**Introduction to Python II**

1. **Pre-work:** 
   1. We will use Google Cloud Platform, and the data and notebooks will be found there as well as in Canvas. This will ensure a consistent experience for everyone. *If you already have Jupyter notebooks (via forge) installed, you can feel free to run the notebooks on your laptop rather than the cloud.*
   2. **Prerequisites:** Completion of the first half of the course is a pre-requisite for taking this course. There are many ways to gain experience with programming in Python and so you may also substitute demonstrated competency in the Python language gained through other sources (please contact the instructor before the course begins to confirm eligibility if you did not take the first half of the course at JAX).
2. **Why Python:** <http://imgur.com/gallery/6t6gE>
3. **Course Description:**

This workshop is the second half of the introduction to the Python programming language course. We will review the fundamental capabilities of Python, undertake a brief look at regex, a deeper dive into functions with recursion, have an overview of Object-Oriented Programming, and explore important libraries and packages such as NumPy, Pandas, SQLite and any other libraries that we have time to investigate.

1. **Course Learning Outcomes:**

Provide you with the tools to take a computational problem through the process of design, implementation, documentation, and testing. By the end of the second week of Python instruction, you will be able to:

* Bundle decision making into functions.
* Turns functions into modules and call modules from your notebooks.
* Improve the design and encapsulation of your earlier procedural Python solutions with Object-Oriented programming principles
* Have a general understanding of the benefits and drawbacks of the major types of programming: procedural, functional, object oriented and Querying (SQLite)
* Understand, Implement, and leverage commonly used libraries such as NumPy, Pandas, SQLite3.

1. **Locations of the course:** 
   1. **Virtual: See Canvas (Module 0) for link**
2. **To receive a “Certificate of Completion” you will need to submit the daily assignment.**
3. **Instructor:** Danielle Presgraves, Ph.D. **email:** [Danielle.presgraves@jax.org](mailto:Danielle.presgraves@jax.org)

|  |  |  |
| --- | --- | --- |
| **0** | * 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** | Review of Jupyter Notebooks & Python:   * Procedural evaluation and variable assignment * Data types: primitive to complex * slicing * Objects with their methods * File objects * Functions and modularization, modules * for loops, conditions, while loops * dictionaries * Tuples | **LO:**   * Recognize: * Basic syntax and data types of Python, including num, floats, strings, tuples, dictionaries * Implement:   + Appropriate methods of data types, including slicing   + Methods for each data type   + Problem decomposition into manageable parts   + Opening, creating, reading and writing to and from a file |
| **1B** | F**unctions**   * Defined within program * Defined outside of program   + modules * Syntax & scope   + Functions return from where they are called   + You have to actually call them | **LO:**   * Recognize:   + The benefits of chunking   + How to create your own functions   + Functions can return None * Implement:   + Rewrite some code bundled into functions |
| **2A** | **Functions review**  **Recursion:**   * leverages scoping rules * **Base case and General (Recursive) case**   Python Tutor with recursion:  **https://pythontutor.com/render.html#mode=display** | **LO:**   * Recognize:   + Functions can call other functions   + Functions can call themselves   + Types of problems that are optimally solved via recursion. * Implement:   + For loop to access individual items in a nested tuple   + Implement a recursion solution to a common problem |
| **2B** | **Modules**   * The joy of bundling functions and only calling them when they are needed | **LO:**   * Recognize:   + The benefits of modularization   + Syntax of modules   + How to call functions within modules * Implement:   + How to create and call your own modules |
| **3A** | **General overview and comparison of the three major type of programming:**   1. **Procedural:**  * What we have done so far  1. **Functional programming:**  * We are going to focus on list comprehensions and higher order function (map, filter) * If you are interested in longer introduction to this topic, I thought this was quite accessible:   <https://maryrosecook.com/blog/post/a-practical-introduction-to-functional-programming>   1. **Object Oriented Programming:**  * Encapsulation into objects (see details in 3AB) | **LO:**   * Recognize:   + Compare the benefits and challenges of the three approaches to problem-solving in programming styles   + Syntax of Functional programming   + Define a side effect   + Importance of removing side-effects to functional programming * Implement:   + Translate a list comprehension into a for loop and vice versa |
| **3B** | **Object-Oriented Programming**   * Objects have attributes (nouns) and methods (verbs) * Bundled, encapsulated object * Classes * Constructors * Inheritance * polymorphism | **LO:**   * Recognize:   + class is the ‘blueprint’   + understanding what a constructor does   + how to create an instance of a class * Implement:   + Build a Platypus |
| **4A** | (optional preview: Module4\_Overview\_Of\_Libraries)  Major libraries: NumPy   * vectorization * subsetting, apply conditions * the powerful np.array   **Excellent questions and answers with Numpy if you are interested in more practice:**   * <https://github.com/rougier/numpy-100/blob/master/100_Numpy_exercises.ipynb> * <https://www.w3schools.com/python/numpy/default.asp> | **Cheat Sheet for Numpy:**  <https://www.datacamp.com/cheat-sheet/numpy-cheat-sheet-data-analysis-in-python>  **LO:**   * Recognize:   + Why the array object in NumPy is considered so powerful and efficient * Implement:   + Different slicing & subsetting techniques   + Broadcasting across entire array |
| **4B** | Major libraries: PANDAS   * <https://pandastutor.com> * Small nod to Series * Inputting dataframes from URL, manually, reading from a file * The ease of dataframes * Tiny introduction to a ‘stack’: Basic plots with matplotlib * The many ways of manipulating data – a brief overview of an extremely large state space of possibilities   **More examples and explanations of methods are here:**   * <https://jakevdp.github.io/PythonDataScienceHandbook/03.08-aggregation-and-grouping.html> * <https://realpython.com/pandas-groupby/> | **Cheat Sheet for Pandas & Pandas methods:**   * <https://www.datacamp.com/cheat-sheet/pandas-cheat-sheet-for-data-science-in-python> * <https://github.com/pandas-dev/pandas/blob/main/doc/cheatsheet/Pandas_Cheat_Sheet.pdf>   **LO:**   * Recognize:   + Series and Dataframe structures   + Dataframe methods * Implement:   + Different slicing & subsetting techniques   + Plotting with Matplotlib   + Conditions |
| **5A** | Major libraries: SQLite3   * <https://cudbg.github.io/sqltutor/> * database introduction – what are **relational** files? Why do we want datasets that are linked, but distinct? * SQL queries and their unique structure | **Cheat Sheet for SQLite:**   * https://www.sqlitetutorial.net/sqlite-cheat-sheet/   **LO:**   * Recognize:   + Structure of SQL queries   + Difference between querying language and familiar programming languages.   + SELECT, APPEND, SORT, FILTER, JOIN, AGGREGATE   + Similarities between functionality of Pandas and SQLite3 * Implement:   + Construct your own SQL queries |
| **5B** | (This section only if time)  Focus on other major bioinformatics libraries:  **BIOPYTHON**   * Objects in Biopython * Methods in Biopython   **SCIKIT-BIO** | **Cheatsheet & tutorial for BioPython:**   * **https://biopython.org/DIST/docs/tutorial/Tutorial.pdf**   **LO:**   * Recognize:   + Seq object * Implement:   + Biopython methods |