the subject or running a new analysis for a study. We need to practice those skills under the guide of a mentor or a community to get better. We need the support of a community that will help us go beyond the expertise of the community. We are then able to give back to that community to raise the community with ourselves. We need time and mentors to help us along. We need to seek review of our work to get an external assessment of how we’re doing. Because we suck, until we don’t."

1. **Why Python:** <http://imgur.com/gallery/6t6gE>
2. **Course Description:** This workshop is the first of two workshops that constitute a broad survey of the Python programming language, a major language central in the fields of bioinformatics, genomics, computational biology, and Data Science. No prior programming experience is required. We will use Jupyter notebooks. The second half of this series, Intro to Python II, will be offered later in the year. The second course will cover libraries and more advanced topics.
3. **Course Learning Outcomes:** By the end of this course, students will be able to use the Python programming language and do the following:

* Clearly deconstruct a problem into manageable smaller problems.
* Apply appropriate (and efficient) programming tools to solve a problem.
* Understand and interpret other peoples' code.
* Identify problems/deficiencies in existing code (i.e. 'troubleshoot' code, your own code as well as peers) and reformulate the code with incorporated improvements.
* Be able to explain your solution to your peers.

1. **Content:** The following topics will be covered:
   * Python syntax and grammar
   * Data types (integers, floats, strings, lists, dictionaries, tuples) and their associated methods
   * Data manipulation, file management, and debugging
   * Loops, logic, and decision-making
   * Import the RE package, and construct & apply regular expressions (‘regex”)
2. **Locations of the course:** 
   * **See Module 0 for information (**Zoom link; in-person room locations)
3. **Certificate of Completion is available for individuals who complete all 5 assignments and receive a complete (check mark) for all of them.**
4. **Instructor:** Danielle Presgraves, Ph.D. **email:**[danielle.presgraves@jax.org](mailto:danielle.presgraves@jax.org)

The following is an "ideal" schedule. It is likely that we won't stick to it exactly. There will be some shifts in material and focus as we work through the material.

|  |  |  |
| --- | --- | --- |
| **0** | * Miniconda downloaded, Jupyter set up   + You will need to seek help from IT * Complete Survey | **Resources:**   * <https://jacksonlaboratory.sharepoint.com/sites/ResearchIT/SitePages/Conda-Tutorial.aspx> * <https://docs.anaconda.com/free/miniconda/>   **Cheat sheet for Conda:**   * <https://docs.conda.io/projects/conda/en/latest/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf> |
| **1A** | **Jupyter notebook nuts & bolts**   * Notebooks contain chunks that are in a programming language (**JU**lia, **PYT**hon, **R**) or in Markdown language. * Variables, assignment, and memory | **Cheat Sheets for Jupyter Notebook:**   * <https://www.edureka.co/blog/wp-content/uploads/2018/10/Jupyter_Notebook_CheatSheet_Edureka.pdf> * <https://www.datacamp.com/tutorial/tutorial-jupyter-notebook> * <https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html>   **LO:**  Learners use and manipulate Jupyter notebooks: evaluating, and adding cells, and switching between Markdown and Coding cells |
| **1B** | **Data & Manipulation**   * Python is an interpreted language that executes one line at a time * Everything in python is an object with dedicated methods (and attributes) * Data types: Integers, floats, and STRINGS   + Slicing * Variables are names/labels * Mutable versus immutable | **Resources:**  <https://pythontutor.com/>  <https://rosalind.info/problems/locations/>  Specifically: <https://rosalind.info/problems/ini2/>  **LO:**   * Recognize:   + Syntax of Python including the use of whitespace and appropriate commenting of code   + Basic elements of Python: variables, data types, including lists, strings, floats, integers and methods * Implement:   + Appropriate methods of data types including slicing |
| **2AB** | **Lists:**   * Homogenous data types * Methods: Access individual values; slice; replace; append; join; pop   **Tuples:**   * immutable lists with heterogenous data (row out of excel); faster than lists; use () * Tuples can be keys in dictionaries | **Resources:**  <https://pythontutor.com/>  Specifically: <https://rosalind.info/problems/ini3/>  **LO:**   * Recognize:   + Best practices/conventions for built in data type lists   + Start counting at 0   + How lists point to memory is a bit different than strings or integers 🡪 scope of lists is different than strings/integers   + Methods for each data type   + Benefits of an immutable data type * Implement:   + Appropriate methods of data types including slicing   + There are many ways to add items to a list or remove items from a list, but they work through different mechanisms so make sure you understand the details of each method. |
| **3A** | **For Loops**   * Syntax and elements of a loop * What is scope? * How to break out of a loop * Common strategy of filling an external blank list from inside a loop * List comprehensions expanded into for loops | **LO:**   * Recognize:   + Opportunity for repetition and   + Syntax of for loops   + Enumeration, range, zip   + Sort() – modifies in place (original list)   + List comprehension is a for loop reduced to one line * Implement:   + For loop   + For loop to access individual items in a nested tuple or a list |
| **3B** | **I/O, Reading & Writing to external files:**   * File objects and methods   + Open, close, write   + Access mode: r+,w+,a+ * Flat files: txt, csv, FASTA files * User input & user validation * What are paths (how do they work) | **Resources:**  Specifically: <https://rosalind.info/problems/ini5/>  **LO:**   * Recognize:   + File objects, file names, file paths   + Methods associated with file objects   + Always close your file objects   + User input * Implement:   + Opening, creating, reading and writing to and from a file   How to be defensive about user input |
| **4A** | **Conditions & loops:**   * Conditions   + Syntax   + Only evaluates statements when appropriate to do so * Booleans   + Precedence: NOT > AND > OR   + ==, !=,>,<, >=, <= * While loops | **Resources:**  Specifically: <https://rosalind.info/problems/ini4/>  **LO:**   * Recognize:   + How conditions are evaluated   + How to get out of a while loop * Implement:   + Conditional loops |
| **4B** | **More complex data types:**   * Dictionaries (paired data: Key: value; get) | **Resources:**  Specifically: <https://rosalind.info/problems/ini6/>  **LO:**   * Recognize:   + The importance of the dictionary data types * Implement:   + dictionaries |
| **5AB** | **Regular Expressions**   * Regex module * Small number of search rules and methods that you should memorize (or AI)   If you need fuzzy you might need HMM etc, but you can get far with regex | * Recognize:   + Discussion of when regex is the most appropriate solution (and when you might need something more sophisticated, like HMM). * Implement:   + Solve a regex problem. |