**Objectives**

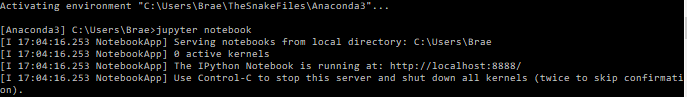
1. Become familiar with using Jupyter Notebooks to write and test python scripts
2. Know how what a string data type is and several common functions to manipulate a string

**Jupyter Notebook**

Jupyter notebook is an integrated development environment (IDE) for python and other computer programming languages. Jupyter notebook opens and runs python scripts in your browser and includes several features we will explore today that make it is easy to write, debug, and share your code.

**Open Jupyter Notebook**

Jupyter notebook is included with the Anaconda python distribution. Open your Anaconda command prompt and type “jupyter notebook” to get started. In most cases a Jupyter notebooks will open automatically as a tab in your web browser. If not, you can manually open it by typing in the web address given in the command prompt:

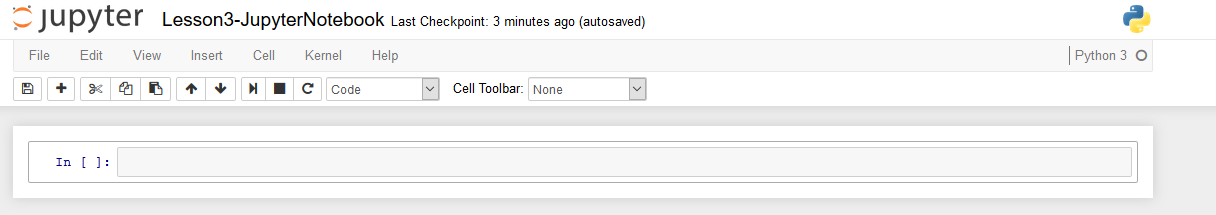


In the notebook’s homepage there are three tabs: “Files”, “Running”, “Clusters”. In the “File” tab you can start a new notebook or open any saved notebooks on your computer. In the “Running” tab you can see, open, and shutdown any active notebooks. The “Clusters” tab is used to access notebooks when running on a computer cluster.

Return to the “Files” tab. Open your desktop folder. Once you are in your desktop select **New>>Python 3** notebook.

**Using your First Notebook**

In a new browser tab, an Untitled Notebook was created. Edit the “Untitled” label and change it to “Lesson3-JupyterNotebook”.



Run Button: Run the script in the cell you are currently editing. (Shortcut! press “CTL + Enter” to run the current cell)

IPython Cell: You can write and edit scripts here.

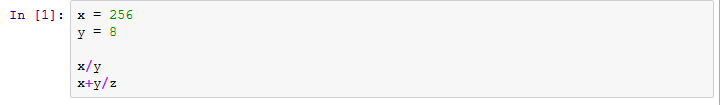
Notebooks use IPython cells to run your code. You can type multiple commands into a given cell, then press the run button to run all the commands together. The after the cells are run, the final output will be displayed on the corresponding “Out” line. Write the following code in your cell and hit run.



You can add a new cell by clicking the  button on the toolbar or by pressing “Alt+Enter”. Any of the variables in a cell that has run can be called by another cell. For example, add our old variable “y” to a new variable “z”.



Variables declared in new cells can be used when you re-run older cells. Edit your first cell to include the following command:



Run it and notice that it used the same z value we declared in our second cell. You will notice that after you run the first cell again, the number on the left changes from 1 to 3. **Caution:** Make sure when running old cells, that the variables it uses haven’t changed since it was first called.

You can restart your environment by clicking on the  button then **Clear all outputs & restart**. This clears all of variables in the notebooks but leaves the scripts inside your cells. Use **Cell>>Run All** to run all your cells from top to bottom. Notice that our first cell has an error because ‘z’ has not been defined.

**Closing Jupyter Notebook**

To close your jupyter notebook you need to shut down kernel (computer engine running the code). To do this go to **File >> Close and Halt**. Exiting out of the browser window will not shut down the kernel. If you forget to shut down your kernel at any time, you can view and shut down all your notebooks in the “Home” browser window under the “Running” tab. You can get to the “Home” window by typing “localhost:8888” into your browsers address bar. You can also shut down the kernels by pressing “Ctl+C” in the command line window you used to launch Jupyter notebook.

**Strings**

Open a new jupyter notebook and title it “Lesson3-Strings”. A string is a data type consisting of a sequence of characters. We will explore how we can use Python’s functions to inspect and modify strings.

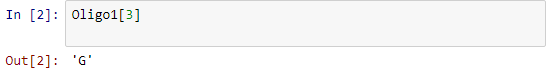
**String Basics**

Strings are defined in python by enclosing characters between either single or double parenthesis. In your Jupyter notebook create a string that contains the DNA bases of the following PCR Primers.



Strings are organized like a list (a data structure we will be covering soon). Each character is given a n index which is used to reference its position in the string. We can retrieve individual characters by using square brackets with an index of which character we want to retrieve. It is important to note that python starts its indexing at 0. In order to retrieve the fourth character we must the number 3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| string | C | A | T | G | G | T | C |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |



We can retrieve a *slice* or multiple characters by selecting a range inside the square brackets using a colon to define the start and stop indices. When using the slice notation, python **does not** include the stopping index. You can see this as we select a slice between the zeroth and seventh indices of our string, the eighth element is not returned.

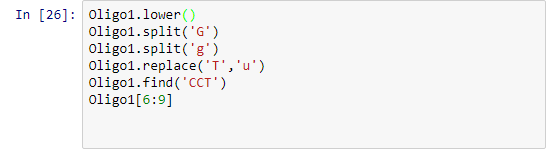


**Critical Thinking with Other String Functionality:**

1. What characters are returned when you insert a negative index into the bracket? Try [-1]
2. Retrieve the last 5 characters of your string using slice notation. Compare the output of using [-5:-1], [-5:-0], and [-5: ]
3. What happens when you use the addition operator on two strings? Try: ‘DNA’+’RNA’
4. Create a 3rd Oligomer string variable where the first half is the first 8 characters of your Oligo1 and the second half is the last 8 characters of Oligo2.

**String Methods**

There a several useful methods built into the string data structure. Type the following into your cell and inspect the results. Run each line in separate cells, or enter and run each command one line at a time to see the different outputs.

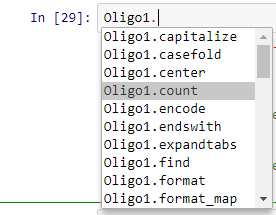


To see all the built in methods associated with strings enter:

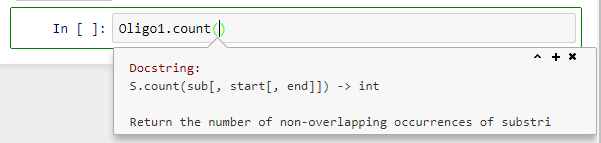


You can also view methods of the string by typing in your string variable, a period, then pressing tab. A drop-down menu of all its methods will appear. You can explore all its methods using the up and down arrow keys on your keyboard.

Type in “Oligo1.” in an empty cell, then press tab. Use the up and down arrow keys to highlight “Oligo1.count”. Press tab to select this method.



Insert a beginning parenthesis and press “Shift+Tab”. A pop-up window appears with the doc string or small help file associated with the method. Press the up arrow button in the pop-up window to see the entire doc-string.

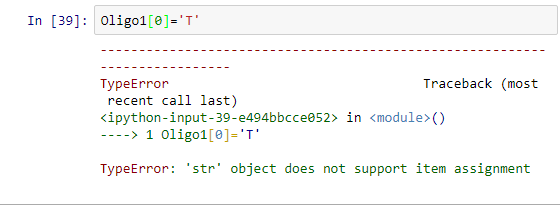


**Critical Thinking Questions with Docstrings:**

1. What does “S” represent in the doc string?
2. What does sub refer to in the doc string ?
3. According to the count docstring, how would you count all occurrences of the character ‘T’ between the 5th and 12th indices of the Oligo1 string?

**Editing a String Variable**

Strings are immutable objects in python. Immutable objects, as the name implies, means that our variable cannot change. This important to consider when using string functions to edit our variables. For example, notice when we try to change the first index only of our Oligo1 string we get an error:



In all of the string methods used above, none of the methods altered the original string. In order to capture and save the changes we need to set the output of these methods equal to a new variable or redefine our older variables.

**Critical Thinking Questions about Editing Strings**

1. Inspect the output of “Oligo2.lower()”. Capture the output of “Oligo2” to a new variable “Oligo4”. Hint: ‘Oligo4 = Oligo2.lower()’.
2. A way to save the output from an immutable object method is to redefine the older variable. Inspect “Oligo2” after running the following: “Oligo2 = Oligo2.lower()”
3. Create a script that replaces the first character of the ‘Oligo1’ string to a ‘T’.

**Homework questions/Group Work:**

1. Write a script that can calculate the percentage of C and G nuceotides for Oligo1 and Oligo2.
2. Calculate the melt temp of Oligo1 and Oligo2 using the following equation:
   1. Tm= 64.9 +41\*(yG+zC-16.4)/(wA+xT+yG+zC)

Where w,y,x,z are the number of A, T G, & C Nucleotides in the primer. Use the “len” command to get the total length of your primers. For example “len(Oligo1)” returns the total number of characters in the string.

1. Create two new variables (comp\_Oligo1 and comp\_Oligo2) that are the complementary DNA sequences of Oligo1 and Oligo2.
2. Copy the sequence found in the GFP.txt file in the Lesson 3 folder. This is the DNA sequence of the GFP, to which the Oligos listed at the start of the lesson are designed for. Determine at what position the forward and reverse primer appears in this sequence. Hint: To find where oligo2 binds in the sequence you need to use its complement sequence and reverse the order.