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 2.2

In this worksheet we will be preforming similar analyses to those in worksheet 1.3.1 and 1.3.2 using Python. We will be introducing the `random` module, the `range` function, `strip` and `split` string methods, and the `remove`, `count` and `append` list methods.

Looping through Python sequences

In this section you will be using the Jupyter notebook to become comfortable using for loops to iterate through different Python sequence types. Remember from the lecture that the basic syntax of a for loop is:

```
for current_value in sequence_variable:
    print(current_value)
    statement2
```

1. Open a new Jupyter notebook and create a string variable with more than one letter, a list with multiple values and an integer variable greater than 2.
2. Write a for loop that uses you string variable as the `sequence_variable` that prints the `current_value` to the screen. What is this loop iterating through?
3. Repeat `question 2` using the list as the `sequence_variable`.
4. A common programming task is looping through a range of numbers. The `range` function aids in doing this by taking an integer and producing a sequence of numbers. Write a for loop that uses the `range` function and the integer variable from `question 1` as the `sequence_variable` and prints the `current_value`. What is the first number in the sequence? The last? How many times did the loop execute?

Exploring genome annotations

In this section we are going to walk through reproducing some of the same analyses that we preformed in worksheet 1.3.1. For the following questions use the Atom text editor to write your code and the terminal to run your code.

5. Open a new file in Atom or text editor of choice and save it as `genome_annotation.py`
6. The first thing we need to do is open the file we want to work with. Write the required code to open

one of the three genome annotation files. Save your script and run it from the command line. If everything is correct you shouldn't see any output.

7. The next step to preforming our analyze is writing a loop that iterates through each line of the file. Write a for loop that prints each line of the file. Which variable contains the current line? Run your code. What was the output? Does it appear that there are extra blank lines? Remember that each line ends with a newline character and that print method adds a newline character to everything.

8. The `strip` string method removes leading and trailing whitespace from a string. Use this method on the variable containing the current line when printing it. Run your script. Did this change anything?

9. Instead of printing each line assign the results of the `strip()` method to a new variable.

10. Add a new variable called `line_count` at the top of your script and assign it the value `0`. We will use this variable to keep track of the number of lines. Add code in the loop block to increment this value by one for each iteration. Test your code to make sure there are no errors. Should you see any output on your terminal?

11. Keeping track of the number of lines is not helpful unless we output the value. Add code after the for loop block to print the `line_count` variable.

12. Now we are going to add code to count the number of features for each `chromosome`. Create an empty list at the top of your script called `chromosome`. Run your script to make sure that everything is still working.

13. The `split` string method splits a string fields and produces a list. Use the `split` method on the variable you created in `question 9`, assign the results to a new variable and print this new variable. What does the `split()` method use as its default delimiter? Now remove the line printing this new variable from your script.

14. In the list created in `question 13` what is the index for the `chromosome number` field?

15. The `append` list method will append a value to the end of the list. Using the index found in `question 14` add code to append the `chromosome number` field to the `chromosome` list we created in `question 12`.

16. We want to count the number of times that each `chromosome` occurs in our `chromosome` list. The `count` list method will take a value and return the number of elements in the last that match that value. After the loop block use the `count()` method to calculate the number of times that each chromosome appears in the `chromosome` list. Print the results of the count method.

17. To make our script easier to use on multiple genome annotation files we can open a file provided as an argument to our script. We can add this functionality by using the `sys` module. Import the sys module at the top of your script. Replace the file name in your script with `sys.argv[1]`. Now run your script providing the file as an argument to the script.

Simulating genetic drift using a Moran process

In this section we are going to create a simple stochastic simulation of genetic drift. The Moran model simulates the process of genetic drift in a group of haploid asexual organisms containing one of two alleles with a constant population size. At each time point an individual is randomly selected to reproduce and an individual is randomly selected to die. The probability of a genotype reproducing or dying at each time step is: *i/N* where *i* is the number of individuals with an allele and *N* is the total number of individuals.

18. Create a new file named `Moran.py`. We need to create variables to hold values for the model parameters and to keep track of the number of alleles at each time step. If there is a name in parentheses following the parameter please put that as the name of the variable or list. The parameters are the starting number of individuals with the `A` and `B` genotype, population size, number of time steps in the simulation (num_gen), and the current time step (step). We will also create an empty list that will contain elements which will represent the individuals in our population (mstate). Do this now.

19. Set the number of individuals of each genotype to 50. Set the population size equal to the sum of the individuals from each genotype. Set the number of generations to 10000 and the current time step to 0.

20. The next step is to populate the list with the starting state of our simulation. We will do this using two for loops, the `range()` function and the `append` list method. Use the first for loop to append an `"A"` to the list for each individual starting with the `A` allele and the second loop to append a `"B"` to the list for each individual starting with the `B` allele.

21. To run the simulation we will use a while loop. Because we have not covered conditional statements here is the code to setup the while loop:.

```
while (mstate.count("A") > 0 and mstate.count("B") > 0 and step <
num_gen):
```

Paste this code into your script.

22. During each iteration of the loop we need to randomly select an individual to reproduce, randomly select an individual to die and increase the current time step by 1. Anytime we need to do something that involves randomness we need to use the `random` module. At the top of the script import the `random` module. Try using the online documentation for the `random` module to see if you can find a function to return a random element from a list or sequence.

23. The random function we want is `random.choice(seq)` where `seq` is the list or sequence we want to randomly sample from. This function returns a value from a list with each element having an equal chance of being sampled. We will want to use this function twice, once to select the individual that will reproduce and a second time to select the individual that will die. Add code in the while loop block to randomly sample from our population represented by the list created in `question 19`. Store the first random sample in a variable named `reproduce` and result from

the second random sampling in a variable named `death`. We can also add the code to increment the current time step by 1.

24. We need to add an individual with genotype contained in the variable `reproduce`. We can do this using the `append` list method and the `reproduce` variable to add the new individual to our population list.

25. We need to use the `remove` list method to remove an element from our population list. Use the `remove` and the `death` variable to remove an individual from our population.

26. We have a fully functioning simulation of genetic drift that is just missing output. We want to print the time step, proportion of individuals with an A allele and the proportion of individual with a B allele separated by tabs on each iteration of the loop. We can do this by using the `count` list method and the print function. Here is the syntax for the print function:

```
print(tstep, propA, propB, sep='\t')
```

where `tstep` is a variable containing the current time step, `propA` and `propB` contain the proportion of A and B genotypes respectively. The `sep='\t'` tells the print to separate the variables by a tab.