BasicPlotting

January 29, 2019

0.1 Objective: Read the tab delimited file "integer_list.txt" into a list variable, iterate over the list and sort the values into even and odd numbers, and write an output file for each new list.

We're going to read in a one line, tab-delimited file of integers into a list. When we read the list, it will come in as a list of string representations of numbers, which need to be converted to integers. Next, we need to sort the integers into even and odd values. To write the lists to tab delimited files, we need to concatenate strings of values separated by tab delimiters, then write these strings to a file.

Implementation Algorithm:

1. Read input file `integer_list.txt`
2. Convert list of strings to list of integers (the map() function makes this very easy)
3. Define empty lists to store even and odd numbers
4. Iterate over list, sorting even and odd integers using logical statements
5. . Define empty strings to store output for writing
6. In separate for loops, write output strings by converting each integer in even and odd lists to a string and adding a delimiter. Concatenate this value with the appropriate output string.
7. Write output files.

```
In [1]: # IMPORT PACKAGES
        import math  # functions like sqrt and pi are parth of the math package
        import re  # regular expressions
        import numpy as np  # linearly spaced vector
        import matplotlib.pyplot as plt  # plotting
```

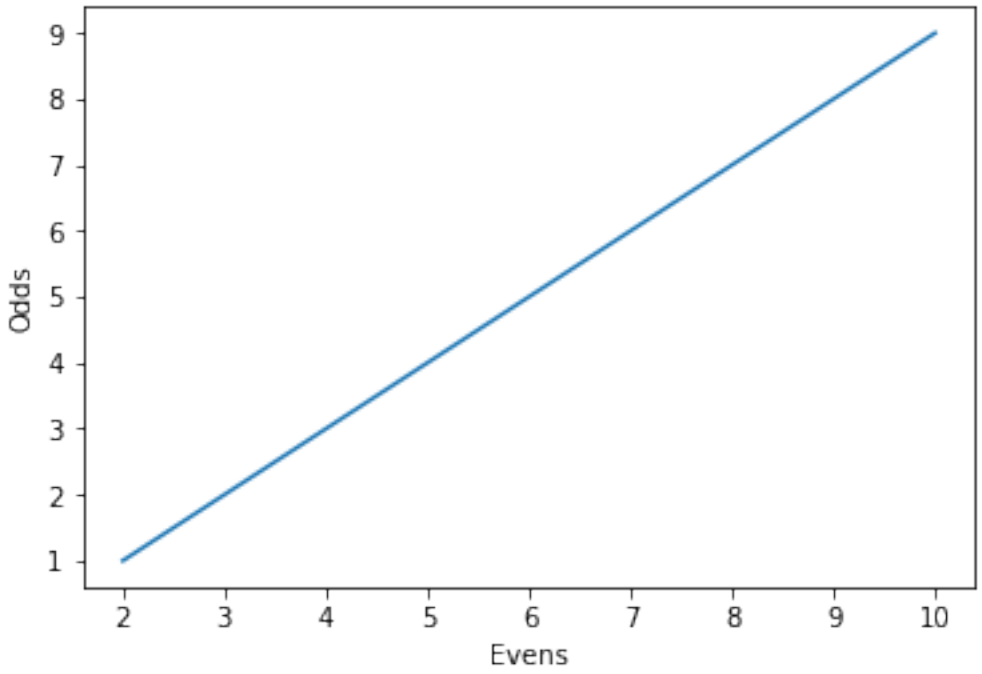
```
In [2]: # READING AND WRITING TEXT FILES

        # open the input file
        f = open('integer_list.txt','r')

        # read the file (it's only one line, so we don't need to read line by line in a loop)
```

```
line = f.readlines()
integer_list = re.split('\t',line[0])

# the list was read in as a list of strings, which we need to convert to numbers
integer_list = map(int, integer_list)  # here we're using the map() function to
# convert all list values at once vs using a loop to convert each value one by one

# iterate over the list of integers, sort, and store values
evens = []  # empty list to store even numbers
odds = []  # empty list to store odd numbers
for number in integer_list:  # initialize for loop to iterate over each number
# in the list
    if number%2 is 0:  # if the number is divisible by 2 with no remainder (even)
        evens.append(number)  # store value
    else:  # if the statement above evaluates to False, the number is odd
        odds.append(number)  # store value

# close the input file
f.close()

# write output files
fevens = open('evens.txt','w')
fodds = open('odds.txt','w')

# write even and odd strings with delimiters
even_string = ''
for even in evens:
    even_string += str(even) + '\t'

fevens.write(even_string)

odd_string = ''
for odd in odds:
    odd_string += str(odd) + '\t'

fodds.write(odd_string)

fevens.close()
fodds.close()
```

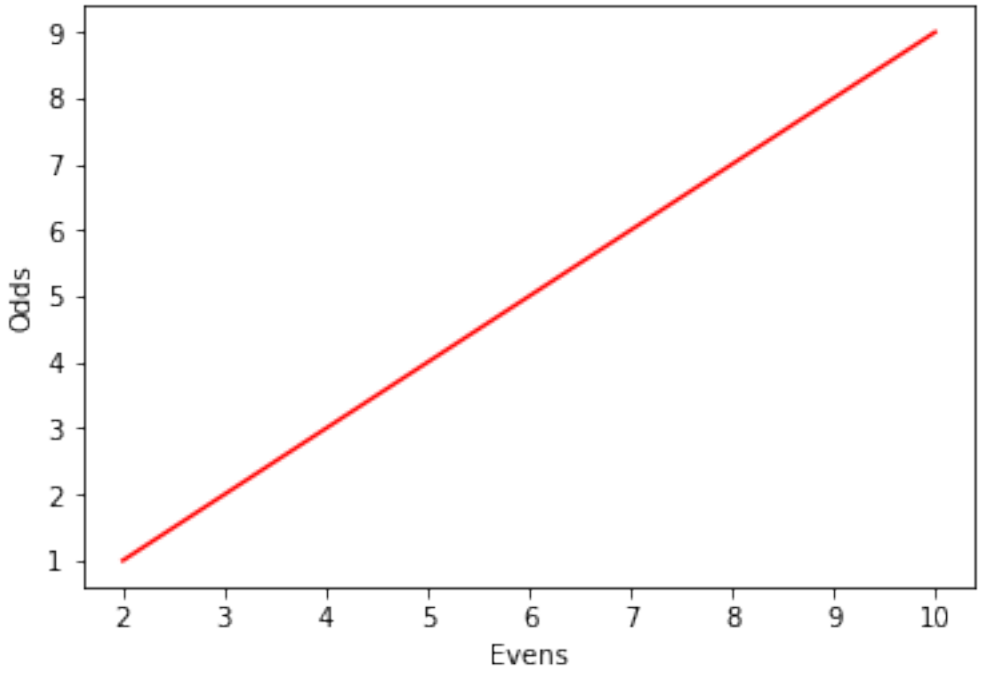
0.2 Plotting

Just for practice, let's make a plot. Matplotlib is the Python plotting package, which is commonly imported using the shortname "plt". To create a basic plot, you need two pieces of information: your **x** and **y** data. These are normally passed to the plotting function as lists, Numpy arrays, or Pandas dataframes, where the respective indicies of each object represent the coordinates of a point. By default, most plotting functions will connect the points in a blue line. You can change the color plotting and/or symbol used by passing an additional argument to the plotting function. Most programming languages will recognize letter-specific shortnames for colors in the CMYK (cyan, magenta, yellow, black) and RGB (red, green, blue) color models, as well as white (w). Additional colors can be passed to the function using an RGB value or hexadecimal code. RGB values are tuples of three values specifying the relative amounts of red, green, and blue needed to create the desired color. Sometimes these values are scaled from 0-255, other times from 0-1. Python accepts RGB values scaled from 0-1. Hexadecimal codes are base 16 strings (RRGGBB) specifying the intensity of a chosen color from darkest (00) to lightest (FF).

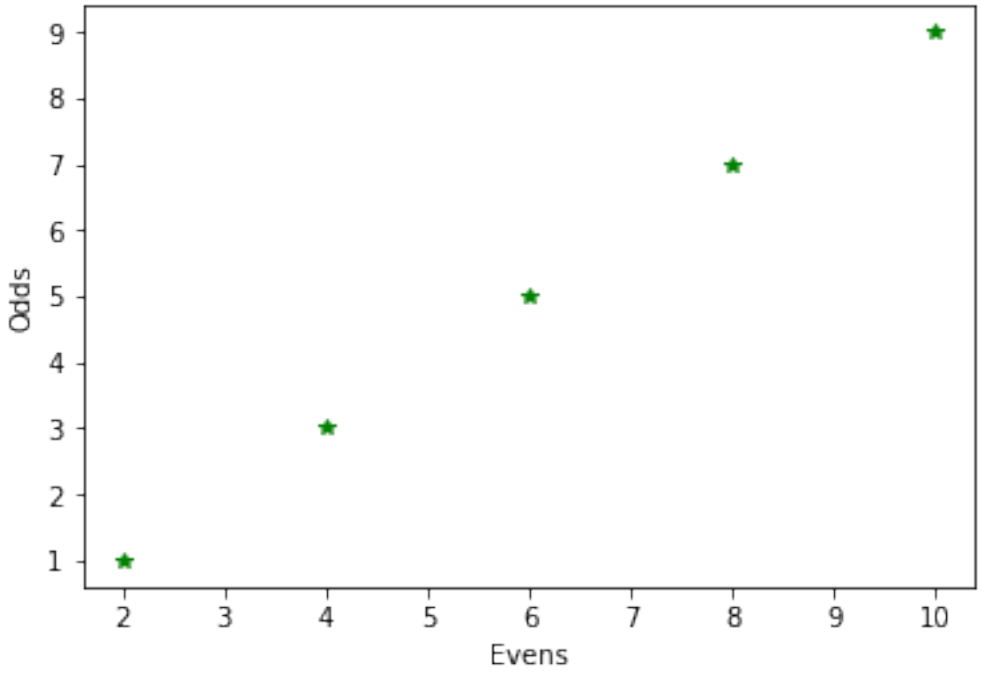
```
In [3]: # plot a blue line - default
        plt.figure(1)  # create figure object
        plt.plot(evens,odds)  # plot x and y data
        plt.xlabel('Evens')  # add a label on the x axis
        plt.ylabel('Odds')  # add a label on the y axis
        plt.show()  # display the plot in the console
```



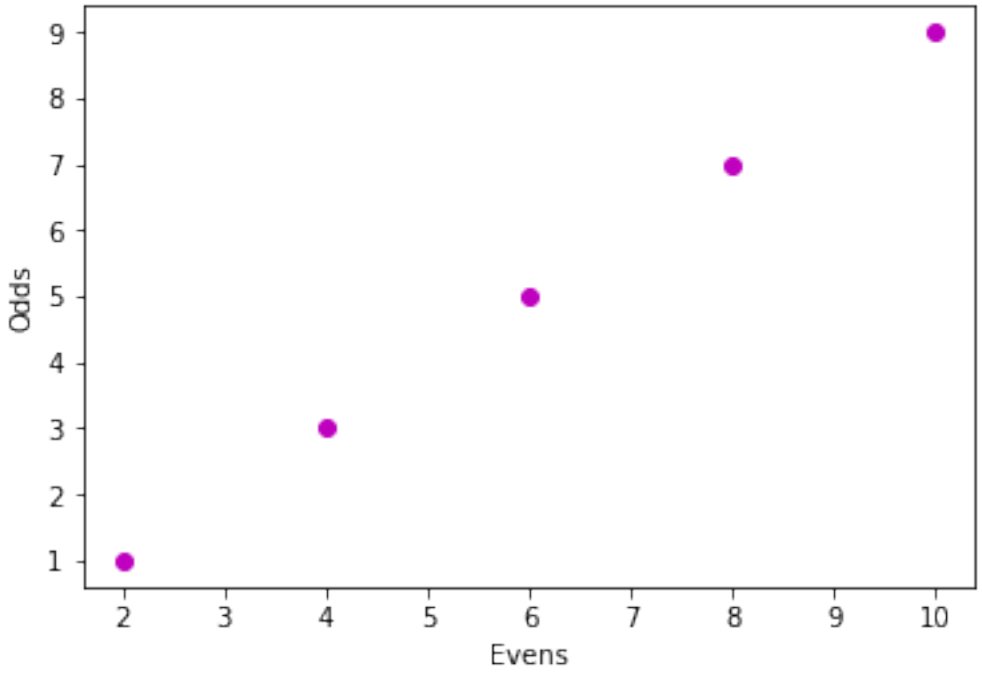
```
# plot a red line
plt.figure(2)
plt.plot(evens,odds,'r')
plt.xlabel('Evens')
plt.ylabel('Odds')
plt.show()
```



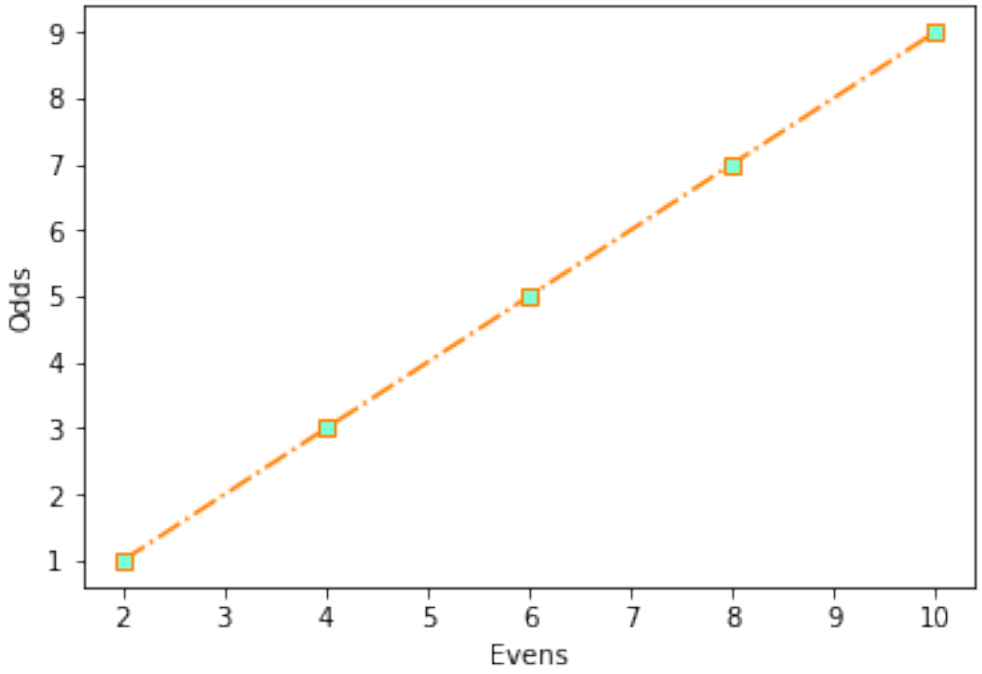
```
# plot points as green stars
plt.figure(3)
plt.plot(evens,odds,'g*')
plt.xlabel('Evens')
plt.ylabel('Odds')
plt.show()
```



```
# plot points as megenta circles
plt.figure(4)
plt.plot(evens,odds,'mo')
plt.xlabel('Evens')
plt.ylabel('Odds')
plt.show()
```



```
# plot points as orange squares
plt.figure(5)
plt.plot(evens,odds,'s',color=(1,0.5,0),linestyle='-.',markerfacecolor='aquamarine')
plt.xlabel('Evens')
plt.ylabel('Odds')
plt.show()
```



```
# plot points as golden yellow triangles outlined in blue
# and connected with a dotted line
plt.figure(6)
plt.plot(evens,odds,'^',color='#f1c40f',linestyle=':',markersize=14,markeredgecolor='b')
plt.xlabel('Evens')
plt.ylabel('Odds')
plt.show()
```