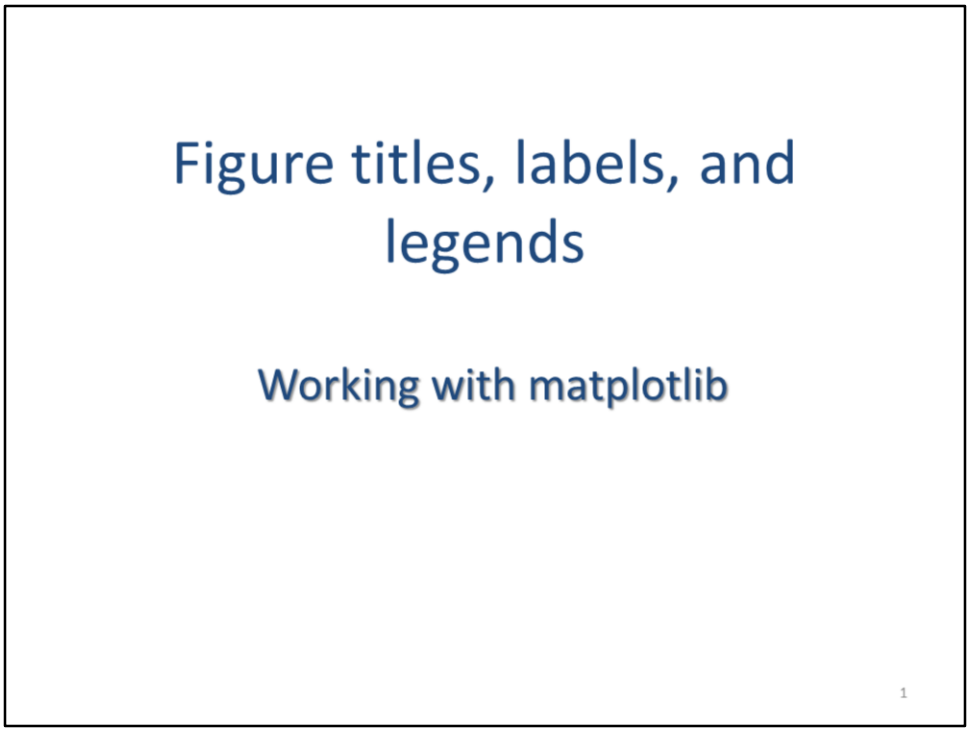
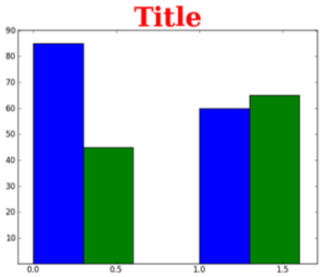


This video will discuss how to add titles, labels, and legends to figures created in **matplotlib**.

Figure title

- Add figure title…

```
title = figure.title('Title', ha = 'center',
        color = 'red', size = 40,
        weight = 'bold')
```



alignment options:
'left'; 'right'; 'center'

- Titles are text objects– additional font options can be set (see slide 7-9).

2

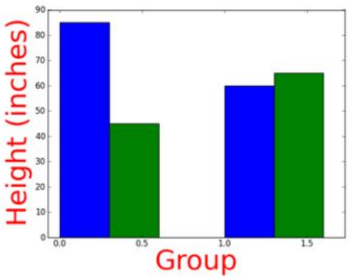
The next several slides will show how to add titles and customize axes labels for a figure. The examples will use bar plots for illustration; however, keep in mind that these methods are applicable to all plot types. The statement shown here will add a title to a figure. The first parameter should be a text string that will be used for the title. Optional parameters can be set by using the appropriate keywords. These parameters include setting the title alignment (i.e. left, right, center), the font color, the font size, and the font weight (e.g. bold).

Additional font options will be discussed later in this video.

Axis titles

- Set the x-axis title…

```
figure.xlabel('Group', size = 40,
        color = 'r')
```



- Set the y-axis title…

```
figure.ylabel('Height (inches)',
        size = 40, color = 'r')
```

- Axes titles are text objects – see slide 7-9 for additional font options.
- Refer to xlabel documentation for more options…
  http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.xlabel

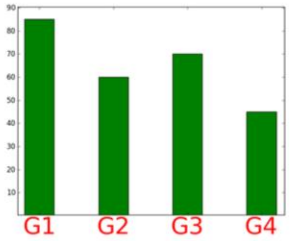
3

The axes titles can be set for a figure by using the **xlabel** and **ylabel** methods. The first parameter for both methods should be a string that will be used for the axis title. Additional parameters are optional and should be indicated by their keywords.

More information on customizing x- and y-axis titles can be found in the pyplot documentation.

Setting ticks labels - categorical

- Set categorical labels (e.g. for x-axis)…




```
labels = ['G1', 'G2', 'G3', 'G4']
left = [0, 1, 2, 3]
locations = [i + (width/2) for i in left]
figure.xticks(locations, labels, size = 40, color = 'r')
```

left coord. for bars

bar location + ½ bar width

use yticks for y-axis

- Ticks labels are text objects – see slides 7-9 for additional font options.

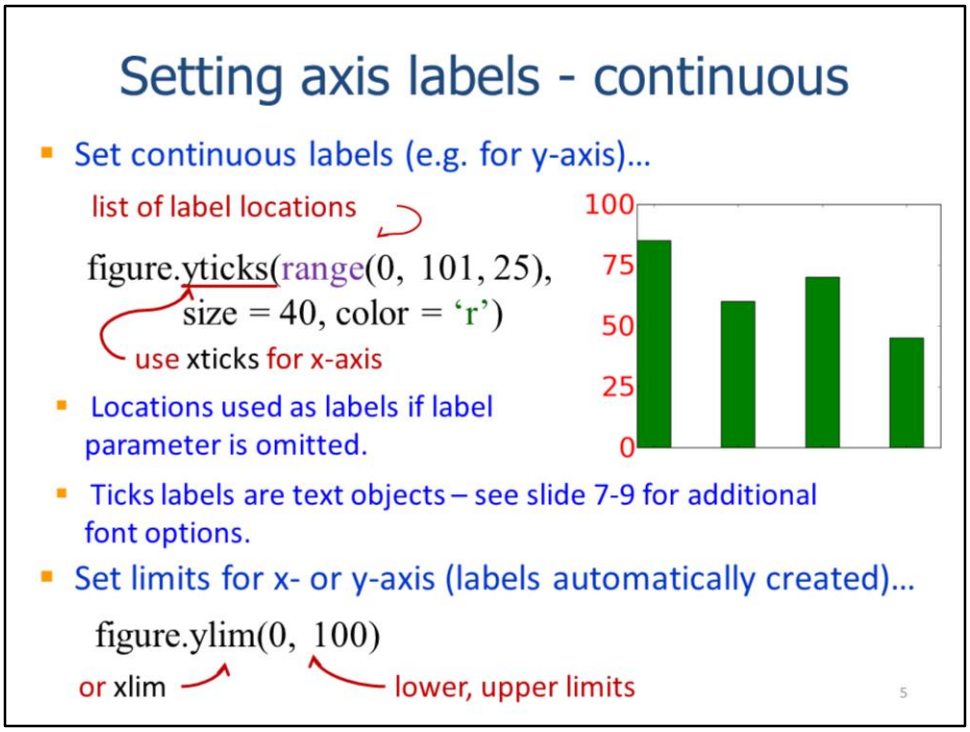
4

The tick labels can be set for both the x- and y-axis. The labels will need to be stored in a list in the order of their position along the axis.

This list stores the left positions for each bar – these positions will be used to calculate the locations of the labels.

The locations list shown here specifies the tick label positions which are centered under the bars. The locations of the centers of the bars was calculated by adding ½ the bar width to the left position of the bar. The center locations can be used as the tick label locations. The type of plot and personal preference determine where the labels should be positioned.

The **xticks** method will create the tick labels at the locations specified. Note that the tick locations should be the first parameter and tick labels should be second. Any optional parameters should follow the required parameters and be indicated by their keyword.



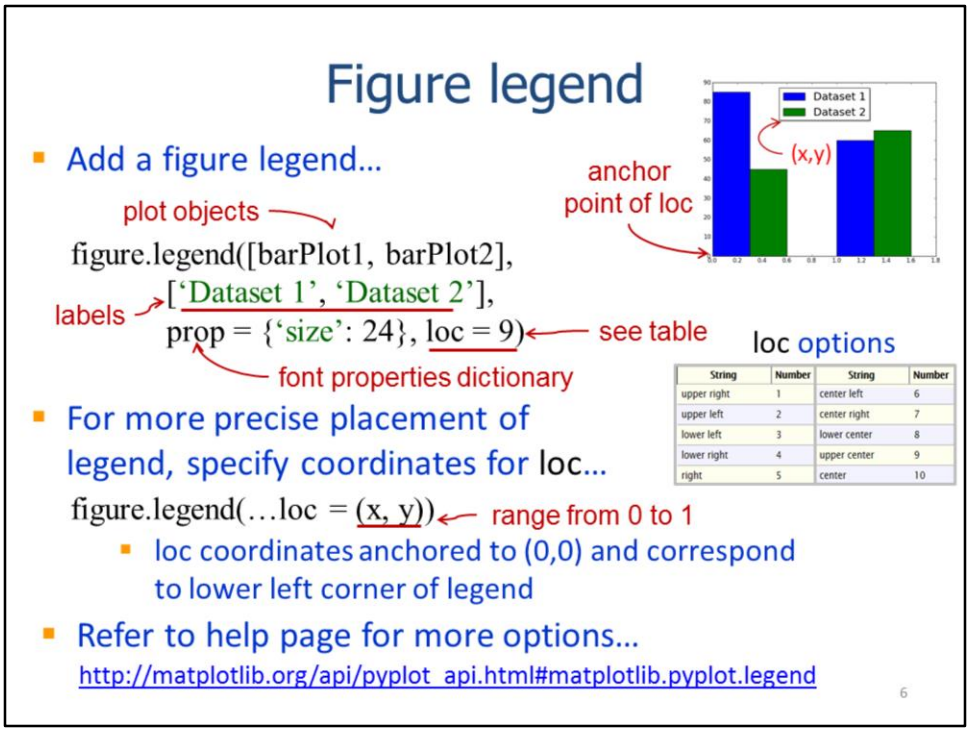
This slide will show how to set continuous tick labels for an axis.

The **xticks** or **yticks** methods are used to set the label positions. In this case, the range function was used to create a sequence of numbers that was used to specify the y-axis tick label locations.

The label positions will also be used as the tick labels if the labels parameter is omitted.

Note that tick labels are text objects and can be formatted like any other text in the figure.

Tick labels can be limited by using the **ylim** or **xlim** methods. Pyplot will generate the labels automatically within the limits that are specified.



The figure legend can be created using the **legend** method.

The plot objects are specified as the first parameter and the legend labels are specified as the second parameter. If multiple datasets have been plotted in the figure, then lists should be used to specify the plot objects and labels parameters.

A font properties dictionary is needed to customize the fonts in the legend labels. The font properties dictionary will be discussed later in this video.

The legend location can controlled through the **loc** parameter. The loc values range from 1-10 and correspond to the positions indicated in the chart.

The legend placement can be specified more precisely by specifying an x and y coordinate for the **loc** parameter. The x and y coordinates should range between 0 and 1 with (0, 0) being the lower left corner of the figure.

Refer to the pyplot documentation for further information on legends.

Text object

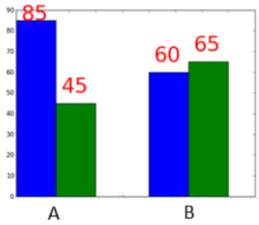
- Add a text object to the figure…

```
figure.text(x, y, value, size = 20)
```

string/number

* required parameter

axis coordinates of lower left corner of text box



- Formatting info can be specified with dictionary…

```
fontPropDct = {'color': 'b', 'size':12, 'ha': 'center'}
```

keyword

parameter setting

```
figure.text(x, y, value, fontdict = fontPropDct)
```

- Note: for legends, the font properties dictionary is required for formatting (keyword is prop for legends)…

7

Text boxes can be added to the figure by using the **text** method.

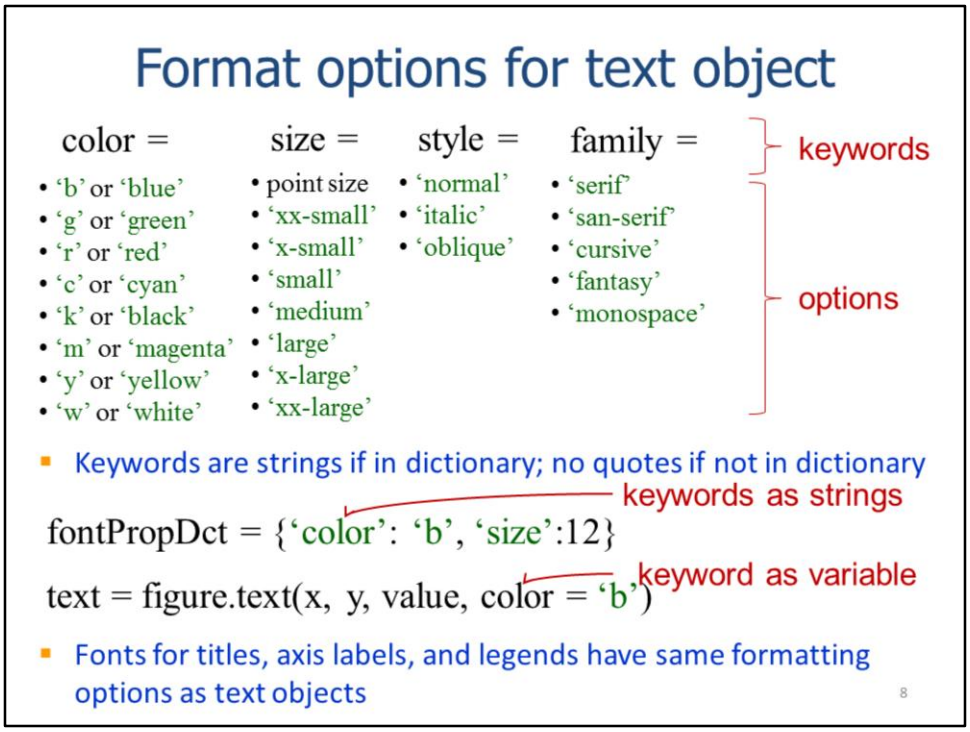
The x and y coordinates of the lower left corner of the text box need to be specified. The x- and y- coordinates can span the full range of values along their respective axes.

The value of a text box can either be a number or a string value.

Text can be formatted using a dictionary to make statements more concise. The font dictionary can also be used for figure and axes titles, legend labels, and tick labels. The keys in the dictionary must correspond to the keywords for the font parameters; the values correspond to the parameter settings.

After the font dictionary is defined, it can be specified for the **fontdict** parameter for methods that create figure text including **text, title, xlabel, ylabel, xticks,** and **yticks**.

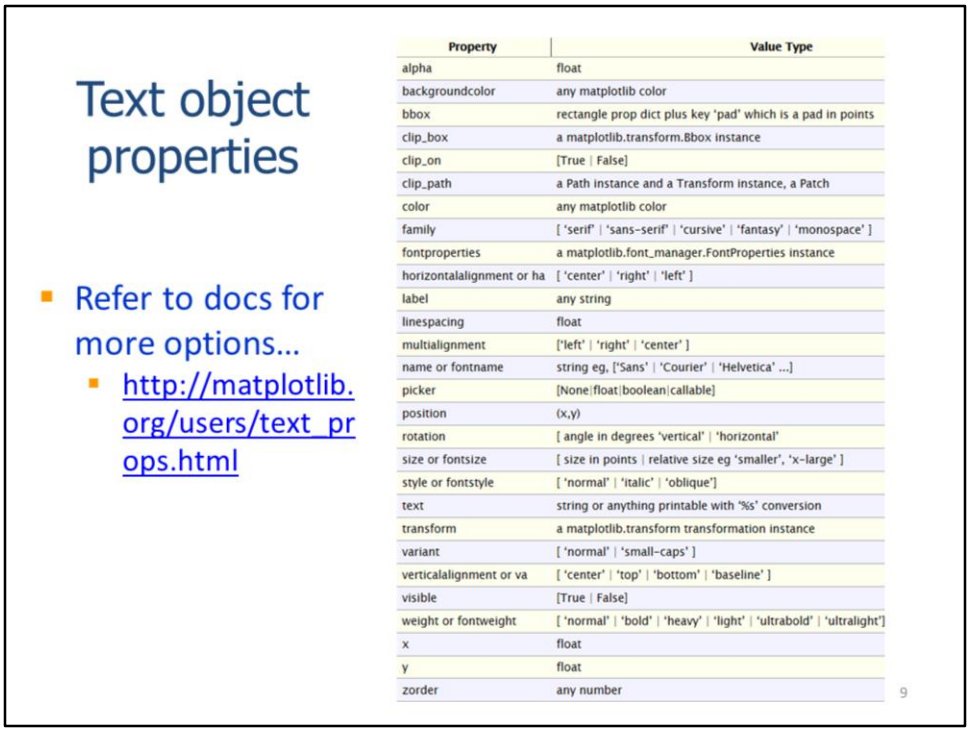
The font dictionary is required to customize fonts for the legend and, unlike the other methods, the **prop** keyword is used to set font properties.



This slide shows the keywords for some common parameters used to format text in a figure. Under each parameter is a list of acceptable values.

Note that the keywords are strings when used in the font dictionary; however, when the keywords are used directly in a method, the quotes are omitted.

All text elements in a figure have the same formatting options.

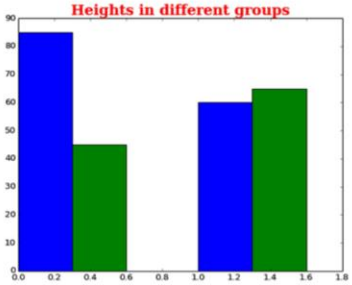


Property	Value Type
alpha	float
backgroundcolor	any matplotlib color
bbox	rectangle prop dict plus key 'pad' which is a pad in points
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a Patch
color	any matplotlib color
family	['serif' | 'sans-serif' | 'cursive' | 'fantasy' | 'monospace']
fontproperties	a matplotlib.font_manager.FontProperties instance
horizontalalignment or ha	['center' | 'right' | 'left']
label	any string
linespacing	float
multialignment	['left' | 'right' | 'center']
name or fontname	string eg, ['Sans' | 'Courier' | 'Helvetica' ...]
picker	[None|float|boolean|callable]
position	(x,y)
rotation	[angle in degrees 'vertical' | 'horizontal'
size or fontsize	[size in points | relative size eg 'smaller', 'x-large']
style or fontstyle	['normal' | 'italic' | 'oblique']
text	string or anything printable with '%s' conversion
transform	a matplotlib.transform transformation instance
variant	['normal' | 'small-caps']
verticalalignment or va	['center' | 'top' | 'bottom' | 'baseline']
visible	[True | False]
weight or fontweight	['normal' | 'bold' | 'heavy' | 'light' | 'ultrabold' | 'ultralight']
x	float
y	float
zorder	any number

Refer to the matplotlib documentation for more information on properties that can be set for text objects. There are many additional properties of text objects that we will not have time to discuss in this video.

Example script: side-by side bar plot

```
import matplotlib.pyplot as figure
heights1 = [85,60]
heights2 = [45,65]
barWidth= 0.3
left1= [0,1]
left2= [val+barWidth for val in left1]
bar1= figure.bar(left1, heights1, width=barWidth, color='b')
bar2= figure.bar(left2, heights2, width=barWidth, color='g')
fontProps ={'family': 'serif', 'color': 'r', 'size': 20, 'style': 'normal',
            'weight': 'bold', 'ha': 'center'}
figure.title('Heights in different groups', fontdict = fontProps, loc='center')
```



10

The next two slides will show an example script that creates a vertical side-by-side bar plot complete with a title, labels, and a legend.

The first statement imports pyplot and assigns it the name figure.

These lists contain two example datasets that will be used to set the bar heights.

This statement defines the width of the bars.

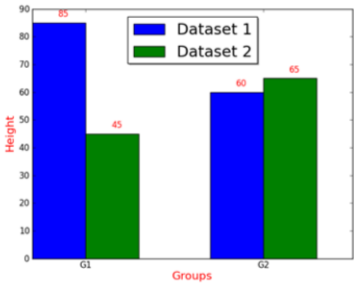
These lists define the left positions of the bars for the two datasets. The bar positions for the 2nd dataset (left2) are shifted to the right of the bars in the 1st dataset (left1) by the width of the bar.

The **bar** method is used to plot each dataset using the appropriate left and height lists. The bars for each dataset are set to different colors. Note that variables have been assigned to each bar plot objects – we will need to refer to these objects when creating the legend.

These statement add the title to the figure. The **loc** parameter was used to center the title above the plot in the figure. A font properties dictionary is used to set the font properties for the title. Using the dictionary helps to keep the **title** statement more concise and it could also be used to set the same font properties for other text elements in the figure.

Example script: labeling bar heights

```
figure.legend([bar1,bar2], ['Heights1', 'Heights2'], loc=9)
figure.xticks(left2, ['G1', 'G2'])
figure.xlabel('Groups', size = 16)
figure.ylabel('Height', size = 16)
fontProps={'color': 'r', 'size':12, 'ha': 'center'}
heights=heights1+heights2
lefts = left1 + left2
for i, height in enumerate(heights):
    x = lefts[i] + barWidth / 2.0
    y = height + 2
    figure.text(x, y, height, fontdict = fontProps)
figure.show()
```



11

This statement creates the legend for the figure. The bar plot objects are specified in a list for the first parameter. The list for the second parameter contains the legend labels that will be used for each dataset. The location of the legend is set to the "upper center" of the figure.

The xtick labels are created at the left bar positions of the 2nd dataset so that the labels will be centered under the pairs of bars.

These statements set the x- and y-axis titles.

We've redefined the font properties dictionary to use for text elements that will be added above the bars.

The bar heights and left positions for both datasets are combined – this will help in defining the locations of text that will be placed above each bar.

We'll use a for loop to add text above each bar that indicates the bar height. Using **enumerate** to get the loop iteration number will help get the left bar position that corresponds to each height.

The x position of the text will be the left position if each bar plus ½ the bar width – this will center the label above the bar.

The y position of the text box will be 2 y-axis units above the top of the bar.

The **text** method adds the text box to the figure. Font properties are set using the font dictionary

The last statement will display the figure.