CSIT121
Object-Oriented Design and Programming

Dr. Fenghui Ren

School of Computing and Information Technology
University of Wollongong

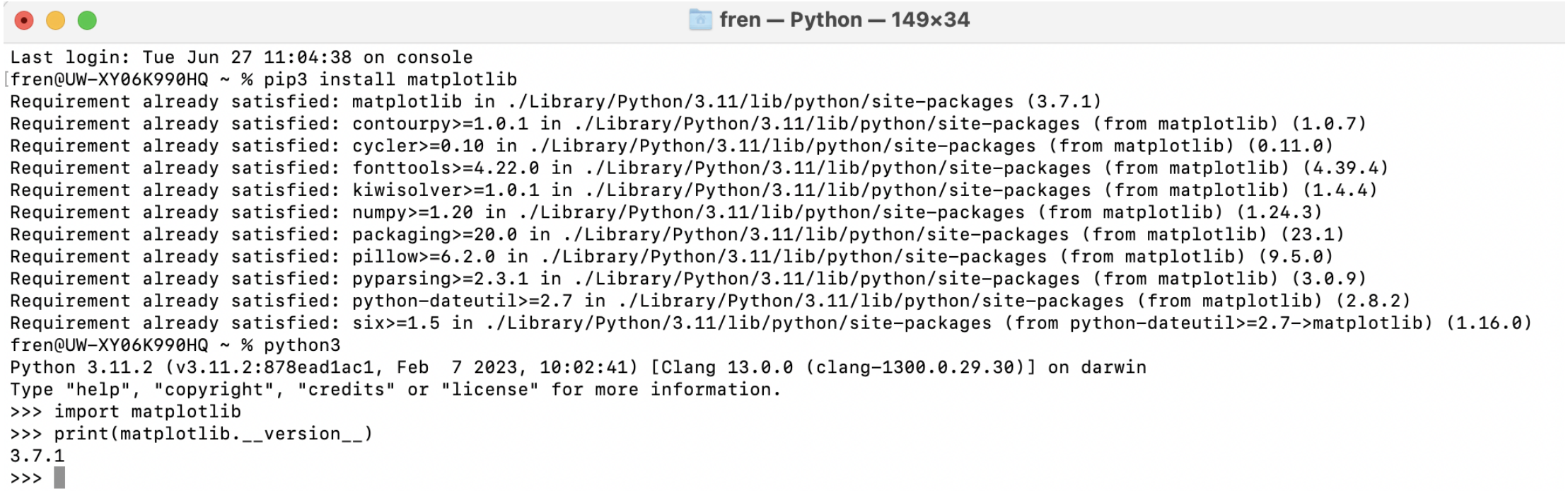
Lecture 10 outline

- Matplotlib library
- Installation of Matplotlib
- Plotting and Markers
- Line and labels
- Grid and Subplot
- Scatter, bars
- histograms and pie charts
- The lifecycle of a plot

Matplotlib library

- Matplotlib is a low-level graph plotting library in Python that serves as a visualisation utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.
- Matplotlib is mostly written in Python, a few segments are written in C, Objective-C and Javascript for platform compatibility.
- The source code for Matplotlib is located at this GitHub repository: https://github.com/matplotlib/matplotlib

Installation of Matplotlib

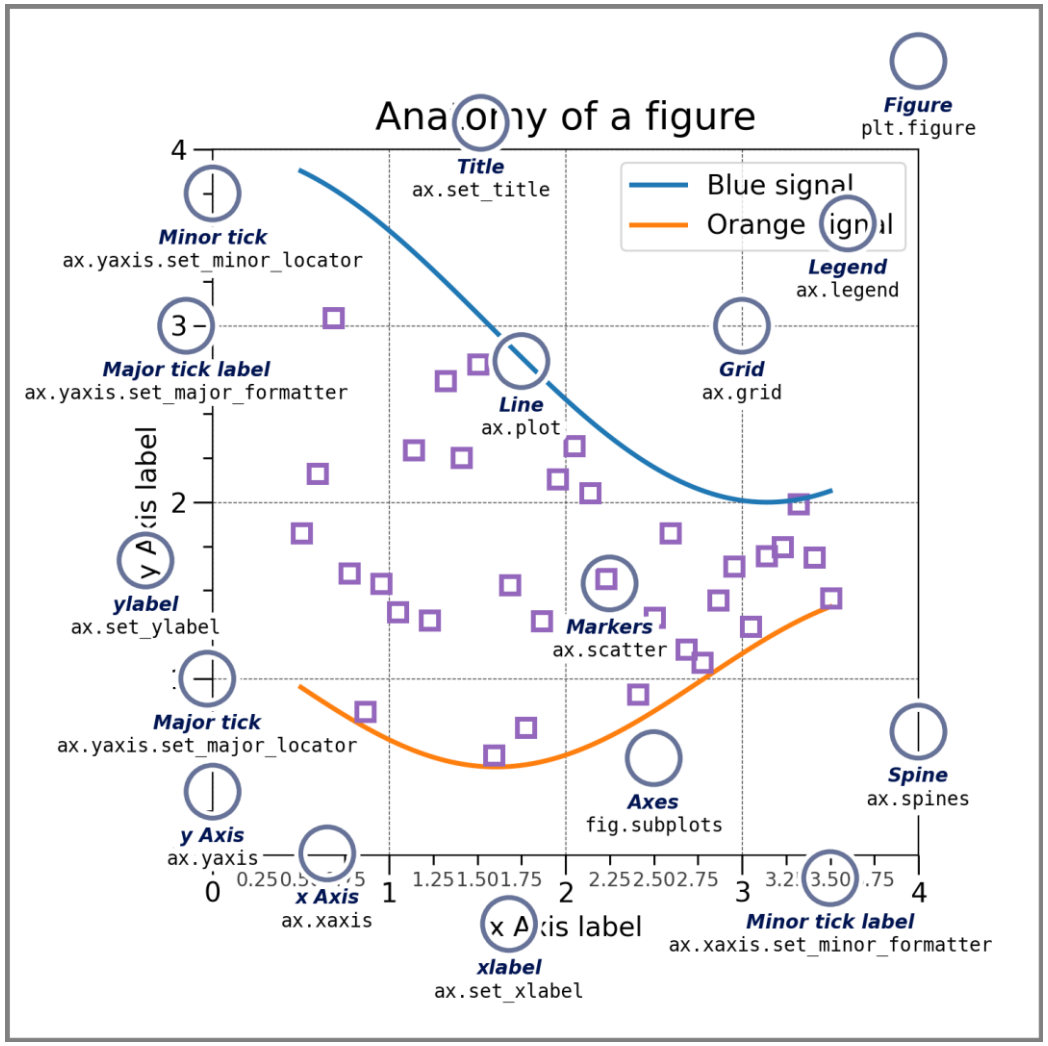


- Matplotlib is not pre-installed in Python, so you have to install it by yourself.
- Our computer lab has installed Matplotlib already.
- If you have PIP already installed on your Python, you can use command ‘pip3 install matplotlib’ to install.
- You can use ‘matplotlib.__version__’ to check the version of your matplotlib

Matplotlib figure

Here are some components of a Matplotlib figure. We will introduce some basic components.

- Plotting
- Markers
- Lines
- Labels
- Grids
- Subplot
- Scatter
- Bars
- Histograms
- Pie charts

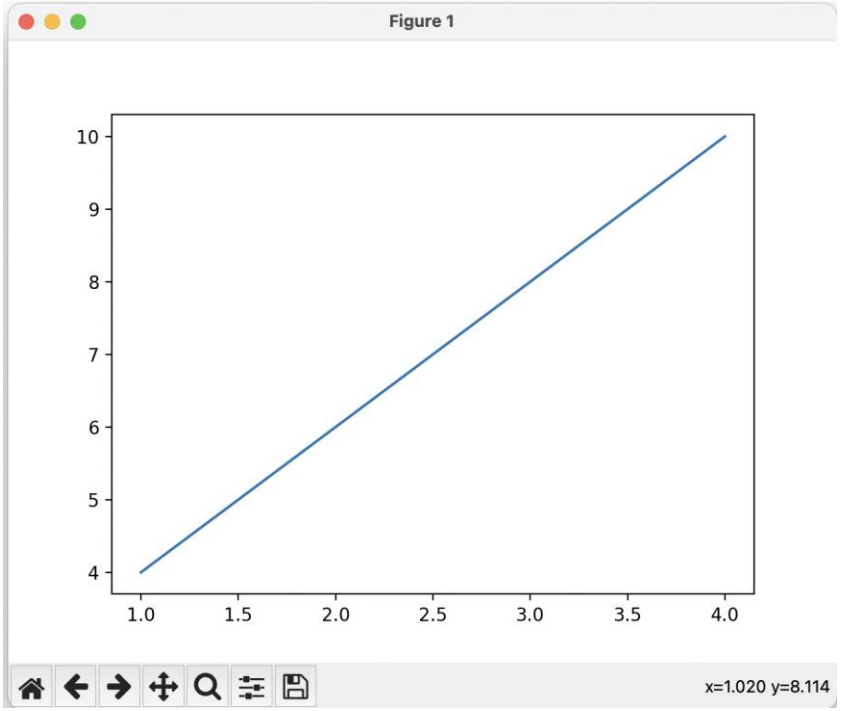


https://matplotlib.org/

Pyplot

- Most of the Matplotlib utilities lies under the ‘pyplot’ submodule, you can import it with the command ‘import matplotlib.pyplot as plt’
- Now you can try your first Matplotlib program to draw a segment between two points.

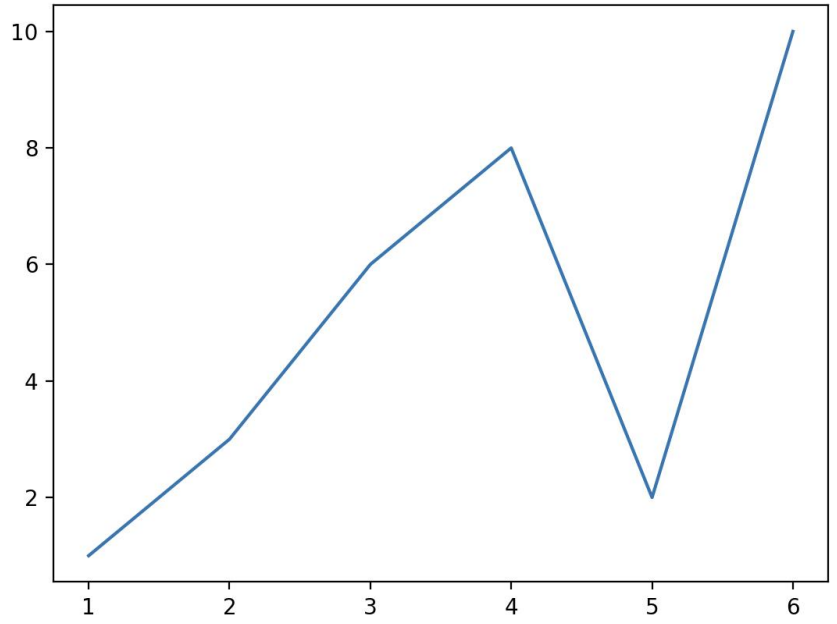
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 xpoints = np.array([1,4])
5 ypoints = np.array([4,10])
6
7 plt.plot(xpoints,ypoints)
8 plt.show()
```



Plotting

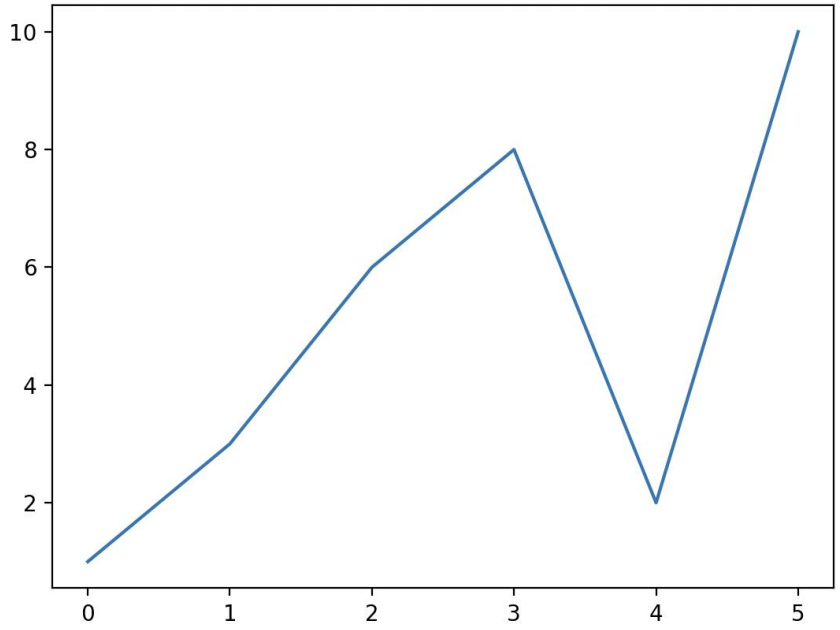
- You can also draw segments with multiple points.

```
xpoints = np.array([1,2,3,4,5,6])
ypoints = np.array([1,3,6,8,2,10])

plt.plot(xpoints,ypoints)
[<matplotlib.lines.Line2D object at 0x10f591290>]
plt.show()
```



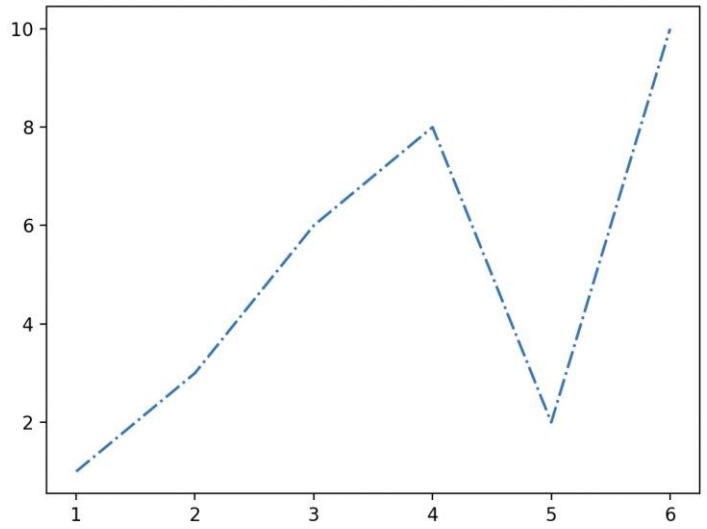
- If you do not specify the x points, they will get the default values 0, 1, 2, … etc, depending on the length of the y-points.

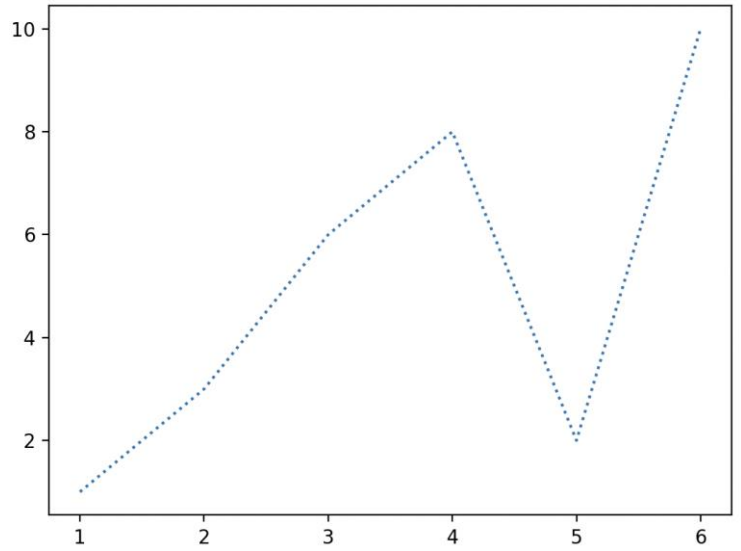
```
plt.plot(ypoints)
[<matplotlib.lines.Line2D object at 0x10f5d1ad0>]
plt.show()
```

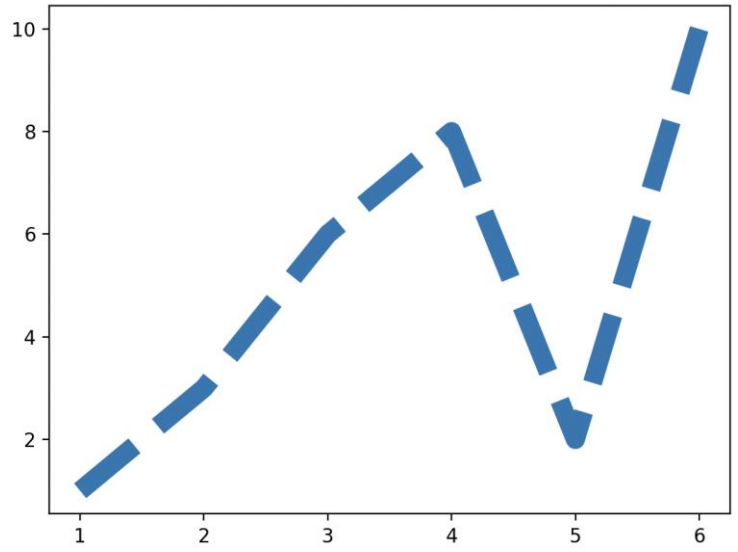


Plotting

- You can change the styles and the size of the line.
  - ‘-’ for the solid line (default)
  - ‘:’ for the dotted line
  - ‘--’ for the dashed line
  - ‘-.’ for the dashed/dotted line







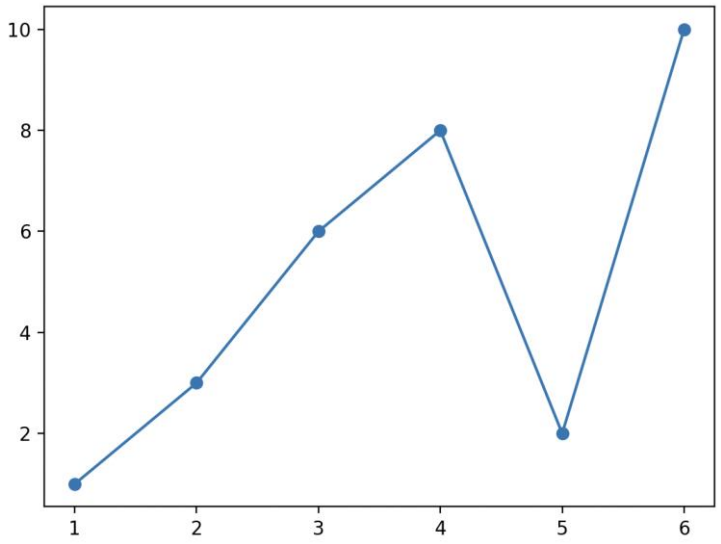

```
plt.plot(xpoints,ypoints, '-.')
```

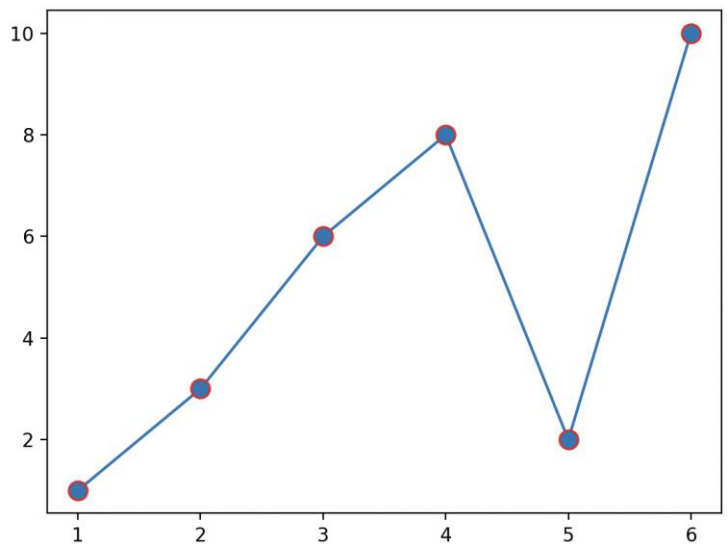
```
plt.plot(xpoints,ypoints, ':')
```

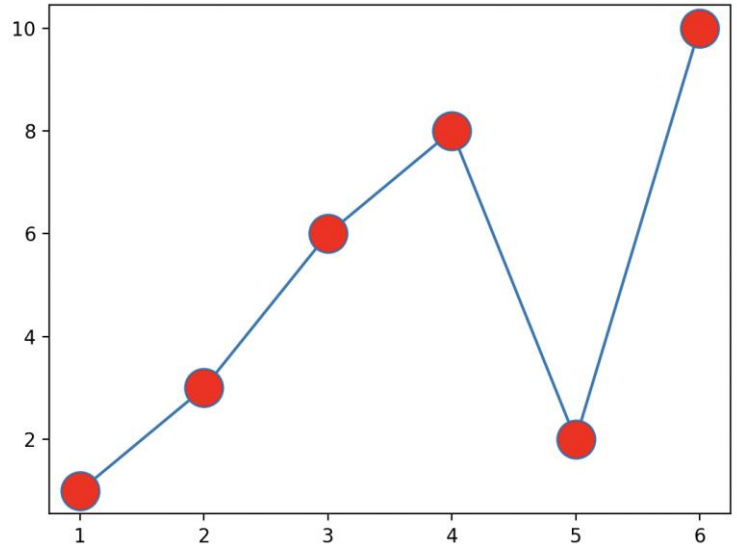
```
plt.plot(xpoints,ypoints, '--', linewidth = '10')
```

Markers

- We can add a specified marker (such as circles, stars and etc) to points.
- We can also customise the marker's size and edge/inside/whole colour.








```
plt.plot(xpoints,ypoints,marker = 'o')
```

```
plt.plot(xpoints,ypoints,marker = 'o', ms = 10, mec = 'r')
```

```
plt.plot(xpoints,ypoints,marker = 'o', ms = 20, mfc = 'r')
```

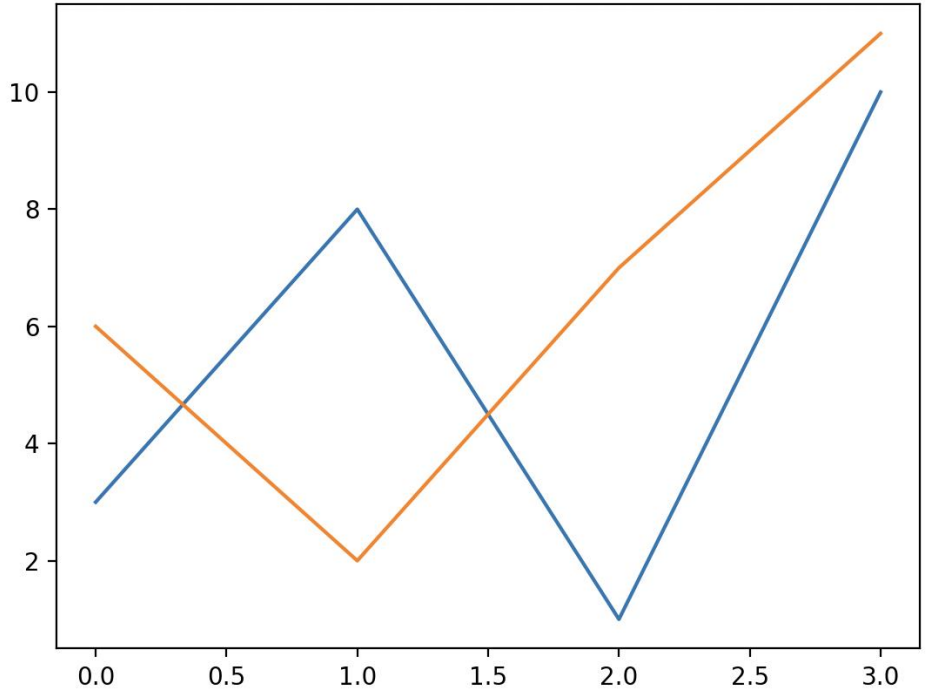
Markers

Marker	Description
'o'	Circle
'*'	Star
'.'	Point
','	Pixel
'x'	X
'X'	X (filled)
'+'	Plus
'P'	Plus (filled)
's'	Square
'D'	Diamond
'd'	Diamond (thin)
'p'	Pentagon
'H'	Hexagon
'h'	Hexagon
'v'	Triangle Down
'^'	Triangle Up
'<'	Triangle Left
'>'	Triangle Right
'1'	Tri Down
'2'	Tri Up
'3'	Tri Left
'4'	Tri Right
'|'	Vline
'_'	Hline

Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

Plotting multiple lines

- We can also plot multiple lines to the same figure.




```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])
plt.plot(x1, y1, x2, y2)
```

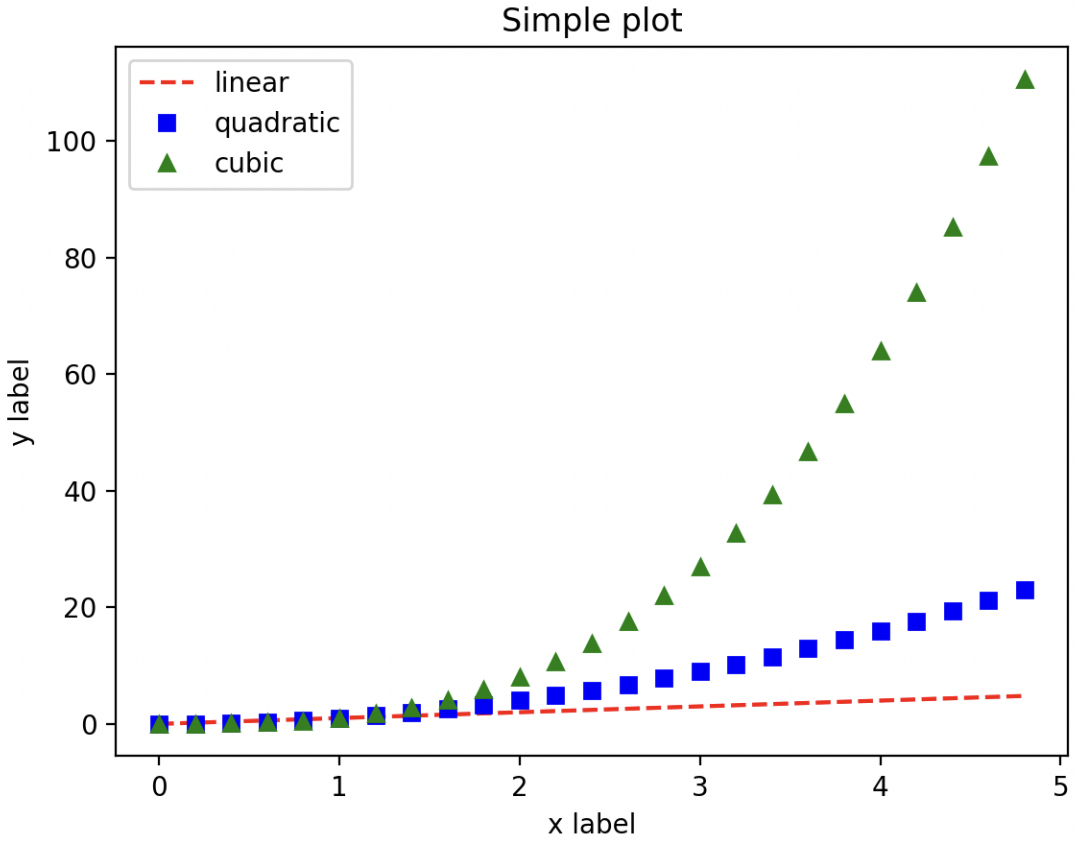
```
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.show()
```

(must be executed as a script)

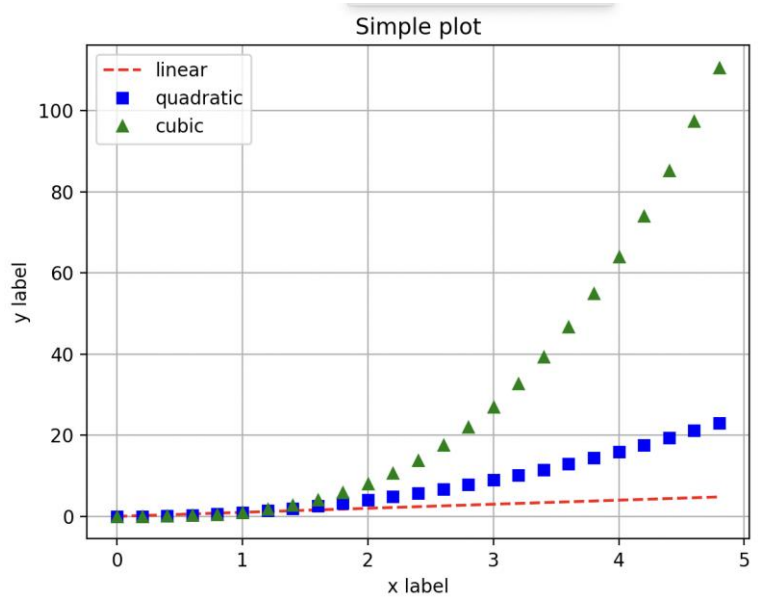
Labels, title and legend

- If we have multiple lines in a figure, usually it will be better to have some labels, a title and a legend for the figure.

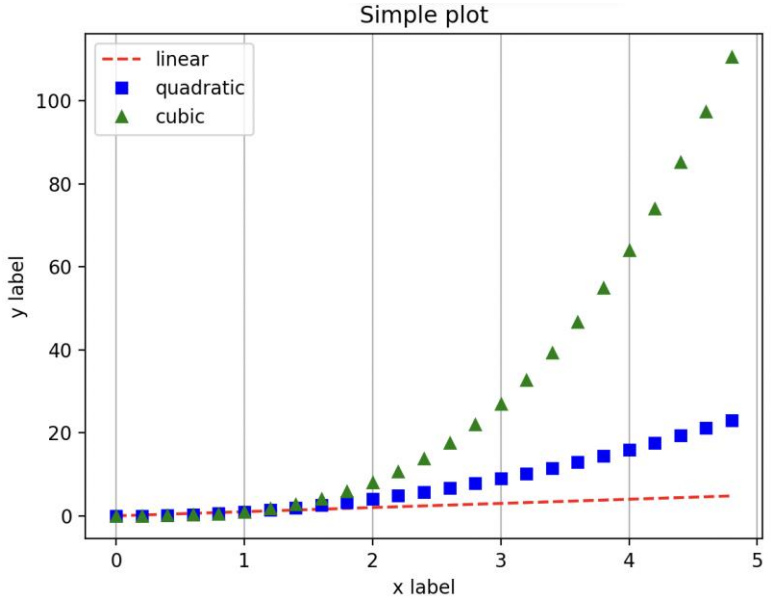
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # evenly sampled time at 200ms intervals
5 t = np.arange(0., 5., 0.2)
6
7 # red dashes, blue squares and green triangles
8 plt.plot(t, t, 'r--', label = 'linear')
9 plt.plot(t, t**2, 'bs', label = 'quadratic')
10 plt.plot(t, t**3, 'g^', label = 'cubic')
11 plt.xlabel('x label')
12 plt.ylabel('y label')
13 plt.title('Simple plot')
14 plt.legend()
15 plt.show()
```



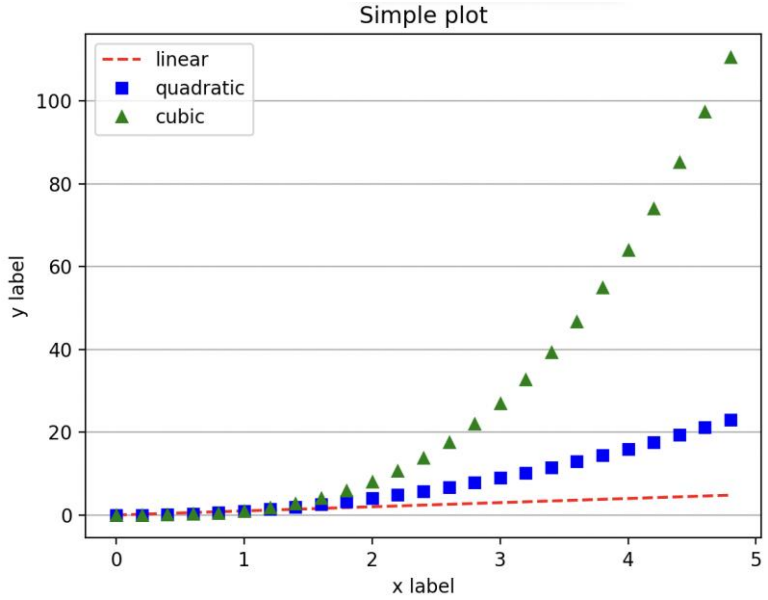
Grids



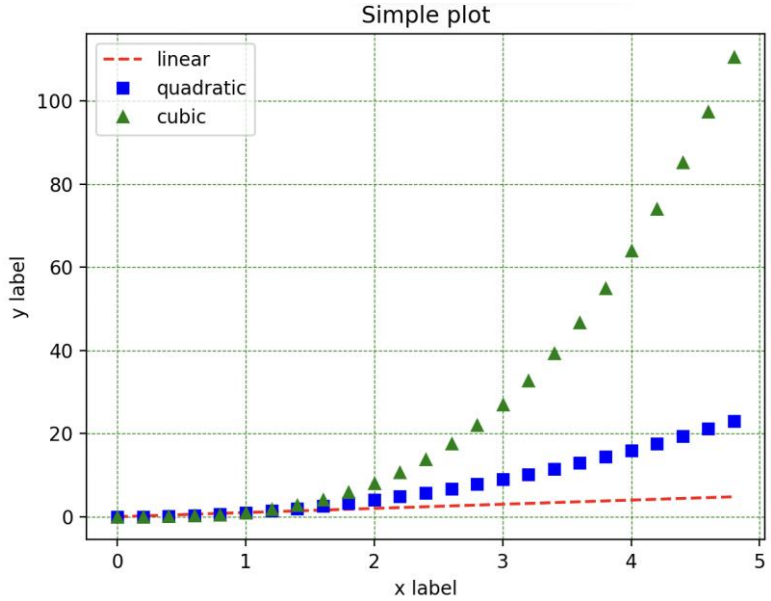
plt.grid()



plt.grid(axis = 'x')



plt.grid(axis = 'y')

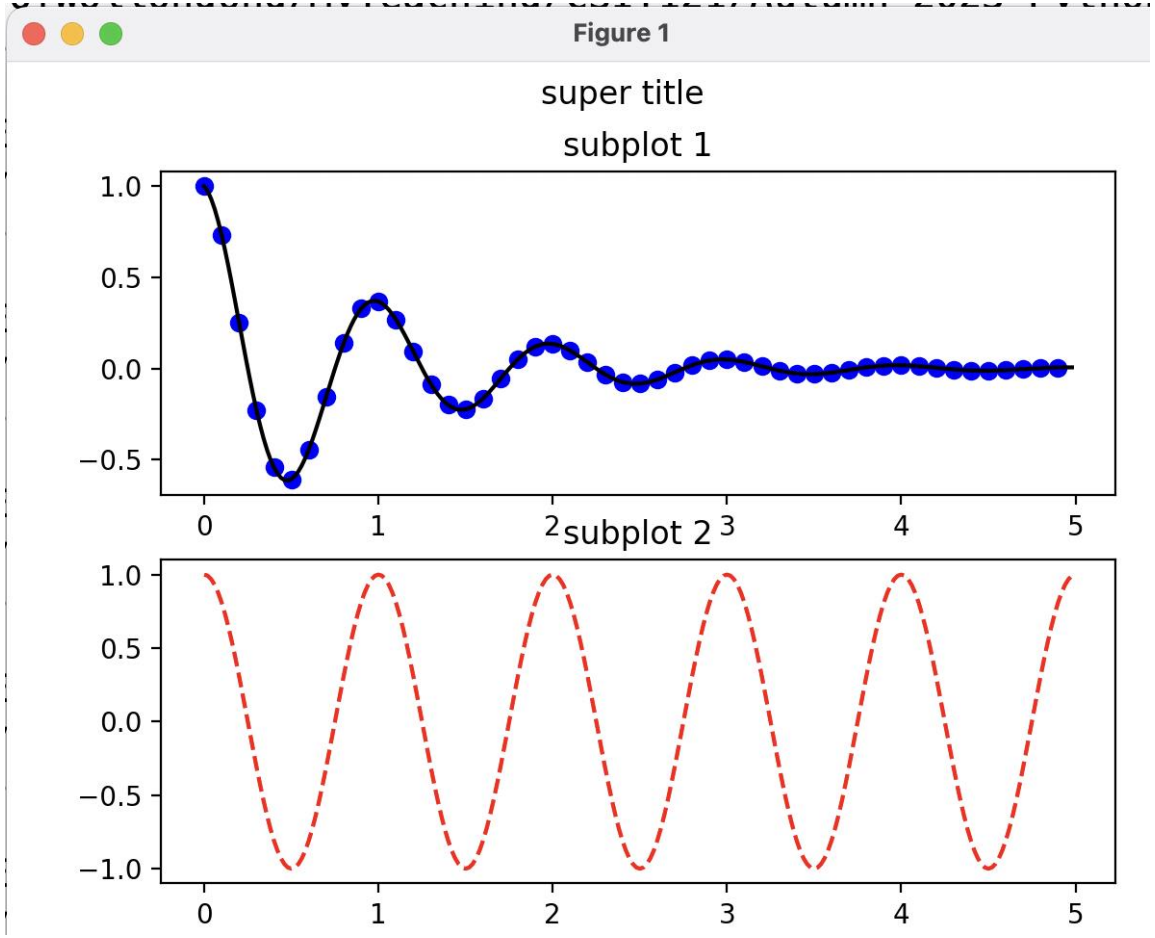



```
plt.grid(color='green',linestyle='--', linewidth=0.5)
```

Multiple plots

- With the subplot() function, we can draw multiple plots in one figure.

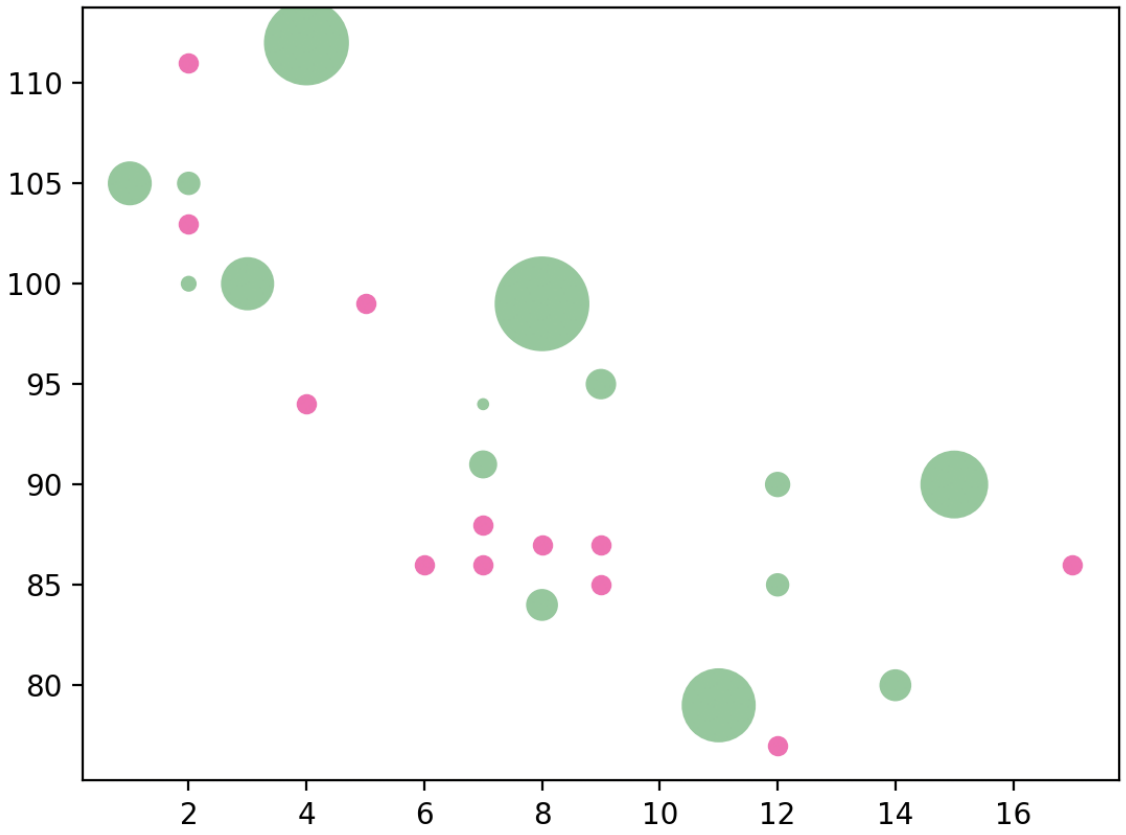
```
import matplotlib.pyplot as plt
import numpy as np

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

#plt.figure()
plt.subplot(2,1,1)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.title("subplot 1")

plt.subplot(2,1,2)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.title("subplot 2")

plt.suptitle("super title")

plt.show()
```



Scatter plot

- With Pyplot, you can use the scatter() function to draw scatter plots. It needs two arrays of the same length, one for the values on the x-axis, and one for the values on the y-axis.
- One figure can contain multiple scatter plots with different styles, such as colors, sizes, shapes and etc.
- You can even set a specific color for each dot by using an array of colors as values for the c (color) argument.
- The transparency of the dots can also be modified with the alpha argument.

Scatter plot



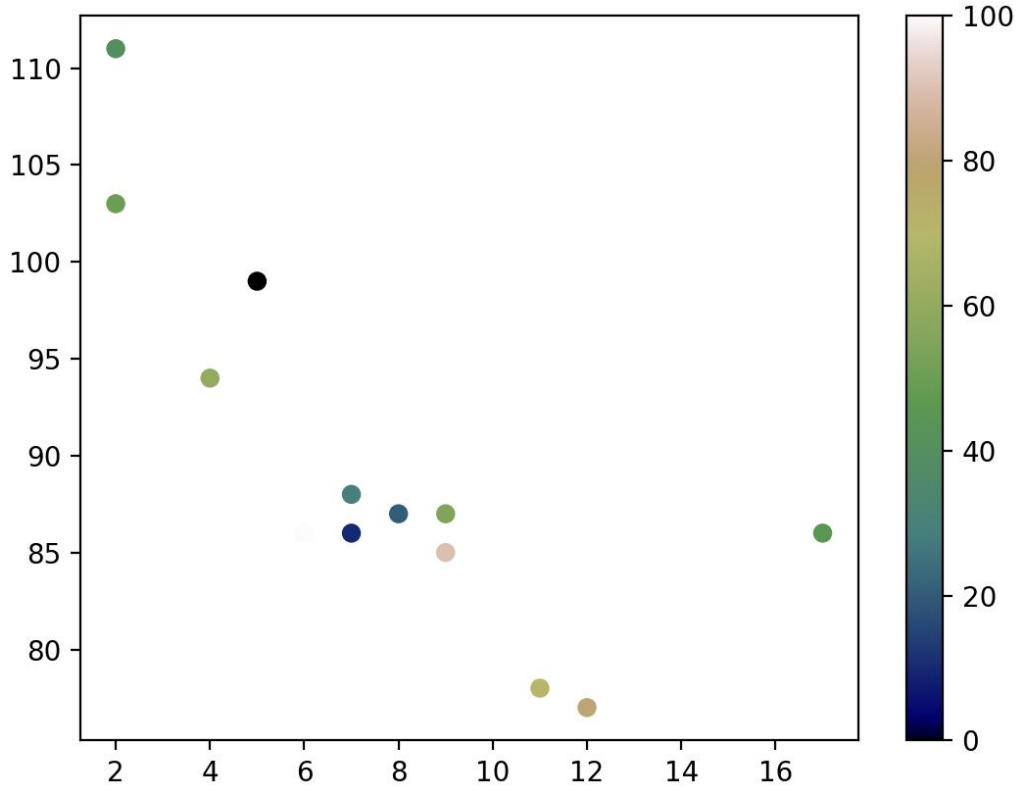

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y, color = 'hotpink')

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75,100, 50])

plt.scatter(x, y, color = '#88c999', s=sizes)

plt.show()
```

ColorMap

- A colormap is a list of colors, where each color has a value that ranges from 0 to 100.
- You can specify the colormap with the keyword argument ‘cmap’ with the value of the colormap, such as ‘viridis’ or create your own colormap.
- You can check Matplotlib APIs for all built-in colormaps.

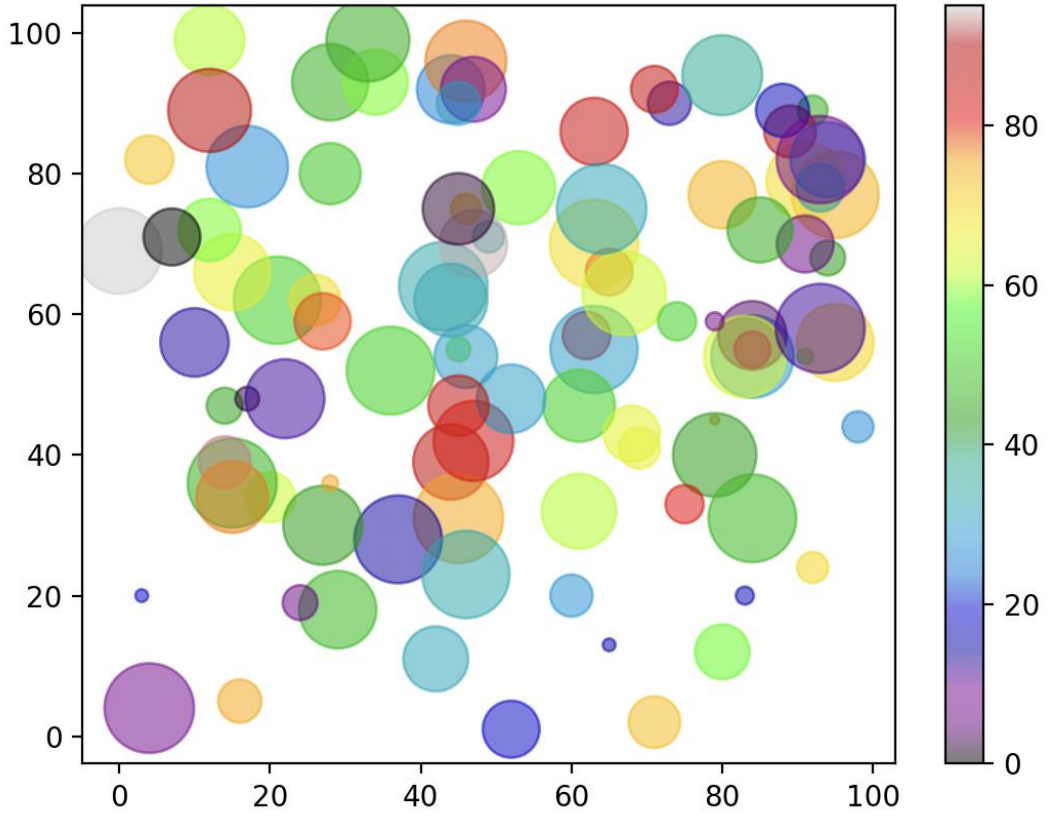
ColorMap

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75,100, 50])

plt.scatter(x, y, c=colors, cmap='gist_earth')
plt.colorbar()
plt.show()
```



Combine color, size and transparency

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```



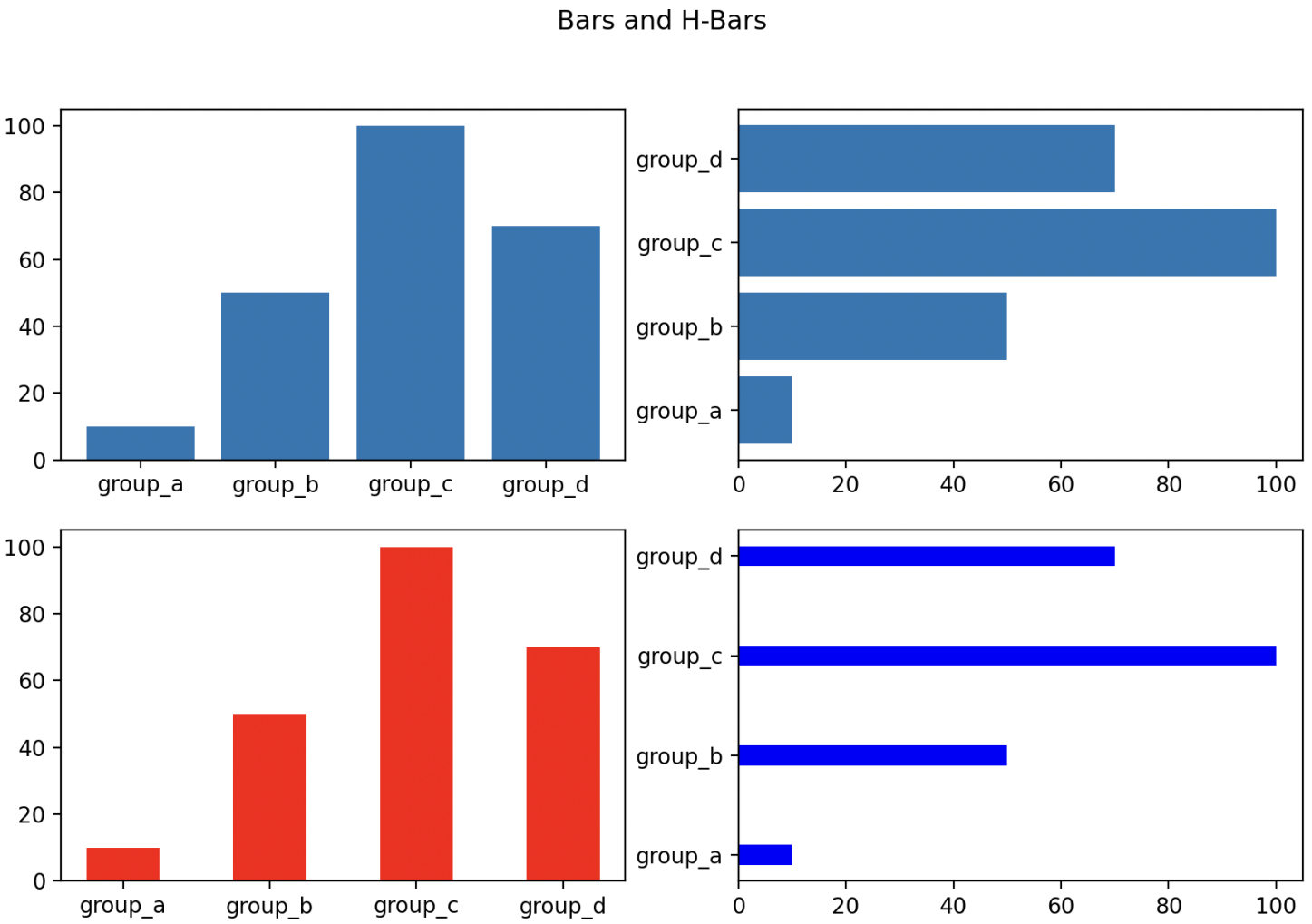
Bars

- With Pyplot, we can use the bar() function to draw bar graphs.
- barh() function can draw horizontal bars.
- The bar’s color and width can be customised.

Bars

```
import matplotlib.pyplot as plt

names = ['group_a', 'group_b', 'group_c', 'group_d']
values = [10, 50, 100, 70]

plt.subplot(2,2,1)
plt.bar(names, values)
plt.subplot(2,2,2)
plt.barh(names, values)
plt.subplot(2,2,3)
plt.bar(names, values, color='red', width=0.5)
plt.subplot(2,2,4)
plt.barh(names, values, color='blue', height=0.2)
plt.suptitle('Bars and H-Bars')
plt.show()
```

Bars

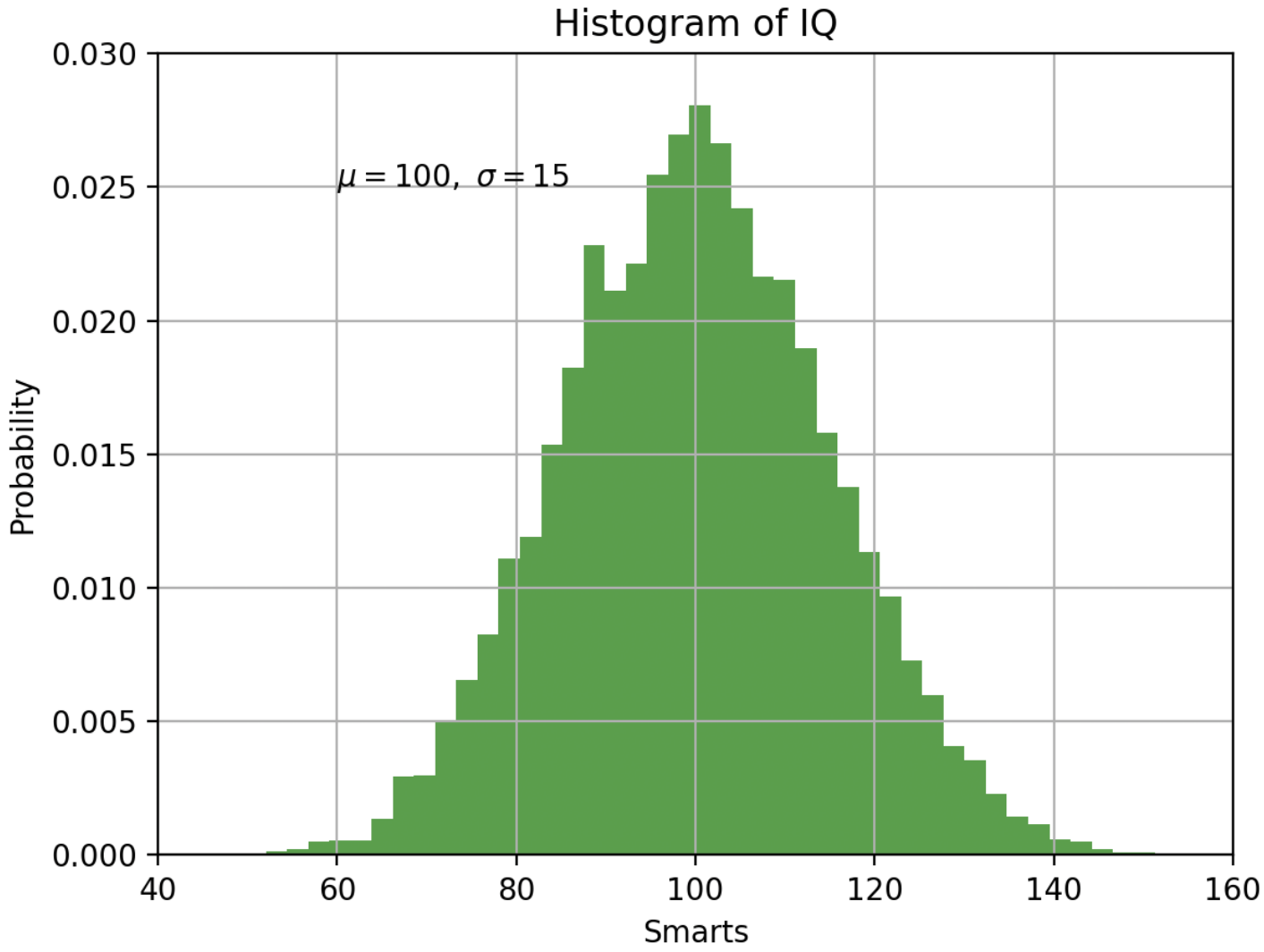

Bars and H-Bars
100
80
60
40
20
0
group_a
group_b
group_c
group_d
group_d
group_c
group_b
group_a
0
20
40
60
80
100

Histogram

- A histogram is a graph showing frequency distributions.

```
import matplotlib.pyplot as plt
import numpy as np

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Histogram

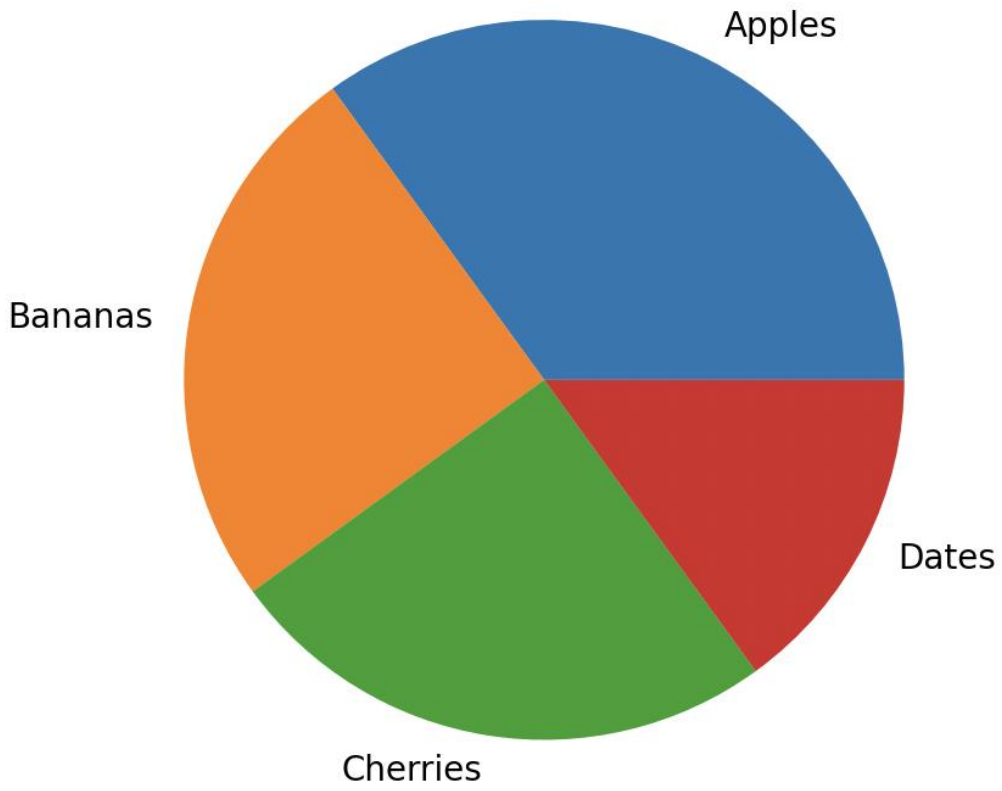

Histogram of IQ
$\mu = 100,\ \sigma = 15$
Probability
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
160
Smarts

Pie charts

- We can use the pie() function to draw pie charts.
- By default, the plotting of the first piece (wedge) starts from the x-axis and moves counterclockwise.
- The size of each wedge is determined by comparing the value with all the other values, i.e., ‘value/sum(values)’

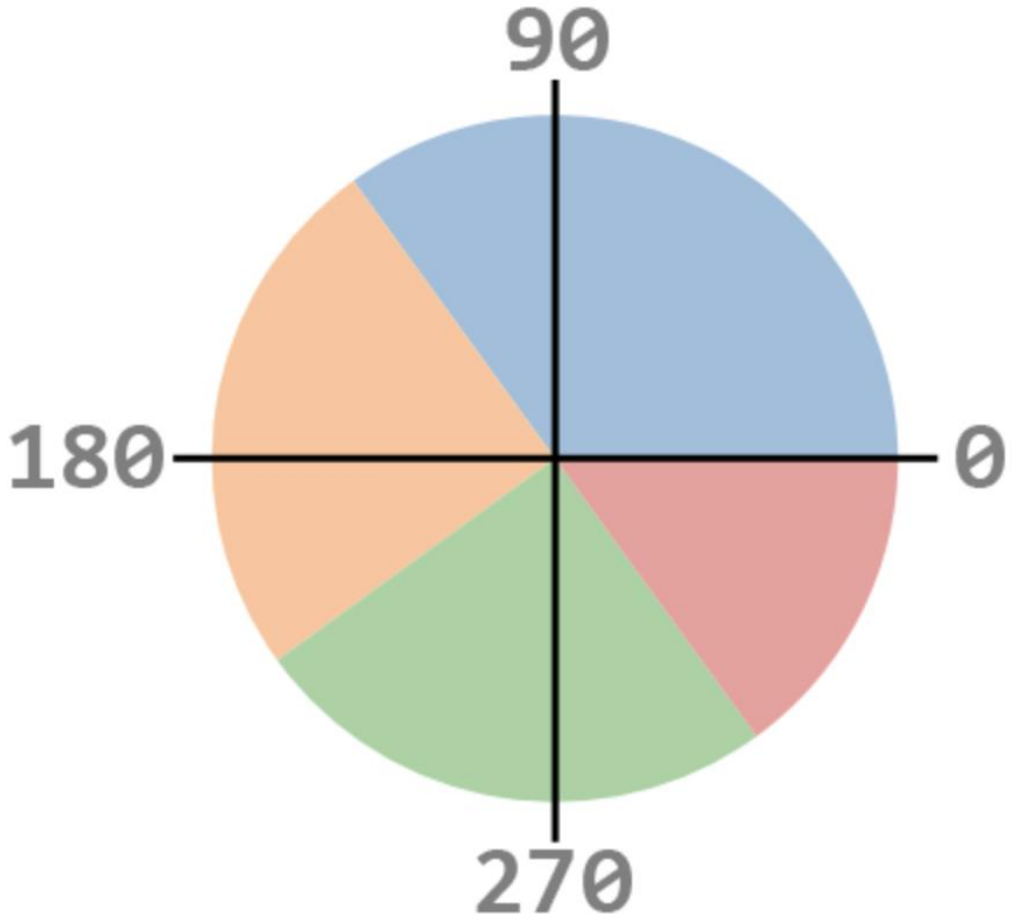
Pie charts

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```

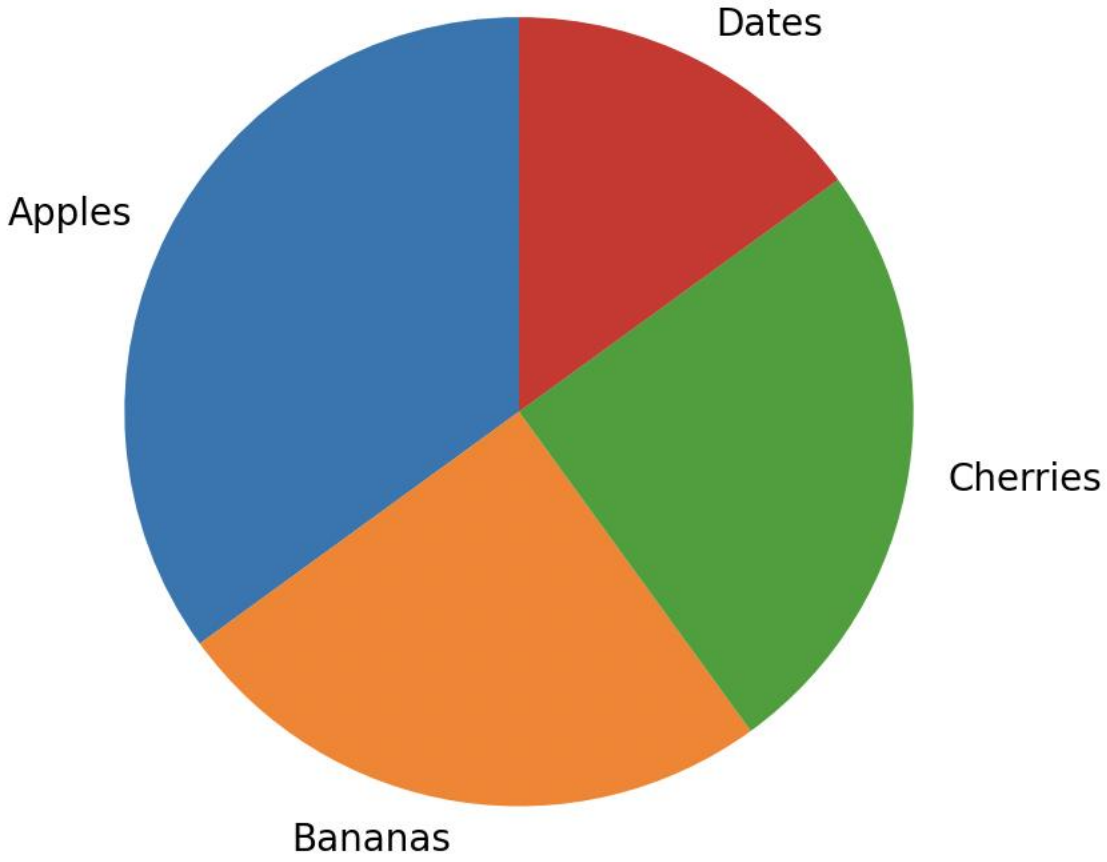


Start angle

- The default start angle of a pie chart is at the x-axis, but you can change the start angle by specifying a ‘startangle’ parameter.
- The ‘startangle’ parameter is defined with an angle in degrees.



Start angle

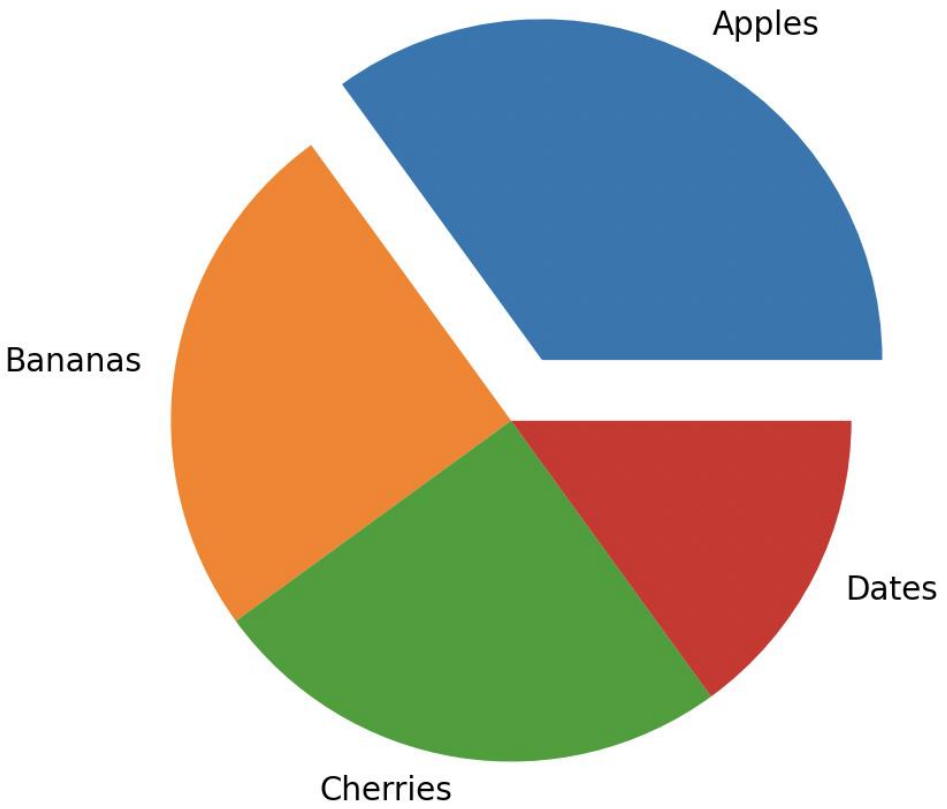



```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

Explode

- The explode parameter allows you to stand out one wedge.
- The explode parameter, if specified, and not None, must be an array with one value for each wedge.
- Each value represents how far from the center each wedge is displayed:

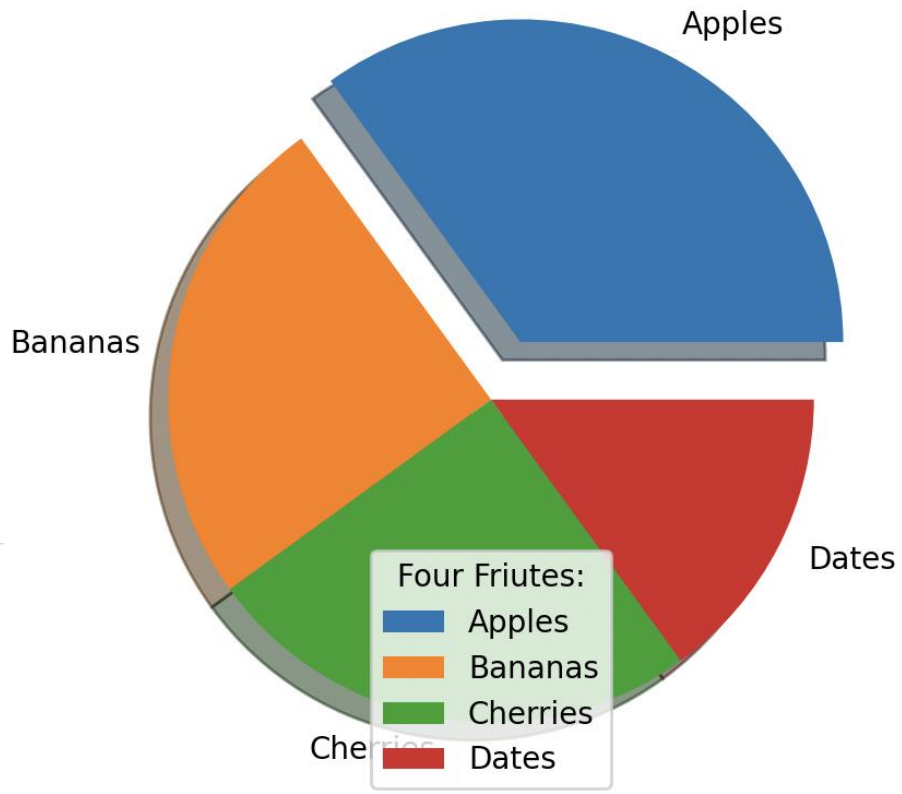
```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```



Title and legend

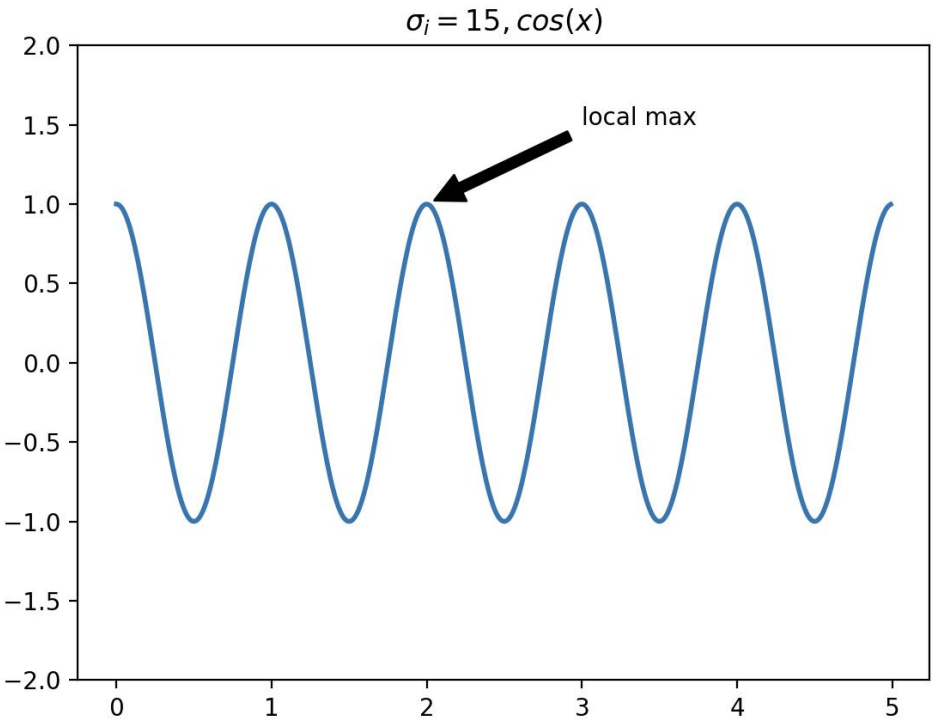
- You can also add the title, legend and the shadow.

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels = mylabels, explode = myexplode, shadow=True)
plt.legend(title="Four Friutes:")
plt.show()
```



Annotating text and math expressions

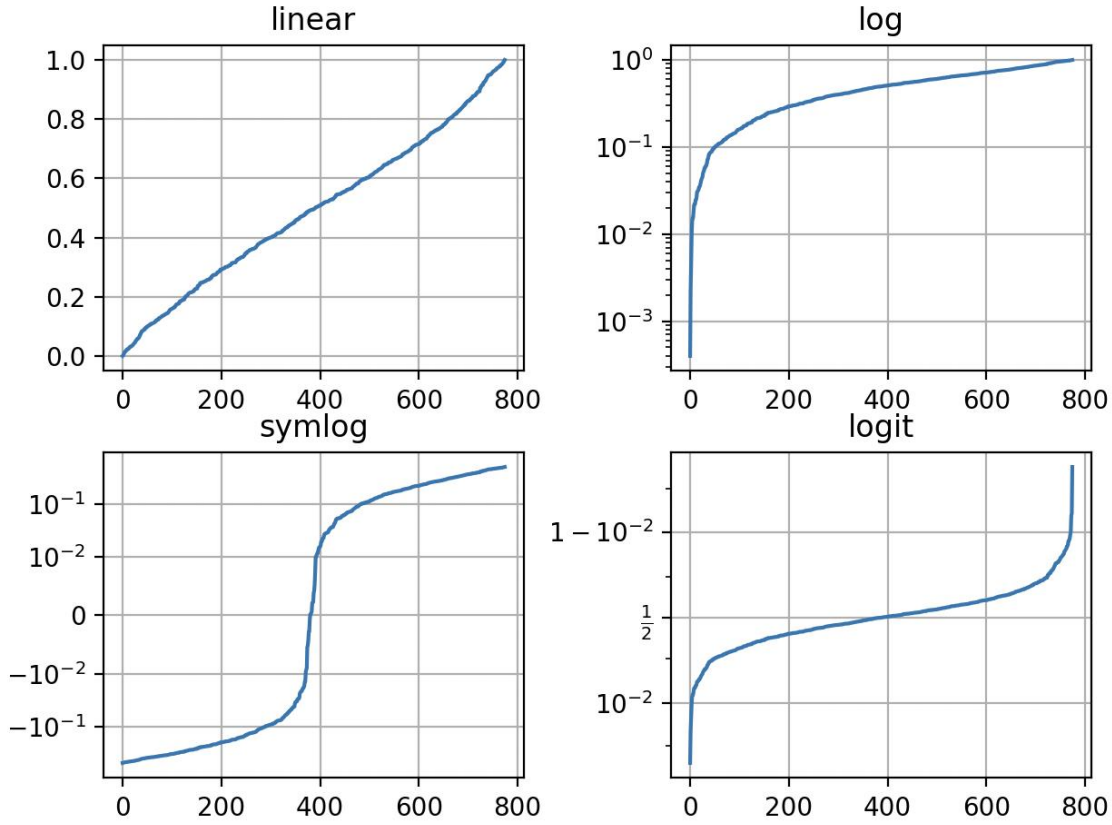
```
import matplotlib.pyplot as plt
import numpy as np

ax = plt.subplot()

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.ylim(-2, 2)
plt.title(r'$\sigma_i=15, cos(x)$')
plt.show()
```



Logarithmic and nonlinear axes

- Matplotlib supports not only linear axis scales but also logarithmic and logit scales. To change the x/y axis scale, just call the methods ‘xscale(new_scale)’ or ‘yscale(new_scale)’.

```
# log
plt.subplot(222)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)

# symmetric log
plt.subplot(223)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthresh=0.01)
plt.title('symlog')
plt.grid(True)
```



The lifecycle of a plot

- Data collection/creation
- Creation of figure instance
- Selection the style
- Customising the plot
- Combining multiple visualisation
- Saving the plot

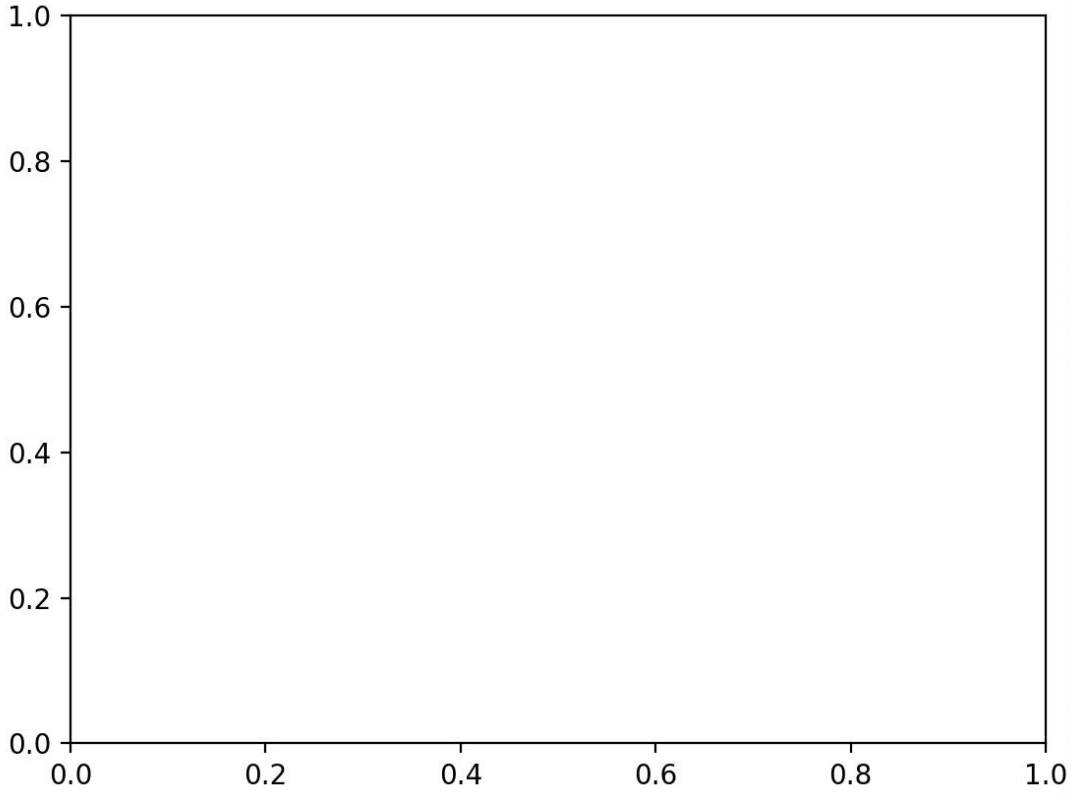
Data creation

The sample data: sales information for a number of companies.

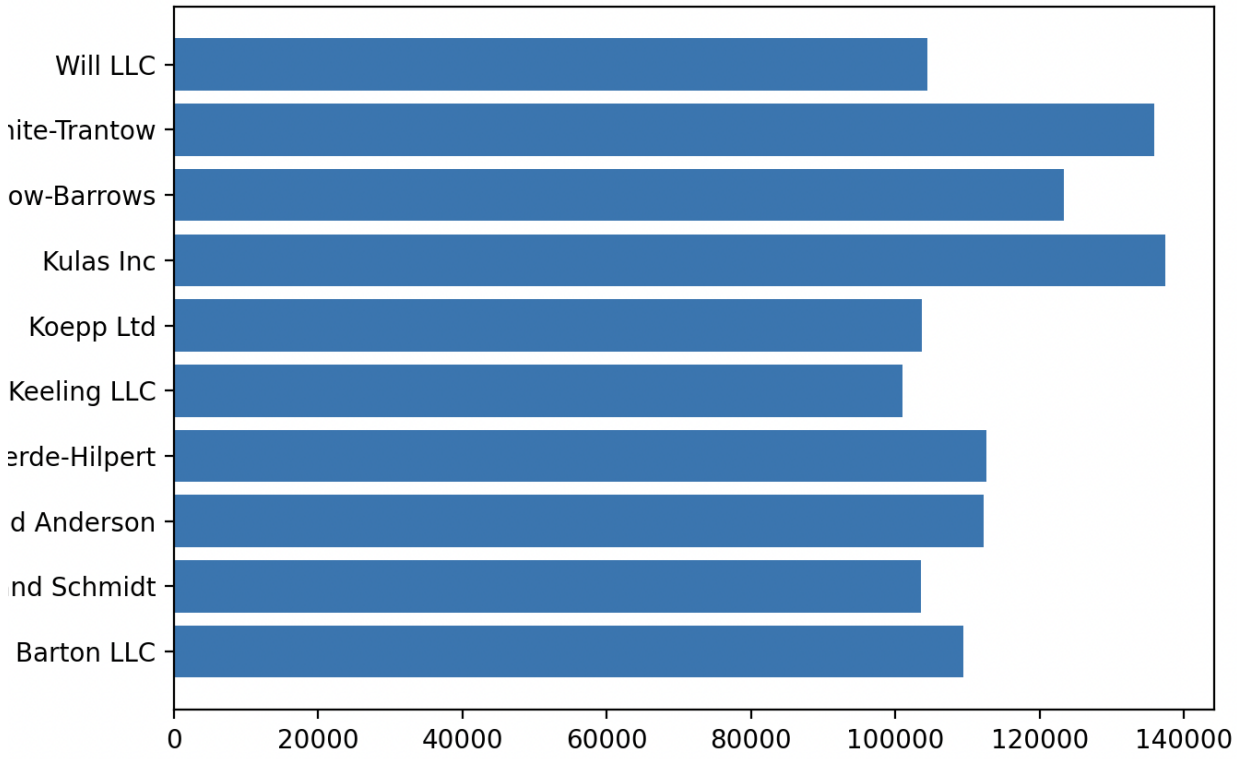
```
57 # sphinx_gallery_thumbnail_number = 10
58 import numpy as np
59 import matplotlib.pyplot as plt
60
61
62 data = {'Barton LLC': 109438.50,
63         'Frami, Hills and Schmidt': 103569.59,
64         'Fritsch, Russel and Anderson': 112214.71,
65         'Jerde-Hilpert': 112591.43,
66         'Keeling LLC': 100934.30,
67         'Koepp Ltd': 103660.54,
68         'Kulas Inc': 137351.96,
69         'Trantow-Barrows': 123381.38,
70         'White-Trantow': 135841.99,
71         'Will LLC': 104437.60}
72 group_data = list(data.values())
73 group_names = list(data.keys())
74 group_mean = np.mean(group_data)
```

Creation of the figure instance

fig, ax = plt.subplots()



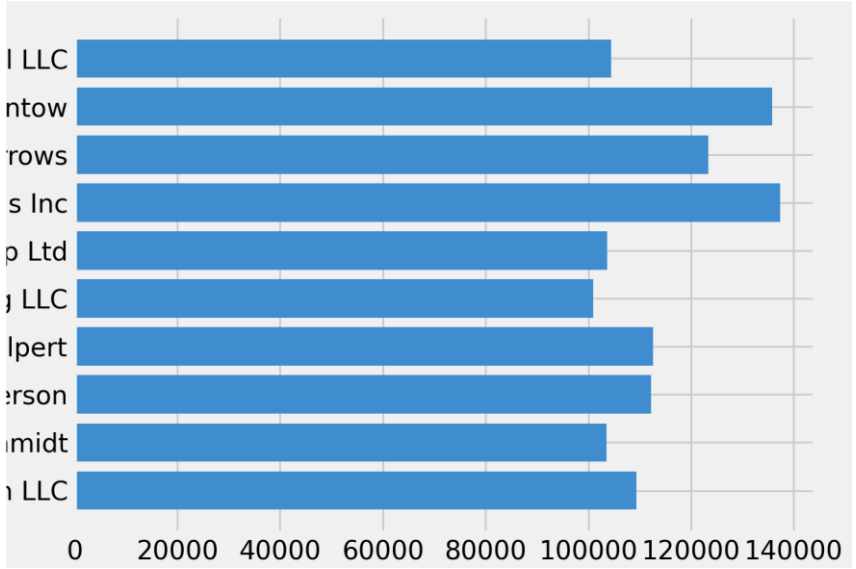
ax.barh(group_names, group_data)



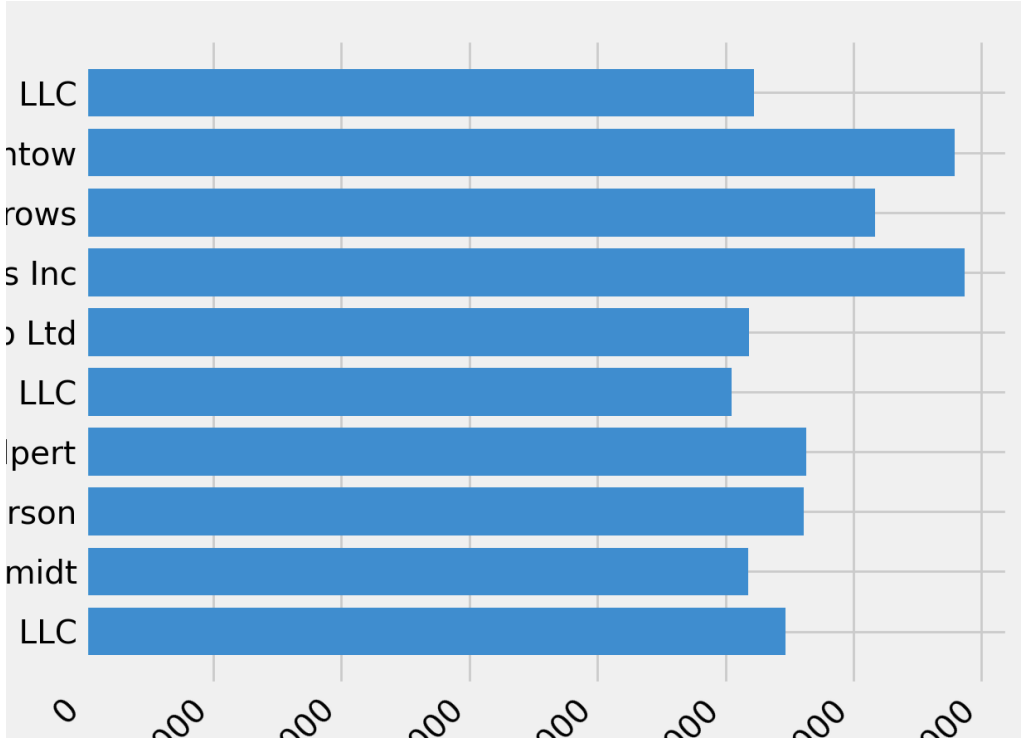
Controlling the style

- The style controls many things, such as color, linewidths, backgrounds, etc.

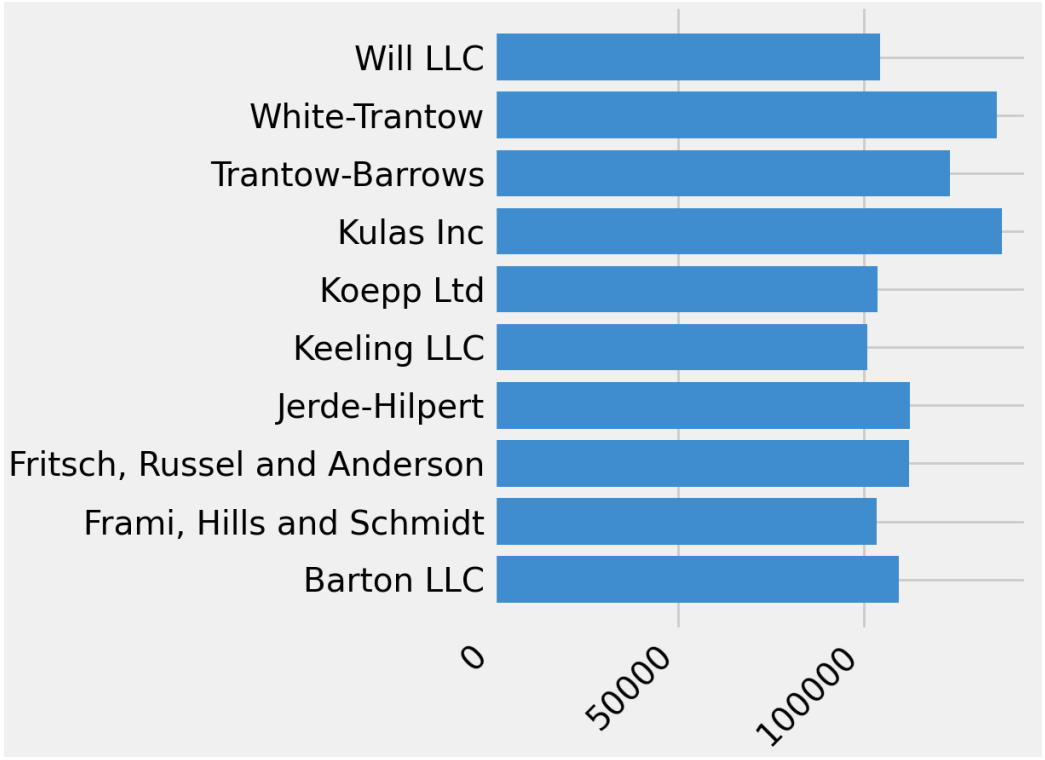
```
print(plt.style.available)
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid'
, 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'gra
yscale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seab
orn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v
0_8-deep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper',
'seaborn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8
-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```



Customising the plot

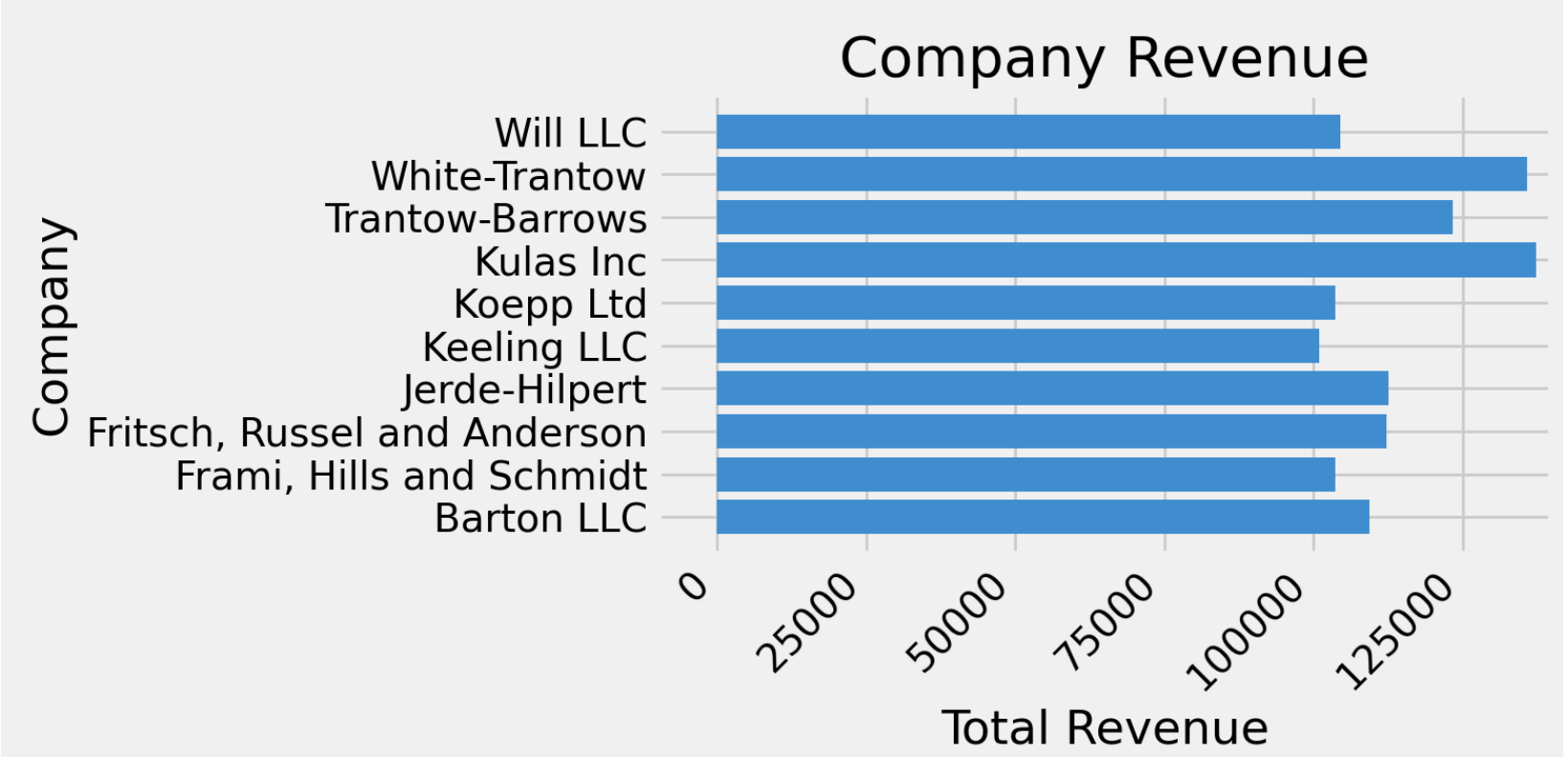



```
fig, ax = plt.subplots()
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
```




```
plt.rcParams.update({'figure.autolayout': True})

fig, ax = plt.subplots()
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
```

Customising the plot




```
fig, ax = plt.subplots(figsize=(8, 4))
ax.barh(group_names, group_data)
labels = ax.get_xticklabels()
plt.setp(labels, rotation=45, horizontalalignment='right')
ax.set(xlim=[-10000, 140000], xlabel='Total Revenue', ylabel='Company',
       title='Company Revenue')
```

Saving our plot

- If you are happy with your plot, you can save it to disk.
- Matplotlib supports many file formats, such as eps, jpg, jpeg, pdf, pgf, png, ps, raw, svg and etc.
- To save your plot, just call the function ‘savefig()’ with the following arguments.
  - transparent=True makes the background of the saved figure transparent if the format supports it
  - dpi controls the resolution (dots per square inch) of the output.
  - Bbox_inches=‘tight’ fits the bounds of the figure to our plot.

```
fig.savefig('example2.png',transparent=False, dpi=80, bbox_inches="tight")
```

Suggested reading

Python

- https://www.python.org/
- https://www.w3schools.com/python/matplotlib_intro.asp
- https://matplotlib.org/stable/tutorials/index.html