Rendering Plots With Matplotlib

CSE/IT 107L

NMT Department of Computer Science and Engineering

> "The greatest value of a picture is when it forces us to notice what we never expected to see."
>
> —John Tukey

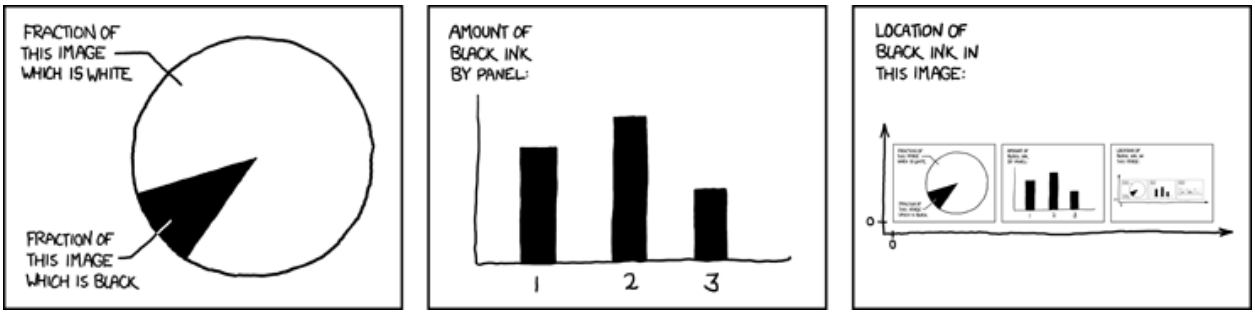


Figure 1: Self description. `https://xkcd.com/688/`

Contents

Introduction

In this lab you will be learning how to utilize built in libraries to generate graphics. You will be learning how to use the library matplotlib to generate and draw simple mathmatic plots to represent given data and information. matplotlib is a powerful tool that gives you much more control over how graphs are drawn. This lab demonstrates the power and practicality of the Python language and its many libraries.

1 Keyword Arguments

Before we start, you should know about a commonly used Python feature called keyword arguments. As you know, functions may take zero or more required arguments, these are called *positional arguments*. There is a special syntax that can be used to define *optional arguments*.

For example, this function takes either one or two arguments. The second argument has a default value of 5.

```
>>> def f(fst, snd = 5):
...     return fst * snd
>>> f(10)
50
>>> f(10, 6)
60
>>> f()
Traceback (most recent call last):
  File ``<stdin>'', line 1, in <module>
TypeError: f() missing 1 required positional argument: 'fst'
```

An important feature is that keyword arguments may be labeled and listed in any order when calling the function. For example, the following function takes two keyword arguments. There are five valid ways to call this function. You may pass no arguments, then the default values aare used. You may pass either argument by itself or you may pass both arguments in any order.

```
>>> def p(greeting="Hello", subject="world"):
...     return "{} {}".format(greeting, subject)
>>> p()
"Hello world"
>>> p(greeting="Goodbye")
"Goodbye world"
>>> p(subject="humans")
"Hello humans"
>>> p(greeting="Bonjour", subject="le monde")
"Bonjour le monde"
>>> p(subject="le monde", greeting="Bonjour")
"Bonjour le monde"
```

2 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats and interactive environments. We will be using the `matplotlib.pyplot` library. It is commonly renamed to `plt` using this code:

```
>>> import matplotlib.pyplot as plt
```

It's a good idea to keep the Matplotlib documentation on hand as you read these examples. All of the following matplotlib functions are documented online at:

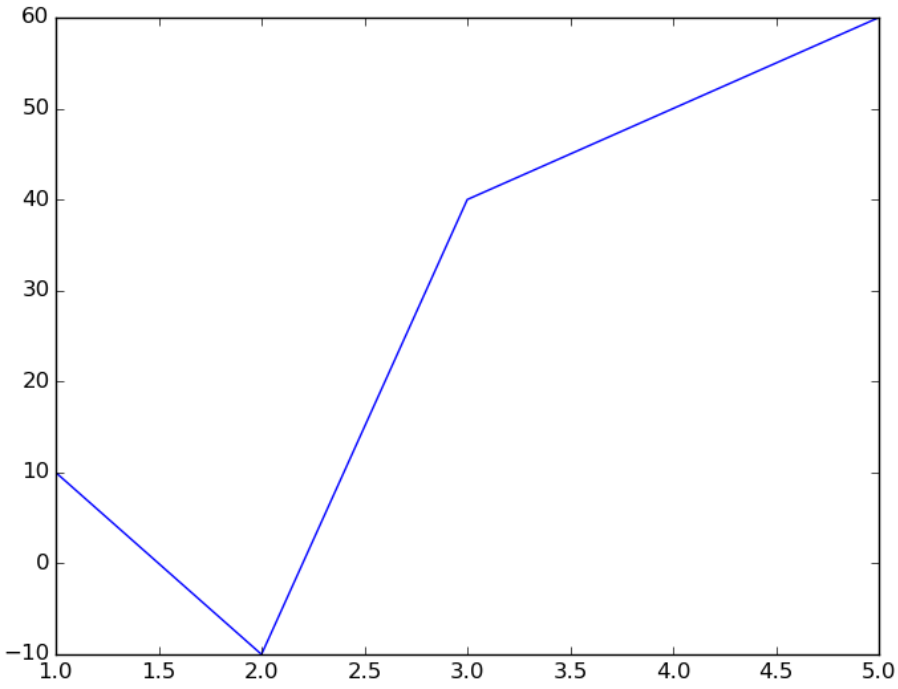
`http://matplotlib.org/api/pyplot_api.html`

2.1 Plotting Points with the `plot` Function

`plt.plot` is a versatile function that is used to plot pairs of numeric values. Matplotlib does not take a list of tuples representing points, instead it takes two lists. The list of values on the x-axis are the first argument and the list of y-axis values form the second argument. One of the lists must be in ascending order.

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([1, 2, 3, 4], [10, -10, 40, 50])
>>> plt.show()
```

The `plt.show()` function shows everything drawn by previously called plotting functions. This image was draw by the previous Matplotlib code:

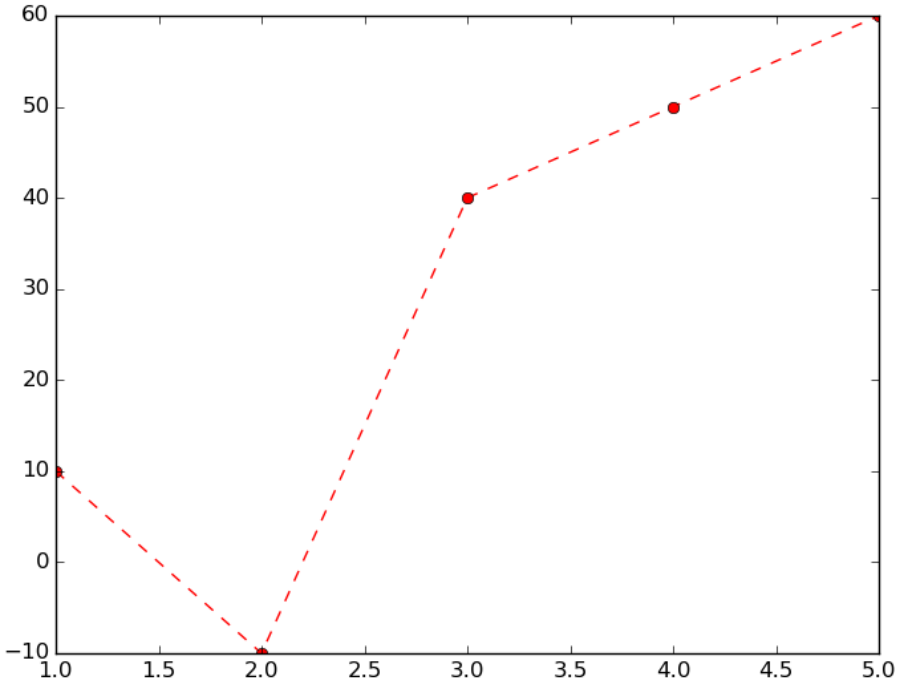


Matplotlib plots can be modified in many ways. The documentation has a table of keyword arguments for this function. The columns are Property and Description. Let's try some of those. The `color`, `linestyle`, and `marker` keyword arguments can be used to draw a differently styled line. As of March, 2016 the Matplotlib documentation for these functions, colors, and line styles are located at

`http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot`

http://matplotlib.org/api/colors_api.html
http://matplotlib.org/api/lines_api.html#matplotlib.lines.Line2D.set_linestyle

Here's the same plot with a red, dashed line and o's on every data point.




```
>>> import matplotlib.pyplot as plt
>>> plt.plot([1, 2, 3, 4, 5], [10, -10, 40, 50, 60],
...          color='red', linestyle='dashed', marker='o')
>>> plt.show()
```

2.2 Describing your Plots

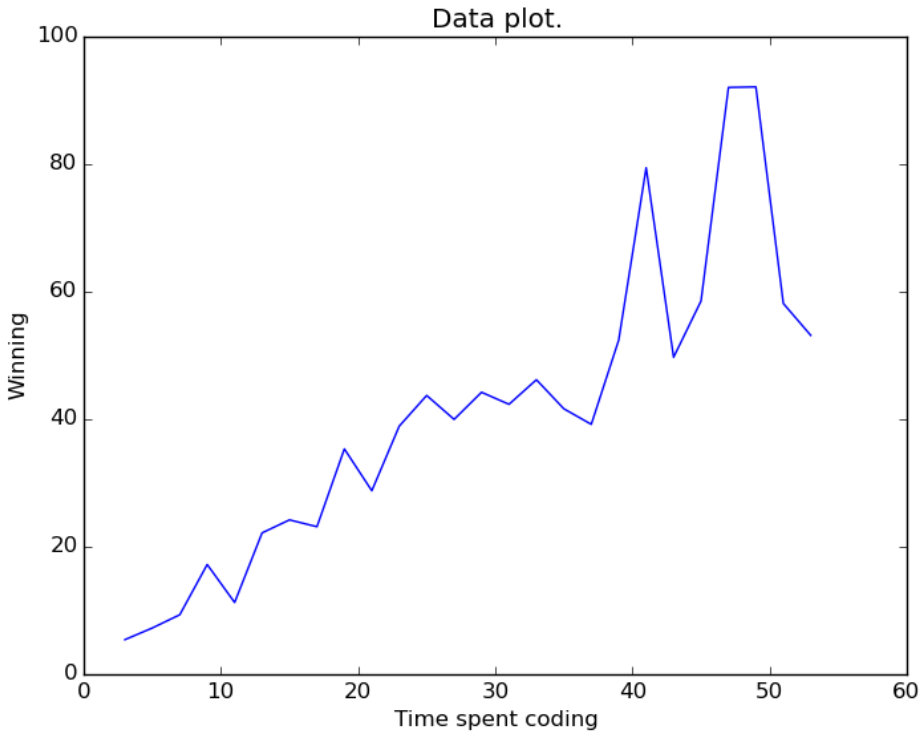
See these links for documentation:

http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.xlabel
http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.ylabel
http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.title
http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.subplot

Plots can be labeled and titled using the `plt.xlabel`, `plt.ylabel`, and `plt.title` functions. Here's an example:

```
import random
import matplotlib.pyplot as plt
```

```
x = [x + 3 for x in range(0,52,2)]
y = [random.random() * x + x for x in x]

plt.plot(x, y)
plt.xlabel('Time spent coding')
plt.ylabel('Winning')
plt.title('Data plot.')
plt.show()
```

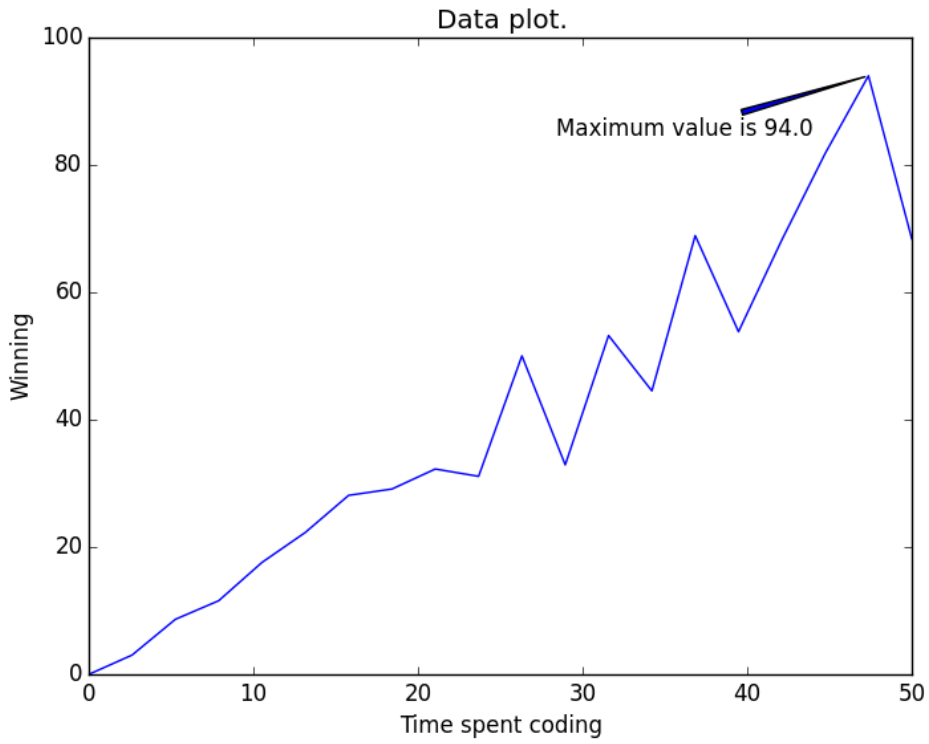


What if you want to call attention to an individual data point? Then you should consider the `plt.annotate` function to add annotations to plots. In this example we indicate the maximum y-value in a randomly generated set of data: This code produces a labeled, annotated plot.

Annotation text is set with the `s` argument. The arrow is styled using the `arrowprops` argument. The `xy` and `xytext` keyword arguments are used to place the arrow and text.

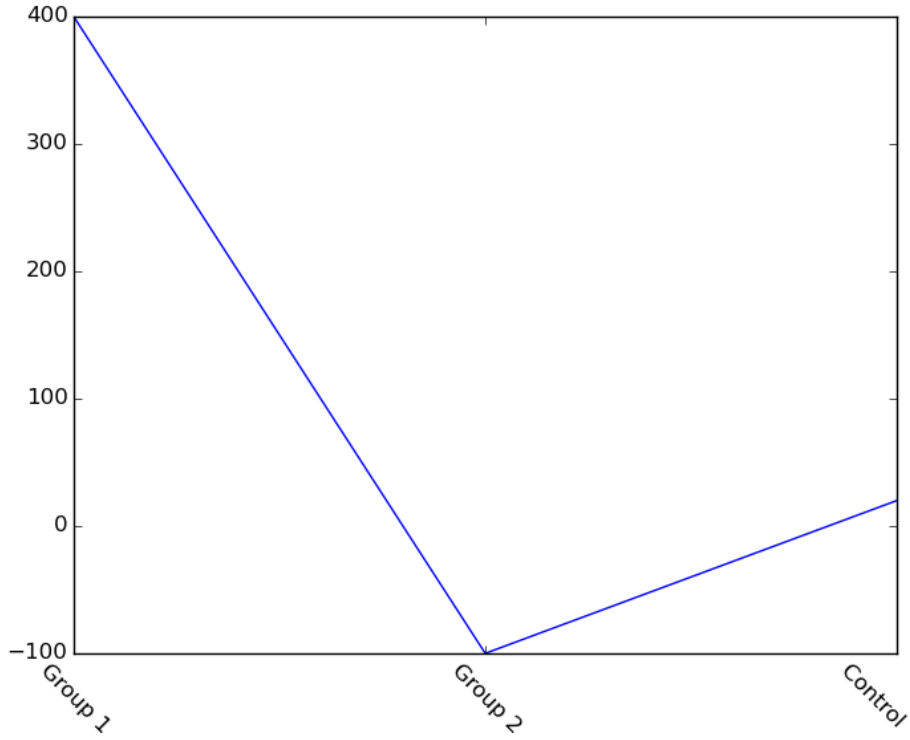
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 50, num=20)
y = np.random.random(20) * x + x

# find maximum
maximum_index = np.argmax(y)
maximum = y[maximum_index]

plt.plot(x, y)
```

```
plt.xlabel('Time spent coding')
plt.ylabel('Winning')
plt.title('Data plot.')

# add annotation at maximum point
plt.annotate(s='Maximum value is {:.3}'.format(y[maximum_index]),
             xy=(x[maximum_index], y[maximum_index]),
             xytext=(x[maximum_index] * 0.6, y[maximum_index] * 0.9),
             arrowprops={'arrowstyle': "wedge,tail_width=0.25"})

plt.show()
```



The tick labels can be changed using the `plt.xticks` and `plt.yticks` functions. The first argument is a list of numbers and the second argument is the list of labels for each value. For example,

```
>>> plt.plot([1, 2, 3], [400, -100, 20])
>>> plt.xticks([1, 2, 3], ["Group 1", "Group 2", "Control"], rotation=-45)
>>> plt.show()
```


400
300
200
100
0
−100
Group 1
Group 2
Control

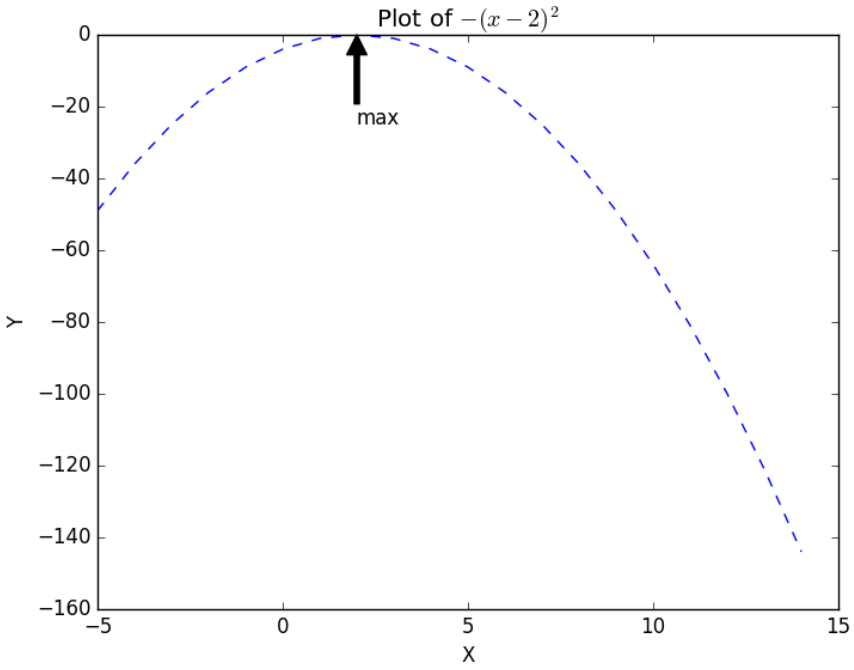
3 Exercises

Labels

Remember to label your axes using `plt.xticks()` and `plt.yticks()` and title your graph using `plt.title()`. Read the documentation to find out more.

Exercise 3.1 (plotpoints.py).
Plot the following points. Add labels, a title, and an annotation to the plot as shown in the example. Make sure to draw a dashed blue line. The annotation contains the text "max" and is located at the point (2, 20). The annotation text must be slightly below and the arrow connecting the text and point must be black and have width set to 2.

```
xs = range(-5, 15)
ys = [-(x - 2) ** 2 for x in xs]

...

plt.title('Plot of $-(x - 2)^2$')
```



You may use `arrowprops={'color': 'black', 'width': 2}` when drawing the annotation.

Submitting

You should submit your code as a `.zip` that contains all the exercise files for this lab. The submitted file should be named

`cse107_groupNUMBER_prelab11.zip`

Or if you are working alone, the submitted file should be named:

`cse107_firstname_lastname_prelab11.zip`

Upload your .zip file to Canvas.

List of Files to Submit

Exercises start on page 7.