Reading Columns of Data into Lists

Leigh J. Little SUNY Brockport November 20, 2020

Contents

1 Introduction

This document will discuss how to read a data file consisting of a rectangular block of numbers. In particular, the process of extracting a particular column of numbers into a list or `numpy` array will be covered.

2 The `.split()` Method

Suppose you have a data file consisting only of numbers arranged in a rectangular grid and that each column of numbers represents a fixed quantity. For example, you could have a file where column 1 is a set of x-coordinates, column 2 is a set of temperatures, column 3 is a set of pressures, *etc.*. You would like to read the values from the file so that each column is put into it's own list. For now we will make the assumption that the number of columns is known.

To do this, we need to loop over each line in the file, then partition out the numbers in each line into the appropriate list. This can easily be done using some new Python commands.

Download the file `data1.dat` from the webpage. This file contains 5 columns of numbers. To start, we will use the `with` structure from before. Type the code below into a script file and run it.

```
with open('data1.dat','r') as fnum:
     for currline in fnum:
          print(currline)
```

You should see the output

```
1 2 3 4 5

5 4 3 2 1

1 2 3 4 5

5 4 3 2 1

1 2 3 4 5

5 4 3 2 1

1 2 3 4 5

5 4 3 2 1
```

As from the previous examples, there are blank lines because of the additional newline (`\n`) character that gets added to the print command.

Now make a small change to the file and run it again

```
with open(filename) as fnum:
     for currline in fnum:
          t = currline.split()
          print(t)
```

You should see that the output has changed to

```
['1', '2', '3', '4', '5']
['5', '4', '3', '2', '1']
['1', '2', '3', '4', '5']
['5', '4', '3', '2', '1']
['1', '2', '3', '4', '5']
['5', '4', '3', '2', '1']
['1', '2', '3', '4', '5']
['5', '4', '3', '2', '1']
```

In this code, we have used the `.split()` method and applied it to the current line in the file (`currline`). The `.split()` method takes a string and puts the 'words' in the string into individual list elements. Here, the words are the individual numbers. The method uses the spaces in the string to determine where one word begins and ends. We call the spaces *delimiters*. A delimiter is a charater that is used to indicate separate entities in a string. The most common delimiters are spaces, commas and colons.

In the event that a file uses something other than a space as the delimiter, you can send in an optional argument to the `.split()` method.

```
currline.split(',')   # uses the comma as the delimiter
currline.split(':')   # uses the colon as the delimiter
currline.split(', ')  # uses the combintation of comma and space as the delimiter
```

Now we can finalize the process of putting the individual elements of the list obtained by splitting the line into their own lists.

Enter the code below into a script file and run it.

```
L1 = []    # Initialize L1 through L5 as lists
L2 = []
L3 = []
L4 = []
L5 = []
with open('data1.dat','r') as fnum:
     for currline in fnum:
          t = currline.split()
          L1.append(t[0])  # Use the append method to insert t[0] into L1
          L2.append(t[1])
          L3.append(t[2])
          L4.append(t[3])
          L5.append(t[4])
print(L1)
print(L2)
print(L3)
print(L4)
print(L5)
```

You should see the output

```
['1', '5', '1', '5', '1', '5', '1', '5']
['2', '4', '2', '4', '2', '4', '2', '4']
['3', '3', '3', '3', '3', '3', '3', '3']
['4', '2', '4', '2', '4', '2', '4', '2']
['5', '1', '5', '1', '5', '1', '5', '1']
```

At this point, we have almost achieved the goal. Each column of the data file has been placed into its own list, however the elements are still strings, not integers. We need to use the `int` function before appending the values

```
L1 = []    # Initialize L1 through L5 as lists
L2 = []
```

```
L3 = []
L4 = []
L5 = []
with open('data1.dat','r') as fnum:
     for currline in fnum:
          t = currline.split()
          L1.append(int(t[0]))  # Use the append method to insert t[0] into L1
          L2.append(int(t[1]))
          L3.append(int(t[2]))
          L4.append(int(t[3]))
          L5.append(int(t[4]))
print(L1)
```

This gives the output

```
[1, 5, 1, 5, 1, 5, 1, 5]
```

Now each list consists of integers. If we need these to be `numpy` arrays instead of standard lists, we can do

```
L1 = np.array(L1)
```

after exiting the `with` structure.

2.1 The `.strip()` Method

Sometimes the `.split()` method alone will not sufficiently isolate the portions of a data file that you would like. The `.strip()` method is used to remove leading and trailing characters from a string. The default character is the space, but you can specify what character you want as an optional input, for example

```
.strip(',')   # removes leading/trailing commas from the string
.strip(':')   # removes leading/trailing colons from the string
.strip('a')   # removes leading/trailing letter a's from the string
```