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 2.3

In this worksheet we are going to continue working with the genome annotation and species occurrence scripts we started in worksheet 2.2. We are going to expand our scripts functionality using flow control, membership testing, and dictionaries.

Practice with dictionaries and flow control

We are going to use a Jupyter notebook to learn the basics of working with dictionaries and flow control.

1. Open a new Jupyter notebook and create this dictionary:

```
example = {"mRNA_count":56, "gene_names":["Gene1", "Gene2"]}
```

2. Add this key:value pair `"gene_count": 24` to the dictionary in `question 1`
3. Use this dictionary to calculate the ratio `mRNAs` to `genes` and print the ratio.
4. Append `Gene3` to the `value` of the gene_names `key`.
5. Add these key:value pairs to the dictionary `"tRNA":0`, `"miRNA":[], "rRNA": ""`, `"snoRNA":{}`.
6. Write a for loop using the dictionary as the `sequence_variable`. Print the `current_value`. What is the for loop iterating over?
7. Modify your loop in `question 6` to print the dictionary value for each `key` instead.
8. Using an `if` statement modify your loop in `question 7` to print the `value` if the `key` is equal to `mRNA_count`.
9. Modify the `if` statement in `question 8` to print the `value` if the `key` contains the string `gene`. You can use membership testing to do this.
10. Add an `else` clause to your if statement that prints the `key`
11. Replace your `if` statement from `question 10` with the one below and run the code. The `key` in the below `if` statement is the dictionary key from the loop variable. Do you see a pattern in the `values` that test as false?

```
if (example[key]):
   print(key, "True")
else:
   print(key, "False")
```

12. We can also use membership testing to see if a dictionary contains a `key`. The structure of this membership test is: `if (key in dictionary):`. Write an if statement to see if our dictionary contains the `key` `"rRNA"`.

Genome Annotations

In the section we are going to modify your original genome annotation file to take advantage of `if` statements and dictionaries. Open your script from worksheet 2.2 and remove any code that does not deal with reading the file and splitting the current line into fields. Your code should look similar to this:

```
import sys
f = open(sys.argv[1], "r")
for line in f:
    line = line.strip()
    spline = line.split()
```

13. Remember from worksheet 1.3.1 that comment lines started with a `#`. Write an `if` statement so that only lines that do not start with a `#` are stripped of leading and trailing whitespace and split into fields. There are two different ways write the conditional statement. The first uses `==` and the second uses the string method `startswith`, and both styles need to use the `not` logical operator.

14. When we encounter a comment line what we really want to do is ignore it and move to the next line in the file. Add an `else` statement to your code from `question 12` with the keyword `continue` in the `else` code block. What does the `continue` keyword seem to do?

At this point your code should look similar to this:

```
import sys
f = open(sys.argv[1], "r")
for line in f:
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue
```

15. Now we are going to add a dictionary to keep track of the number of annotations for each chromosome. Add an empty dictionary to your script named `chromosome` right after your import statement. Add code after the `if/else` statement, but still within the loop block, to use the `chromosome number` field as the `key` for the dictionary and increase the `key's` `value` by

one. Run your script. Did you get a similar error message to the one below? This happened because the `key` did not exist in the dictionary so there is no `value` to increase by one. We will fix this in the next question.

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'test'
```

Before continuing to `question 15` ensure that your code looks similar this:

```
import sys
chromosome = {}
f = open(sys.argv[1], "r")
for line in f:
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue
    chromosome[spline[0]] += 1
```

16. To gain the functionality that we want with the dictionary we need to check if the `key` exists and add it if it does not. We tested if a key existed in `question 12` and can reuse similar code. Replace our original statement incrementing the dictionary `value` by one and use an `if` statement to test if the key exists and if it does increment the value by one. If it doesn't add the `key` with the `value` being equal to `0` and then increment the value by one.

We should now have a fully functional annotation parser that counts the number of records per chromosome. Before moving on your code should look similar to this:

```
import sys
chromosome = {}
f = open(sys.argv[1], "r")
for line in f:
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue
    if (spline[0] in chromosome):
       chromosome[spline[0]] += 1
    else:
       chromosome[spline[0]] = 0
       chromosome[spline[0]] += 1
```

17. Add code to loop through the `chromosome` dictionary and print the `key:value` pairs to the screen. This code should run after you have finished parsing the file.

18. Modify your loop in `question 17` to only print the chromosome with the highest number of annotation records. You need to setup two variables, one to keep track of the chromosome with the highest annotation record count seen so far and the second to hold the record count. You will need to us an `if` statement to update these variables.

Before moving on check your code against the example solution below. It should contain similar logic.

```
import sys
chromosome = {}
f = open(sys.argv[1], "r")
for line in f:
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue
    if (spline[0] in chromosome):
       chromosome[spline[0]] += 1
    else:
       chromosome[spline[0]] = 0
       chromosome[spline[0]] += 1

highest_chromosome_name = ""
highest_chromosome_count = 0
for key in chromosome:
    if (chromosome[key] > highest_chromosome_count):
       highest_chromosome_name = key
       highest_chromosome_count = chromosome[key]

print(highest_chromosome_name, highest_chromosome_count)
```

21. We are now going to add code that will allow us to count the number of annotation records for each `feature_type` per chromosome. To do this we will use a nested dictionary. A nested dictionary has dictionaries has its `values`. This kind of data structure is useful when you have hierarchal data. This is an advance topic so if you having trouble grasping the concept of a nested dictionary at first do not worry. We are going to build our nested dictionary in a way so that the highest level `keys` are `chromosome number` and its `values` are a second dictionary whose `keys` are `feature_types` and values are the count for that `feature_type`. You can access nested dictionary values with this syntax: `chromosome[chromosome_name][feature_type] += 1`. In this example the `key` for the top dictionary is `chromosome_name`, which gives us access to its `value`, that happens to be a second dictionary who's `key` in this example is `feature_type`. Before starting this problem either edit your code or create a new script with the code below:

```
import sys
chromosomes = {}
f = open(sys.argv[1], "r")
for line in f:
```

```
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue

    if (spline[0] in chromosome):
        chromosome[spline[0]] += 1
    else:
        chromosome[spline[0]] = 0
        chromosome[spline[0]] += 1
```

We need to change the code within the `if/else` statement to work with our nested dictionary. Instead of adding one if the `chromosome_name` `key` already exists we need to add a nested `if` statement or to check if the `feature_type` key of the nested dictionary exists. If both exist increment the `feature_type` value by one. If not add the necessary keys and increment the `feature_type` by one. Remember that the `feature_type` is the third field in the line. Attempt to do this now and compare your work to the two examples below.

```
import sys
chromosomes = {}
f = open(sys.argv[1], "r")
for line in f:
    if (not line.startswith("#")):
       line = line.strip()
       spline = line.split()
    else:
      continue

    if (spline[0] in chromosome):
        if (spline[2] in chromosome[spline[0]]):
            chromosome[spline[0]][spline[2]] += 1
        else:
            chromosome[spline[0]][spline[2]] = 1
    else:
        chromosome[spline[0]] = {}
        chromosome[spline[0]][spline[2]] = 1
```

Species Occurrence Data

You have been introduced to all the tools you need to write scripts for basic data analysis. In this section we are going to revisit a problem from worksheet 1.3.2 where we identified the families with the highest and lowest number of records without a species identification. We could not normalize by the total number of records for the family using command line tools, but we can with Python. There will be no guidance on how to solve this problem. I have provided the problem statement below:

22. Using Python and the `Plantae.csv` file which family had the most records not identified to species? Which had the

least? This is absolute count data which can be misleading because of total number of records for each family. Normalize the number of records missing species identification by dividing by the total number of records for that family. Which family had the highest normalized number of missing species? Which had the lowest?