420-PRO-LCU Programming in Python - Lab Exercise 5

October 12, 2017

Goals for this lab:

- Use built-in modules.
- Modify and use your own module.
- Learn about files.
- Do some very basic data analysis.

In this exercise, we will use a data file that contains a set of data from classic experiments that used statistical methods to classify three different species of flower based on measurements of the flower structure. The file is called `iris.txt`. Each line in the file consists of the following fields:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. species name

There are 50 measurements for each of three species, giving a total of 150 data items.

Files

Data in the computer is almost always stored in a *file*. Files can contain text (like Python files, HTML files, or Windows Notepad file), or other data such as images, music, videos, or applications. Text files are usually read or written as a series of lines, each ending with the newline (`'\n'`) character.

A file has a name, which is used to *open* the file. Once a file is opened, we can read from (or possibly write to) the open file. After you are finished with a file, you tell the operating system you're finished by *closing* the file. This frees up the resources used to access the file.

In Python, you open a file using the built-in function `open()`. The function returns an object that represents the open file. You can use this object to read, write, and finally close the file. For a text file, this object is an iterable value that allows you to access each line sequentially in a `for` loop. You can process most text files using a very simple pattern:

```
file = open('data.txt') # open the file for reading.
for line in file:
   # process each line here!
file.close() # close the file.
```

Preliminary

1. Open the file `iris.txt` in Wordpad or another text editor, just to examine the structure and read the comments.
2. Open the file `correlation.py` using IDLE. This is a file which contains an already-implemented function, `correlation(v1, v2)`, which will compute one of the statistics you need to report for this data.

   Notice that this file makes use of a built-in module `statistics` to compute some of the quantities it uses.

Exercise 1

1. Open the file `irisfile.py` and find the function `iris_data()`. This function is incomplete. You need to finish it by writing the code to convert each line in the data file into list of numeric data, and combining the sublist that will represent each line into a bigger list.

   To do this you need to split each line into its five fields, then convert each individual field to the correct type. The four measurements will be of type `float`, while the species will be encoded as an integer using a function described below.

   You will also need to ignore lines that start with the character '#'. Here are two hints about how to approach this:

   - There's a string method that you can use to check this easily.
   - This is a potential application of the `continue` statement.

   Another string method that is occasionally useful is the `strip()` method. It will return a new string that has the same contents as the original string, but with any leading or trailing "whitespace" removed. "Whitespace" refers to spaces, tabs, and newlines.

2. Notice also that this file contains some other functions and data that have already been provided for you:

   (a) `N_FIELDS` - this is just a constant that records the number of fields on each data line. This might be useful in your main program.

   (b) `latin_name_to_number(name)` - this function will be used *in this module* to convert the Latin name of the species to an integer. We need to use numeric values to compute a correlation coefficient between the species and the various measurements.

   (c) `extract_column(data, col)` - the function will return a list that contains all of the data elements for a particular column. For example, assuming you have completed your `iris_data()` function correctly, the following two lines of code should allow you to get all 150 sepal length values:

   ```
   data = iris_data()
   sepal_length = extract_column(data, 0)
   ```

   You could get all of the sepal width values by using `extract_column(data, 1)`, etc.

Exercise 2

In this part, you will write a main program that loads the data file and reports some statistics about the data.

Create a file named `lab5.py` in IDLE and write the code to do the following:

1. Import the functions `mean` and `stdev` from the builtin `statistics` module.
2. Import the function `correlation` from the `correlation` module.
3. Import the useful functions and data from the `irisfile` module.
4. Use all of these functions, along with the appropriate builtin functions, to print the following statistics for each measurement:

   - minimum value
   - maximum value
   - mean (rounded to 2 decimal places)
   - standard deviation (rounded to 2 decimal places)
   - correlation with the species type (rounded to 4 decimal places)

   You may use the `extract_column` function to get all 150 values associated with a particular measurement.

   Your results should look like this:

```
sepal length: 4.3 7.9 5.84 0.83 0.7826
sepal width: 2.0 4.4 3.06 0.44 -0.4267
petal length: 1.0 6.9 3.76 1.77 0.949
petal width: 0.1 2.5 1.2 0.76 0.9565
```

You can improve the formatting slightly by using tab characters (`'\t'`) between the fields as you print them out. We will talk about how to format data more neatly in class soon.

Optional

See what happens if you compute the correlation between a vector of 150 random numbers and the species labels.

You can use the `random` function from the `random` builtin module to generate a series of random float values between 0 and 1.

Submitting your work

For this lab, you will submit two files, your completed `irisfile.py` and `lab5.py`. Please combine them into a single ZIP file before submitting!