Lists

Can you think of a list you have made?

We are going to go back to our list of numbers and practice some more!

In [1]:
```
# now lets do lists
FirstList=[2,5,10,15,20]
```

In [2]:
```
FirstList
```

Out[2]:
```
[2, 5, 10, 15, 20]
```

In [3]:
```
FirstList[0]
```

Out[3]:
```
2
```

In [4]:
```
FirstList[4]
```

Out[4]:
```
20
```

What did I just do? I accessed the list. But I accessed the list one at a time.

But how come we used a zero to access the list? Wouldn't the first item be number 1?

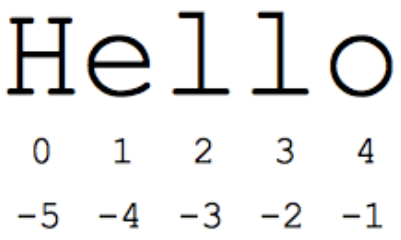
Look at the image below. It is for a list of letters or a string but the numbering is the same? Is this confusing?

(remember the picture is there for you to learn, not for you to add)

DO NOT PROGRAM THIS NEXT CELL. JUST LOOK!

In [5]:
```
from IPython.display import Image
Image('https://developers.google.com/edu/python/images/hello.png')
```

Out[5]:



I just inserted one of the most critical programming issues in on you. this idea of indexing. Where are things stored in lists and strings. This can be confusing when you first begin! So lets stop and talk about bins and numbering. A lot is written about strings and lists. Here is a google link

To try indexing just start indexing! Lets do it! For each of them try and write down what you think will happend before you program it. Remember it is [Start:Stop:Skip] but you can leave some out.....

I am going to repeat the last sentence because it is important!

to look at a list you use this nomenclature. [Start:Stop:Skip]

If you leave one out it will assume, beginning, end, all.

It stops 1 before the end. So if the end is 5 it will print the 4th cell

So for each cell below write down what you think will happen. Then remove the comment # and run it and see if you are correct.

The best way to learn is by doing! So lets get going! Some will give errors!

Plus you don't need to use a new cell each time. You can just go back and change the cell and rerun it.

I just need new cells so I can show you everything!

In [7]:
```
#print (FirstList[:])
```

In [6]:
```
# FirstList[1]
```

In [40]:
```
#FirstList[5]
```

In [41]:
```
#FirstList[-1]
```

In [42]:
```
#FirstList[-5]
```

In [43]:
```
#FirstList[-6]
```

In [44]:
```
#print (FirstList[0:5])
```

In [9]:
```
#print (FirstList[0:10])
```

In [46]:
```
#print (FirstList[0:4])
```

In [47]:
```
#print (FirstList[:0])
```

In [48]:
```
#print (FirstList[:2]) #this lists from start up to an not including spot 2
```

In [49]:
```
#print (FirstList[0:]) # this is 0 to end
```

In [50]:
```
#print (FirstList[1:]) #this is 1 to end
```

In [51]:
```
#print (FirstList[0:5:2]) #this is 0 to 5 by 2's
```

In [52]:
```
#print (FirstList[::2]) #this is beginning to end by 2's
```

In [53]:
```
#print (FirstList[:::])
```

In [54]:
```
#print (FirstList[0:5:3]) #this is 0 to 5 by 3's
```

In [55]:
```
#print (FirstList[0:5:-1]) #this is beginning to end by -1 and won't work
```

In [56]:
```
#print (FirstList[5:0])
```

In [57]:
```
#print (FirstList[5:0:-1]) #this is end to beginning but stops before 0
```

In [58]:
```
#print (FirstList[::-1]) # a better way to go in reverse
```

In [59]:
```
#print (FirstList[5:-1:-1])
```

In [60]:
```
#print (FirstList[5:-6:-1])
```

In [61]:
```
#print (FirstList[-1:-6:-1])
```

In [62]:
```
#print (FirstList[-1])
```

In [63]:
```
#print (FirstList[5:0])
```

In [64]:
```
#print (FirstList[-2:])
```

Now Lets start using Lists!

Graphs and List

We want to learn more about lists and to make graphs with our central park monthly data. What kind of graphs would you want to make???

This is NOT the big data set from the first class. This is monthly data and much simpler

Here is our monthly precip if you forgot!

month	PRCP
1	3.463263
2	3.334789
3	3.995651
4	3.697536
5	3.755582
6	3.674013
7	4.253052
8	4.348163
9	3.870993
10	3.646531
11	3.527670
12	3.624314

Lets use this to make a list

I am doing a little trick on you that we will learn more about later. Since we will be using the numbers I am making the list of numbers a numpy array. This allows us to work with the numbers more easily. Numpy is

a python package or libary made for numerical work.

In [10]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
#import matplotlib as mpl
```

Github

I am going to assume you are lazy and don't won't to type in the next cell. You can cheat and download my notebook from github and copy and paste. This notebook that you are readig on paper you can go download and thus can look at how markdown is down and copy things when needed. BUT type as much as you can. It really helps you lean. Here is how to download.

1. Use the link on the class web page to get to my gitbub page
2. Or you can google bmaillou github python and then choose the file
3. github shows you the notebook. To download right click on raw. click "save link as" and save to your python folder
4. I find it works best with chrome. If you get a .txt extension added to your file this will get you in trouble.
5. Now you can open the notebook in jupyter and see how I do everything.
6. But don't copy and paste everything. You learn by typing!
7. I am repeating. Only copy and paste long lists of numbers otherwise type and learn!

In [11]:
```
monthly_precip=np.array([3.46,3.33,3.99,3.69,3.75,3.67,4.25,4.34,3.87,3.64,3.52,3.62])
print (monthly_precip)
```

```
[3.46 3.33 3.99 3.69 3.75 3.67 4.25 4.34 3.87 3.64 3.52 3.62]
```

In [12]:
```
monthly_precip[1] #lets review and go through how to access a list.
```

Out[12]:
```
3.33
```

Go back above and look at how you accessed lists. Now practice on this list.
Can you print out February

In [ ]:
```

```

Can you show just the even months?

In [2]:
```

```

The odd months?

In [ ]:
```

```

Can you do it backwards from December to January?

In [ ]:
```

```

Can you just show the month you were born?

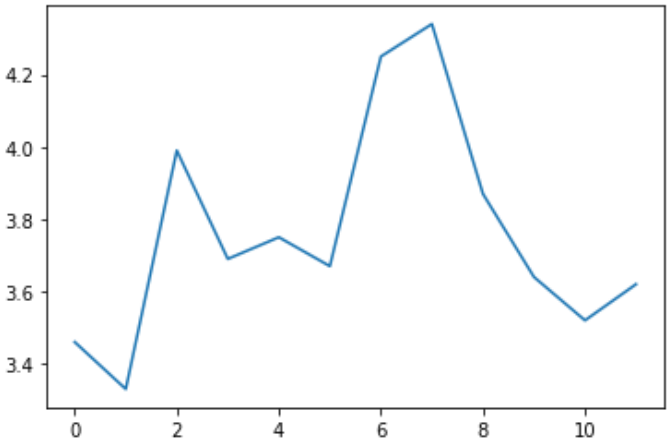
In [ ]:
```

```

Now lets do a few plots.

In [13]:
```
plt.plot(monthly_precip)  # what is wrong with the x axis.
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x7f80a67d5880>]
```



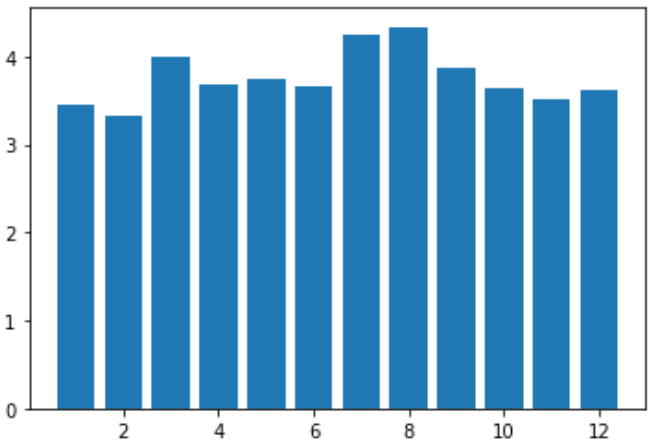
What is wrong with the x-axis? What is being plotted?

It is just plotting the number of the list not the number of the month! Instead we need to give it an x-axis with the month number of 1-12 and not 0-11. Pyhton starts counting at 0! I am using our numpy arange function. We will do more on this next class.

Also, plt.bar() is a function. We are passing data to that function to make the graph. We are passing X and Y values.

In [14]:
```
plt.bar(np.arange(1,13),monthly_precip)
```

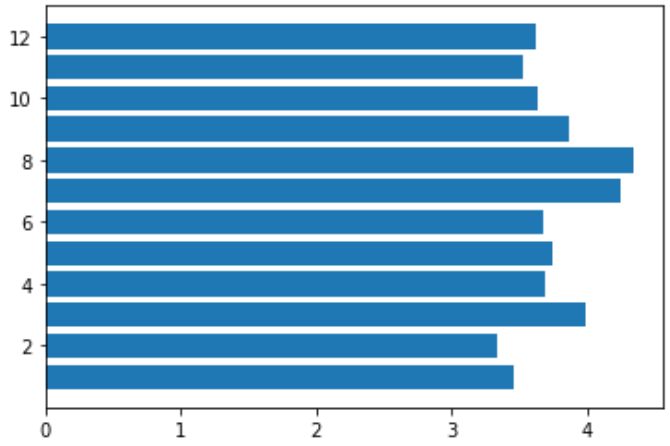
Out[14]:
```
<BarContainer object of 12 artists>
```



There are many types of plots you can make! Look at matplotlib You can make almost anything you can think of.

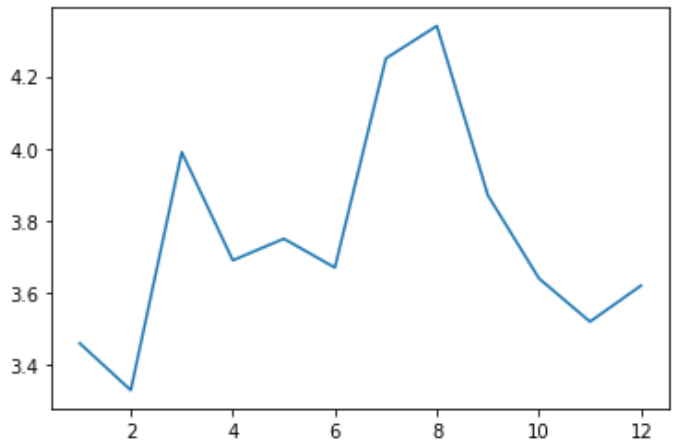
In [15]:
```
plt.barh(np.arange(1,13),monthly_precip)
```

Out[15]:
```
<BarContainer object of 12 artists>
```



In [16]:
```
plt.plot(np.arange(1,13),monthly_precip)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7f80a6c07c40>]
```

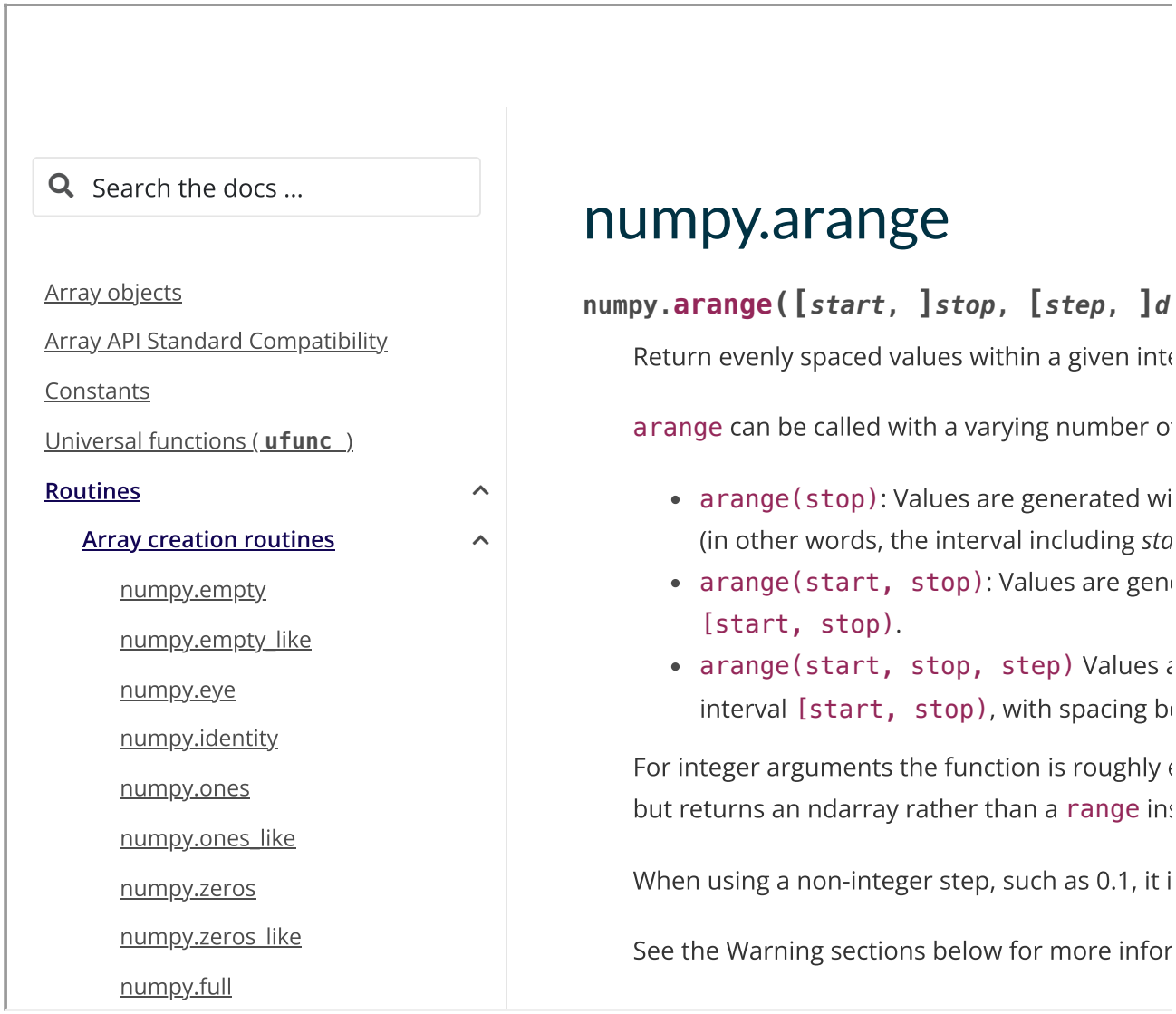


What is range and what did I just do to get the axes to change?

Lets see if we can figure this out. Pyhton has a built in function called range But I find the numpy version works much better np.arange But as an aside if we want to get fancy we could put a webpage into our ipython notebook! Remember np.arange is a function you are calling. More on this to come next time. Let's see what we can learn today.

In [5]:
```
from IPython.display import HTML
HTML('<iFrame src=https://numpy.org/doc/stable/reference/generated/numpy.arange.html wid
```

Out[5]:



More easilty we could get help. Thre are 2 easy ways.

1. type ?range or ?and the function and press shift-enter
2. you start typing the function and add the first brakcet and press shift-tab. Then press the little triangle to see the help. try both
3. This is a function and not a list. Functions use parantheses and commas. We will learn more about this!
4. Remember. np.arange is a function. This is weird your first few days seperating lists and functions and other things. But be patient you will get there.

In [17]:
```
?np.arange
```

In [19]:
```
np.arange(
```

Out[19]:
```
<function numpy.arange>
```

Can you count to 20?

In [20]:
```
np.arange(20)
```

Out[20]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

Can You include 20?

In [ ]:

Can you count to 20 by 2's?

In [ ]:

Can you count down from 20 to 0 and include 0?

In [ ]:

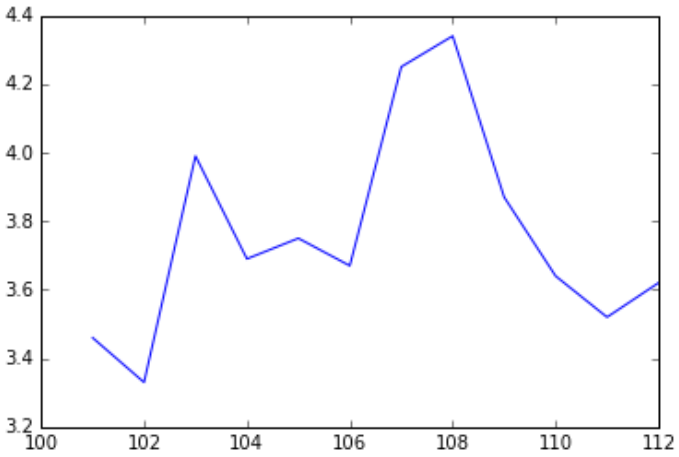
That was a nice aside but back to our plots? What did I do?

use tab help on plt.plot(and see what the first two entries are.)

can you make a plot where the x asix is from 101-112?

In [5]:

Out[5]:
```
[<matplotlib.lines.Line2D at 0x78333c8>]
```



In [16]:

Matplotlib has a lot of plot types. In the future we will explore more

In [21]:

In [ ]:

Lets get fancier and add the months to the x-axis

In [11]:
```
#but we need to do better
# we could add the months
months=['Jan','Feb','Mar','Apr','May','June','July','Aug','Sep','Oct','Nov','Dec']
print (months)
```

```
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

List properties

I just did a trick on you I just showed that lists do not need to be numbers. I made the list a set of strings. In fact lists are muteable and can be mixed. so lets read and play with lists some!

Here is a nice overview Here is the python page Here is the more in depth python page plus remember you could tab help. so you could type months. and hit tab

In [13]:
```
months.reverse()
print (months)
months.reverse()
print (months)
```

```
['Dec', 'Nov', 'Oct', 'Sep', 'Aug', 'July', 'June', 'May', 'Apr', 'Mar', 'Feb', 'Jan']
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

Can you just print Aug

In [ ]:
```

```

Can you change the list to spell out August by setting it equal to a new value? Something = something...

In [ ]:
```

```

Print the list again and make sure it shows august

In [14]:
```
print (months)
```

```
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

You can also call functions on lists. for example what is the length? Use

In [15]:
```
len(months)
```

Out[15]:
```
12
```

Can you set August to 8? This is really easy. You can just take months[7]= and set it equal to whatever you want!

In [ ]:
```

```

Lets set months[7] back to 'Aug' and see if we can make a prettier graph!

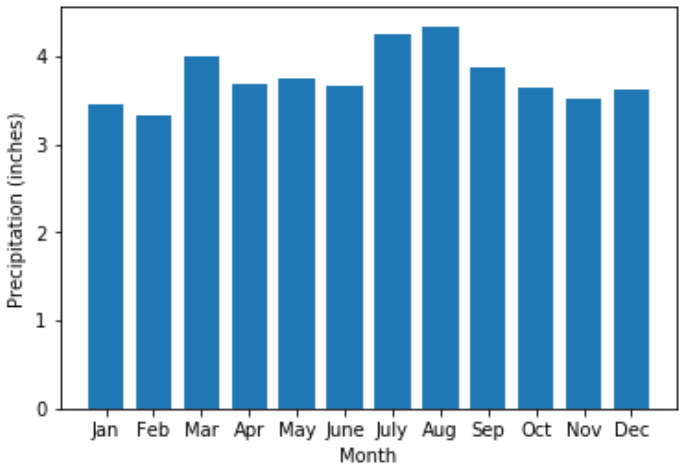
In [ ]:
```

```

I am going to throw some fancy plotting at you! What we do is use subplots to set fig and ax. These are variables that control the plot. I think of fig as the page the plot is on and I think of ax as the plotting parameters. Then we can set a lot of things within that. We are going to set the axis titles. I will also rename the months. This takes some digging but the chart is a nice reference!. But start to try and remember these tricks. They will help us throughout the semester

In [32]:
```
#http://matplotlib.org/examples/lines_bars_and_markers/barh_demo.html
fig,ax=plt.subplots()
width=0.75
ax.bar(np.arange(12),monthly_precip,width,align='center')
ax.set_xticks(np.arange(12))
ax.set_xticklabels(months,minor=False)
ax.set_xlabel('Month')  #added in
ax.set_ylabel('Precipitation (inches)')  #added in
```

Out[32]:
```
Text(0,0.5,'Precipitation (inches)')
```



Now we can also do a little math!

can we convert the list so it is a percent of annual rainfall?

In [7]:
```
#could we plot the percent rain?
monthly_percent=monthly_precip/np.sum(monthly_precip)*100
```

In [8]:
```
type(monthly_precip)
```

Out[8]:
```
numpy.ndarray
```

In [9]:
```
#how much rain falls each year?
np.sum(monthly_precip)
```

Out[9]:
```
45.129999999999995
```

In [40]:
```
monthly_percent
```

Out[40]:
```
array([ 7.66674053,  7.3786838 ,  8.84112564,  8.17637935,  8.30932861,
        8.13206293,  9.41723909,  9.61666297,  8.57522712,  8.0655883 ,
        7.79968979,  8.02127188])
```

Can you make a nice plot showing the monthly percent precipitation instead of amount?

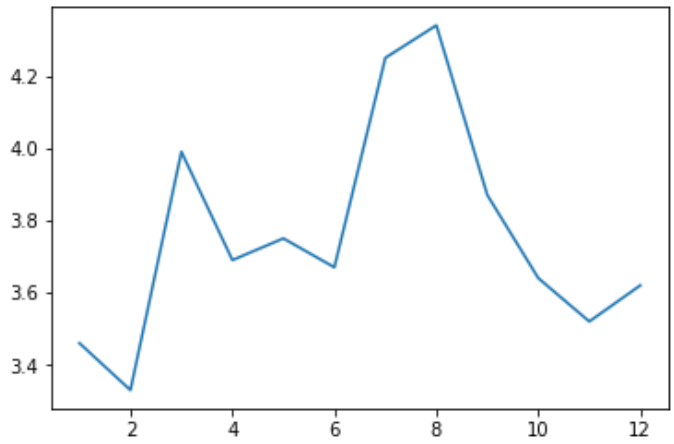
In [ ]:
```

```

How to make line graphs nicer. If not this will also get repeated later or is easy to follow after class for the homework.

In [33]:
```
#http://matplotlib.org/examples/lines_bars_and_markers/barh_demo.html
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip)
```

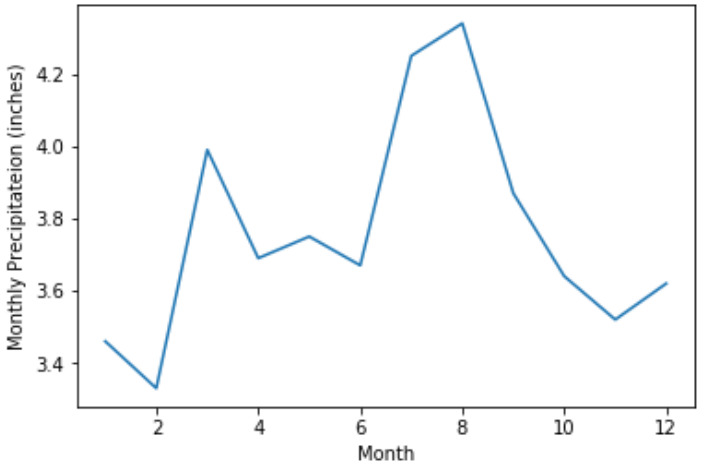
Out[33]:
```
[<matplotlib.lines.Line2D at 0x2545cef8eb8>]
```



We can add axis titles

In [34]:
```
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip)
ax.set_xlabel('Month')
ax.set_ylabel('Monthly Precipitateion (inches)')
```

Out[34]:
```
Text(0,0.5,'Monthly Precipitateion (inches)')
```

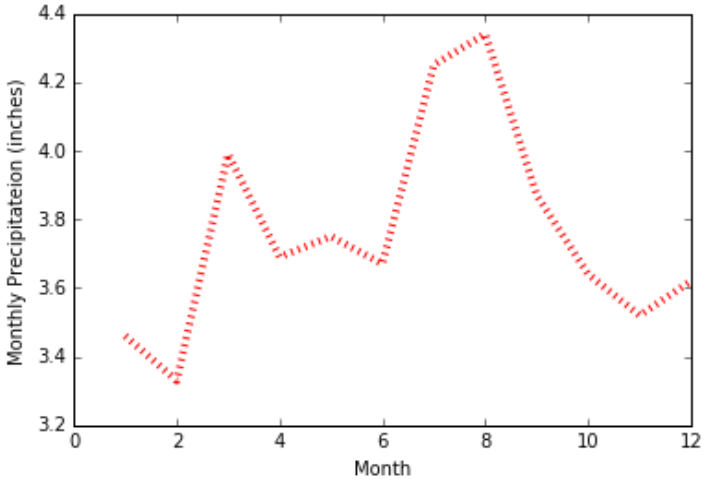


Line styles
https://matplotlib.org/examples/lines_bars_and_markers/line_styles_reference.

You can also google matplotlib colors!

In [15]:
```
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip,linestyle=':',color='red',linewidth=4)
ax.set_xlabel('Month')
ax.set_ylabel('Monthly Precipitateion (inches)')
```

Out[15]:
```
<matplotlib.text.Text at 0x8934630>
```

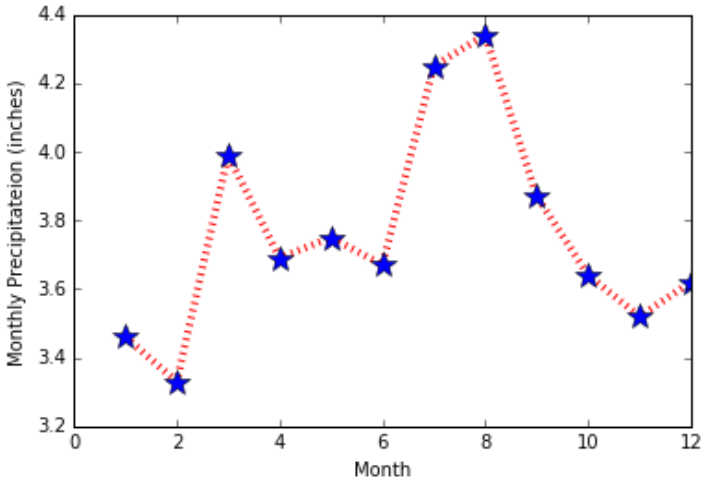


Markers

https://matplotlib.org/examples/lines_bars_and_markers/marker_reference.html
https://matplotlib.org/examples/lines_bars_and_markers/marker_fillstyle_reference.html title title

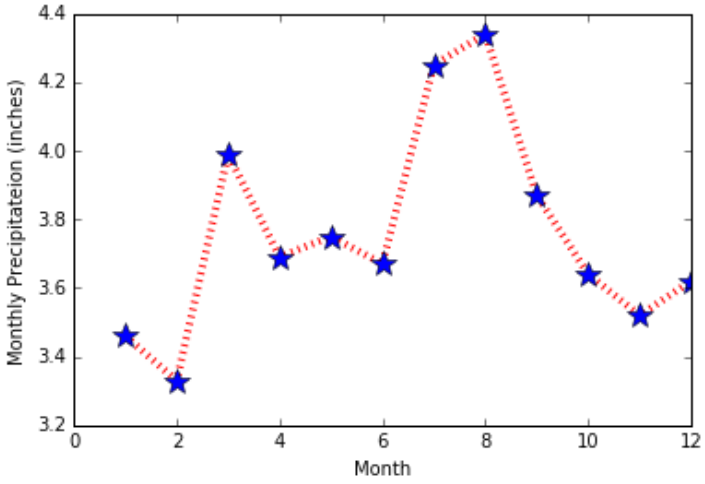
In [27]:
```
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip,linestyle=':',color='red',linewidth=4,marker='*',mark
ax.set_xlabel('Month')
ax.set_ylabel('Monthly Precipitateion (inches)')
```

Out[27]:
```
<matplotlib.text.Text at 0x9965cc0>
```



In [3]:
```
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip,linestyle=':',color='red',linewidth=4,marker='*',mark
ax.set_xlabel('Month')
ax.set_ylabel('Monthly Precipitateion (inches)')
```

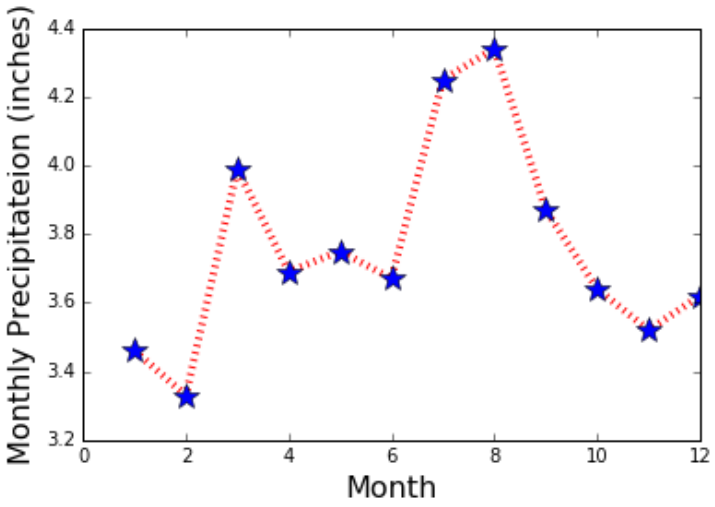
Out[3]:
```
<matplotlib.text.Text at 0x6eab3c8>
```



you can also change label fontsize

In [12]:
```
fig,ax=plt.subplots()
ax.plot(range(1,13),monthly_precip,linestyle=':',color='red',linewidth=4,marker='*',mark
        ,markerfacecolor='blue',label=16)
ax.set_xlabel('Month',fontsize=16)
ax.set_ylabel('Monthly Precipitateion (inches)',fontsize=16)
```

Out[12]: <matplotlib.text.Text at 0x89173c8>



You can do a lot more. More to come!

In [ ]:

In [ ]: