Useful Functions

The following are some useful functions in matplotlib library that you may want to use:

```
matplotlib.pyplot.bar(left, height, width=0.8, bottom=None, hold=None, color, align)
```

Make a bar plot with rectangles bounded by: left, left + width, bottom, bottom + height (left, right, bottom and top edges). For this exercise, you are not required to use `bottom` and `hold` option, but you need to use `color` and `align` option.

The return value of `matplotlib.pyplot.bar()` is `matplotlib.patches.Rectangle` instances.

```
import matplotlib.pyplot as pyplot

bar1 = pyplot.bar(1, 10, width=1, color='yellow', align='center')
bar2 = pyplot.bar(2, 15, width=1, color='red', align='center')

pyplot.axis([0, 4, 0, 18])
pyplot.show()
```

The result should be:
Interestingly, the variable left and height for matplotlib.pyplot.bar() can be list. Consider the following example:

```python
import matplotlib.pyplot as pyplot

pos1 = [1, 3.5]
pos2 = [2, 4.5]
data1 = [10, 16]
data2 = [15, 13]

bar1 = pyplot.bar(pos1, data1, width=1, color='yellow', align='center')
bar2 = pyplot.bar(pos2, data2, width=1, color='red', align='center')
pyplot.axis([0, 6, 0, 18])
pyplot.show()
```

The result should be:

![Bar chart example](chart.png)

To distinguish different data, we need a legend in the graph. A legend contains a list of the variables appearing in the chart and an example of their appearance. This information allows the data from each variable to be identified in the chart. The following graph is an example of legend.

```python
matplotlib.pyplot.legend()
```
The function `matplotlib.pyplot.legend()` places a legend on the axes. To make a legend for lines which already exist on the axes (via `plot` for instance), simply call this function with an iterable of strings, one for each legend item. For example:

```python
import matplotlib.pyplot as pyplot

pos1 = [1, 3.5]
pos2 = [2, 4.5]
data1 = [10, 16]
data2 = [15, 13]

bar1 = pyplot.bar(pos1, data1, width=1, color='yellow', align='center')
bar2 = pyplot.bar(pos2, data2, width=1, color='red', align='center')

#pyplot.legend() in this case accepts three parameters.
#The first parameter is the set of existing bars.
#The second is the legend text corresponds to the variables appearing in the graph.
#The third is the location of this legend placed in the graph.
pyplot.legend((bar1, bar2), ('Sample One', 'Sample Two'), loc=1)
pyplot.axis([0, 6, 0, 18])
pyplot.show()
```

The location of the legend can be specified by the keyword argument `loc`, either by string or a integer number.
The result of the above example should be:

```
import matplotlib.pyplot as pyplot

pos1 = [1, 3.5]
pos2 = [2, 4.5]
pos3 = [1.5, 4]
data1 = [10, 16]
data2 = [15, 13]

names = ['test1', 'test2']

bar1 = pyplot.bar(pos1, data1, width=1, color='yellow', align='center')
bar2 = pyplot.bar(pos2, data2, width=1, color='red', align='center')

pyplot.legend((bar1, bar2), ('Sample One', 'Sample Two'), loc=1)
pyplot.xticks(pos3, names)
```

Matplotlib.pyplot.xticks(*args, **kwargs)

Get or set the x-limits of the current tick locations and labels. The function call will return locs, labels where locs is an array of tick locations and labels is an array of tick labels.
The result should be:

![Bar chart with labels](image)

```
mathtext
```

**matplotlib.pyplot.setp(*args, **kwargs)**

Set a property on an artist object. matplotlib supports the use of `setp()` ("set property") and `getp()` to set and get object properties, as well as to do introspection on the object.

```python
... 
pyplot.setp(pyplot.xticks()[1], rotation=15)
```

Recall that the `pyplot.xticks()` will return `locs` and `labels`, where `locs` is an array of tick locations and `labels` is an array of tick labels. In the example we have `pyplot.xticks()[1]`, which means `labels` is accessed in the `pyplot.setp()` function. The second parameter in the example is `rotation=15`, which means `pyplot.setp()` will rotate `labels` by 15 degree. The result is:
pyplot.show()

Display what you draw in a figure.

The code snippet for the above example is:

```python
import matplotlib.pyplot as pyplot

pos1 = [1, 3.5]
pos2 = [2, 4.5]
pos3 = [1.5, 4]
data1 = [10, 16]
data2 = [15, 13]

names = ['test1', 'test2']

bar1 = pyplot.bar(pos1, data1, width=1, color='yellow', align='center')
bar2 = pyplot.bar(pos2, data2, width=1, color='red', align='center')

pyplot.legend((bar1, bar2), ('Sample One', 'Sample Two'), loc=1)
pyplot.xticks(pos3, names)
```
Note: This example is just code snippet, not a function!

`pyplot.plot(x, y, linestyle, label)`

In order to show the relationship between radius and area in a circle, the scatter plot can be used.

```
import matplotlib.pyplot as pyplot

radius = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
area = [3.14159, 12.56636, 28.27431, 50.26544, 78.53975, 113.09724]
pyplot.plot(radius, area, linestyle='--', label='Circle')
pyplot.xlabel('Radius')
pyplot.ylabel('Area')
pyplot.title('Area of Circle')
pyplot.legend()
pyplot.show()
```

The results is:
```python
import matplotlib.pyplot as pyplot

labels = ['Group1', 'Group2', 'Group4', 'Group5']
sizes = [15, 30, 45, 10]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
pyplot.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=90)
pyplot.axis('equal')
pyplot.show()
```

The results is:
Reference

For more information about matplotlib, please visit http://matplotlib.org/
The plotting commands mentioned above and other commands that you may find interesting can be found at http://matplotlib.org/api/pyplot_summary.html
More examples using matplotlib can be found at http://matplotlib.org/examples/index.html