Introduction to Scientific Computing: A Crash Course

Presented by Travis J Lawrence and Dana L Carper
Quantitative and Systems Biology
University of California, Merced

Worksheet 2.1

For this worksheet we will be using the Jupyter Notebook to run our Python code. To start the Jupyter Notebook server type `jupyter notebook` into your terminal and press enter. This should have opened a web browser. Click on the dropdown menu labeled new and select Python3 to open a new notebook.

Python variables

1. To setup variables for later questions assign the values `0.5`, `8`, `42`, `Penstemon`, and the string `1` to any variable names you want.
2. Use the `print` function to print the value of each variable to an output cell/screen.
3. Using the numeric variables created in `question 1` explore the `+`, `-`, `*`, `/`, `**`, `%` operators. Assign the output of each operation to a new variable and print the value of the new variable. Is the function of each operator clear? If not, try different combinations of variables. What operation does `%` perform?
4. You are not restricted to using only one operator per statement. Using the numeric variables from `question 1` combine multiple operators in a single statement. Devise a set of statements to determine if Python operators follow the same order of operations everyone learned in algebra. Is the order of operations the same? You can also use `()` in math operations. Do these affect the evaluation order in the same way they do in algebra?
5. There are several operators we didn't introduce in the lecture. Test the function of the `+=`, `-=`, and `\=` operators on the numeric variables from `question 1`. What are the function of these operators?
6. Reassign your variables to the original values from `question 1`
7. Try using the `+`, `-`, or `/` operators using a string and numeric variable. What was the outcome? Does this outcome make sense?
8. Use the `*` operator on one of the numeric variables that contains a whole number and one of the string variables. What was the outcome?
9. Use the `+` operator on two string variables. What does this operator do when used on two strings?

Indices and Slicing

Being able to reference a position or range of positions in a string is essential to programming since most data is text. We briefly covered the syntax in the lecture and basic concepts of indexing and

slicing strings. The syntax for string slicing is `string_variable[begin:end:stride]`. The `begin`, `end`, and `stride` indices are optional, but you must at least include either a single index or a `:`. The questions below will help you become familiar with slicing strings.

10. Assign a variable the string `0123456789`. You will be using this variable for the remaining questions and we will assuming its nameed `string_variable`.
11. To effectively use slicing you need to understand how to reference each position in a string. Using the following syntax `string_variable[position]` trying values between `0` and `9` for position. What position references the start of the string? What position references the end of the string? This kind of indexing is called zero-based indexing.
12. Now that you understand the basics of indexing you can start exploring slicing. Using the statement `string_variable[0:end]` provide different values for the `end` index. Is the position referenced by the end index included in the output? What index value for `end` must you use to include the last character?
13. We didn't cover the length (`len`) method in the lecture. Try using this method on your string variable. The syntax is `len(string_variable)`. What is the output of this method? Does it seem like the length method is using zero-based indexing?
14. Using the statement `string_variable[start:5]` try different `start` indices. Is the position referenced by the `start` index included in the output?
15. The `start` and `end` indices are optional when slicing a string and are set to a default value if not provided. Use the statement `string_variable[:end]` with several values for the `end` index to determine the default `start` index. What is the default value? Use a similar method to determine the default value for the `end` index. What happens when you run this statement `string_variable[:]` Was this expected based on the default values for the `start` and `end` indices.
16. In the previous questions you have been using positive indices. In addition to positive indices Python allows negative indices for slicing. Run the statement `string_variable[position]` with values `-1` through `-9` for position. What is the negative index for the last position? For the first position? This might seem confusing at first, but just remember that negative indices are counting from the end of the string instead of the beginning.
17. Using negative indices write a statement that will slice the first four characters. (Hint: `string_variable[:negative_pos]`).
18. Write a statement using negative indices to slice the last four characters.
19. Try combining multiple string slices using the `+` operator (Example: `string_variable[start:end] + string_variable[start:end]`). What was the outcome of using this operator on string slices? Was this behavior expected based on your answer to `question 9` ?
20. The last part of slicing to explore is the `stride` term. Using the default `start` and `end` indices change the value of the `stride` value from `1` to `9` (Example: `string_variable[::stride]`). What effect does the `stride` option have on string slicing?
21. Repeat `question 20`, but this time use values from `-1` to `9` for the stride option. Did you expect this behavior based on your experience with negative index values?
22. When working with molecular sequence data you often need to slice the sequence of interest out

of a much longer sequence. Assign a variable this value: `ATGCGTatatcgacCTGACTCCCtgtactgaCGGCATTAA`. In this sequence uppercase letters represent exonic regions and lowercase letters are intronic regions. Use slicing notation and the `+` operator to assign a new variable with only the exonic regions. Provide two different ways to do this. The first solution must use a default value for either the `start` or `end` index and the second solution must use a negative index.

23. When analyzing protein coding DNA sequences it is common to want to look at codon positions individually. Use the exon sequence obtained in `question 22` and the `stride` index to create three variables each containing a different codon position.