06-Graphs slides

Laboratory in Schedule Applied

¶

Math Applied The Economy and the Management

¶

Graphic

s

Matplotlib module

Install PIP3

¶

Introduction to the graphics in Python ¶

THE representation print shop gives information It is often necessary.

O **matplotlib** (https://matplotlib.org/) It is built in Python and uses O numpy for improve The performance about array in bigger dimensions.

O *package* pode ser importado fazendo:

In [ ]:

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

Example:¶

In [14]:

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,2.5])
plt.ylabel('some numbers')
```

Out[14]:

```
Text(0,0.5,'some numbers')
```

plot ¶

The `plot() function` is very versatile and can take an arbitrary number of arguments. For example, to draw *x* by *y* ,to do:

in [16]:

```
plt.plot([1, two, 3, 4], [1, 4, 9, 16])
```

Oct[16]:

```
[<matplotlib.lines.Line2D at 0x114fa5780>]
```

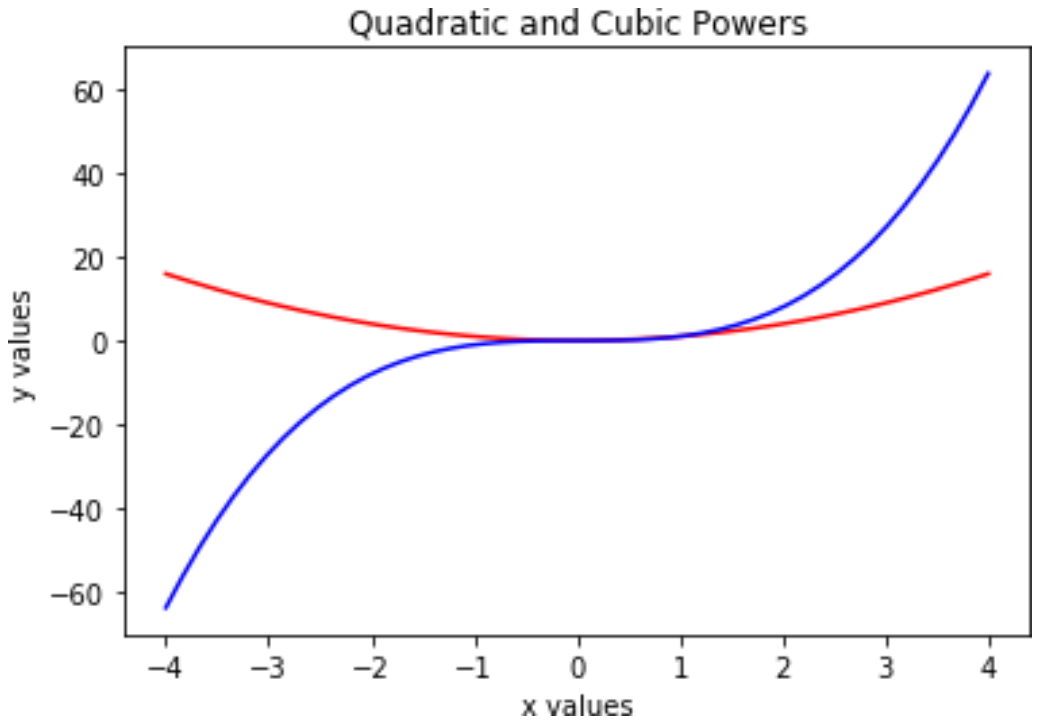
It is Object oriented¶

In [20]:

```
x = np.linspace(-4,4)
y1, y2 = x**2, x**3
```

In [21]:

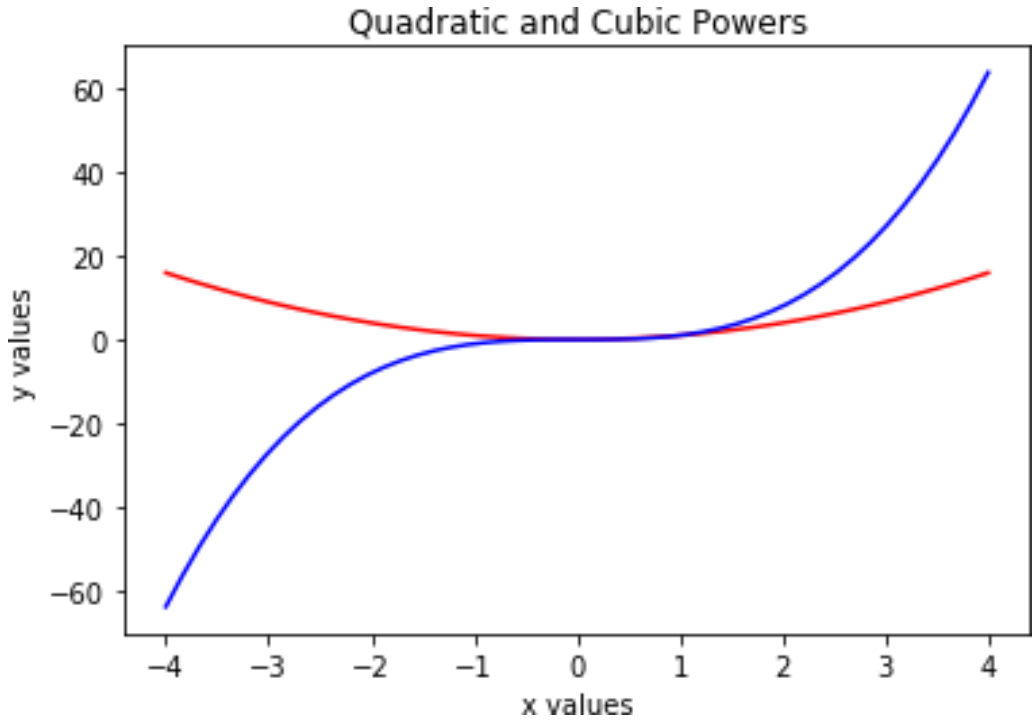
```
fig, ax = plt.subplots()
ax.plot(x, y1, 'red')
ax.plot(x, y2, 'blue')
ax.set_title('Quadratic and Cubic Powers')
ax.set_xlabel('x  values')
ax.set_ylabel('y values');
```



On one line¶

In [22]:

```
plt.plot(x, y1, 'red', x, y2, 'blue')
plt.title('Quadratic and Cubic Powers')
plt.xlabel('x values')
plt.ylabel('y values');
```



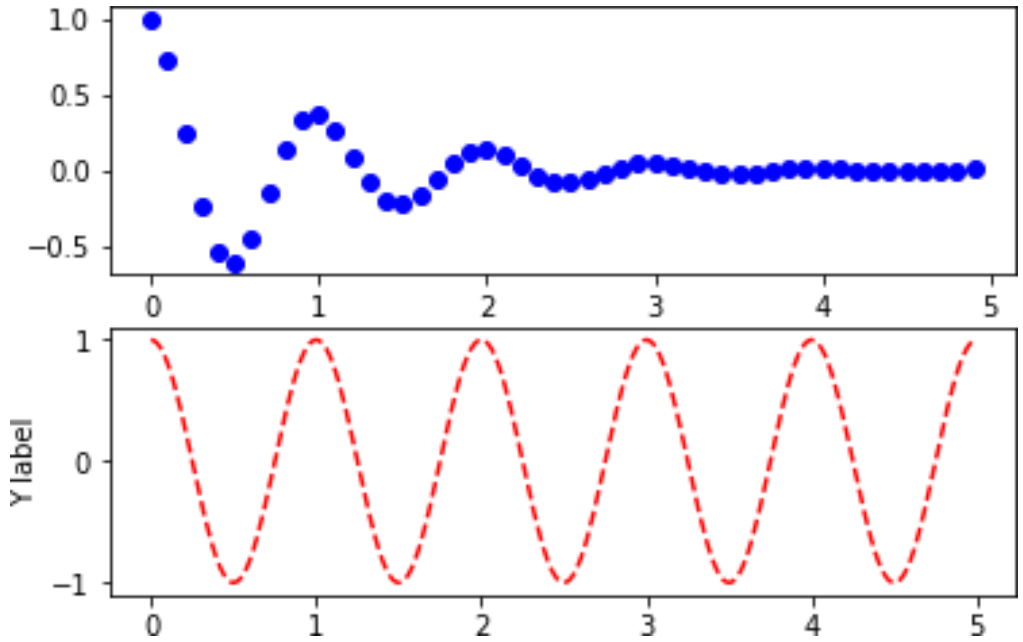
Subplots¶

In [18]:

```
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)
```

In [19]:

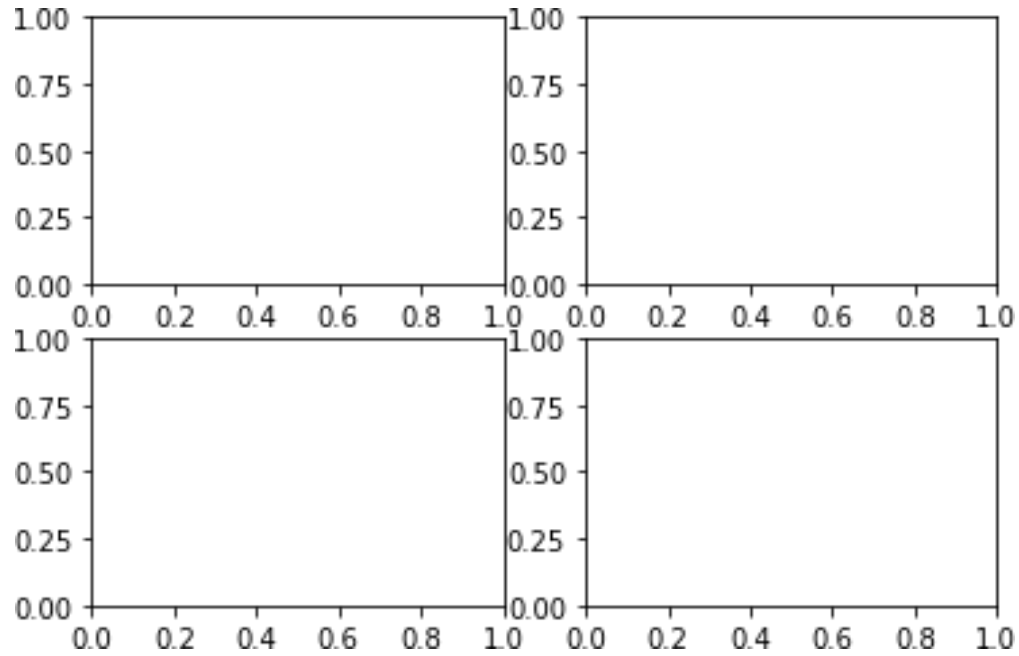
```
plt.figure()
plt.subplot(2, 1, 1)
plt.plot(t1, f(t1), 'bo')

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

plt.ylabel("Y label"); # Which subplot is modifying this function?
```



Subplots¶

In [23]:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
```

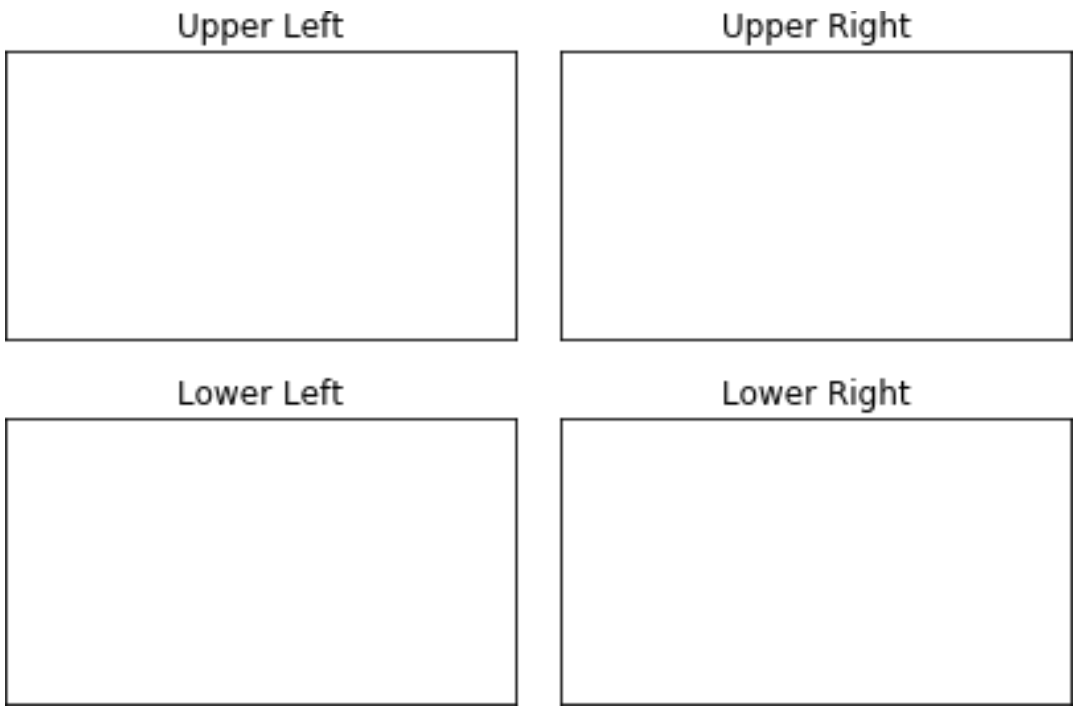


Subplots with title¶

In [24]:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
axes[0,0].set_title('Upper Left')
axes[0,1].set_title('Upper Right')
axes[1,0].set_title('Lower Left')
axes[1,1].set_title('Lower Right')

# To iterate over all items in a multidimensional numpy array, use the `flat` attribute
for ax in axes.flat:
    # Remove all xticks and yticks...
    ax.set(xticks=[], yticks=[])
```

```
fig.tight_layout();
```

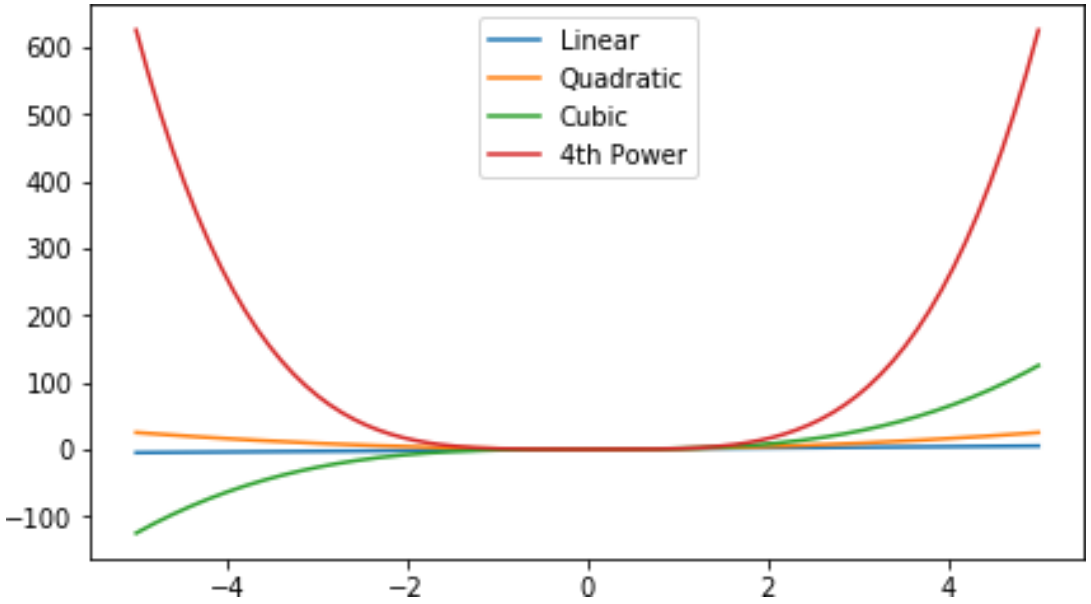


Example: plot of the function power¶

In [25]:

```
x = np.linspace(start=-5, stop=5, num=150)

# All functions in one axes
fig, ax = plt.subplots(figsize = (7,4))
ax.plot(x, x, label='Linear')
ax.plot(x, x**2, label='Quadratic')
ax.plot(x, x**3, label='Cubic')
```

```
ax.plot(x, x**4, label='4th Power')
ax.legend();
```

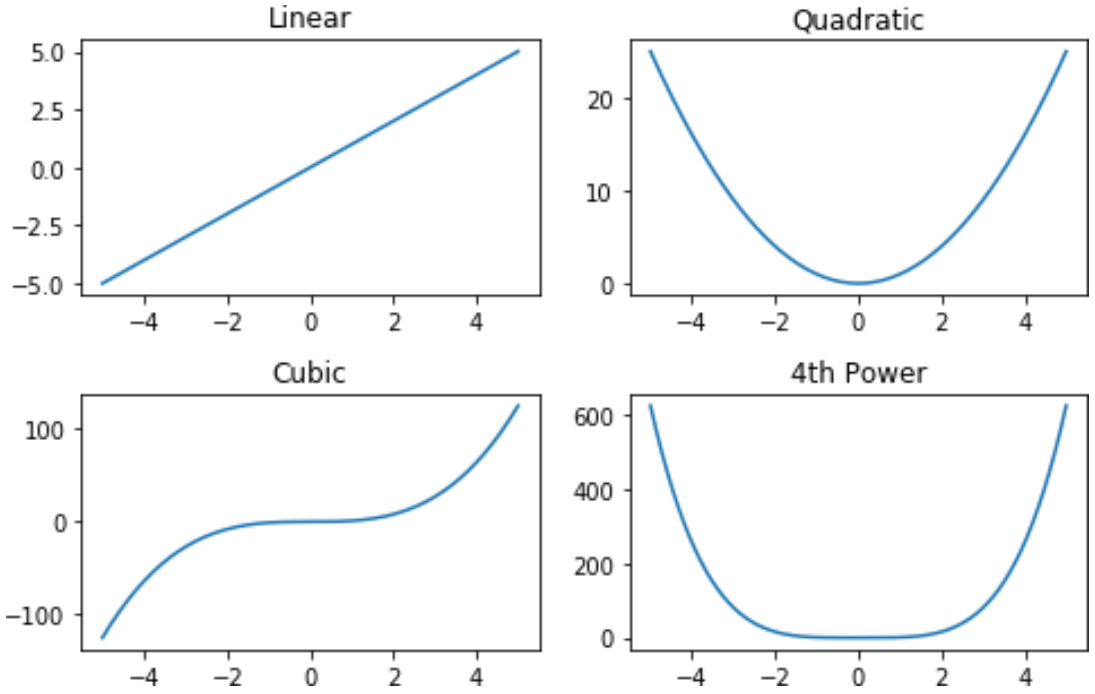


Example: subplots of the power function¶

In [27]:

```
# 4 Axes/Subplots: One function in one axes
fig, axes = plt.subplots(nrows=2, ncols=2, figsize = (7,4.5))
axes[0,0].set_title('Linear')
axes[0,0].plot(x, x)
axes[0,1].set_title('Quadratic')
axes[0,1].plot(x, x**2)
axes[1,0].set_title('Cubic')
axes[1,0].plot(x, x**3)
axes[1,1].set_title('4th Power')
```

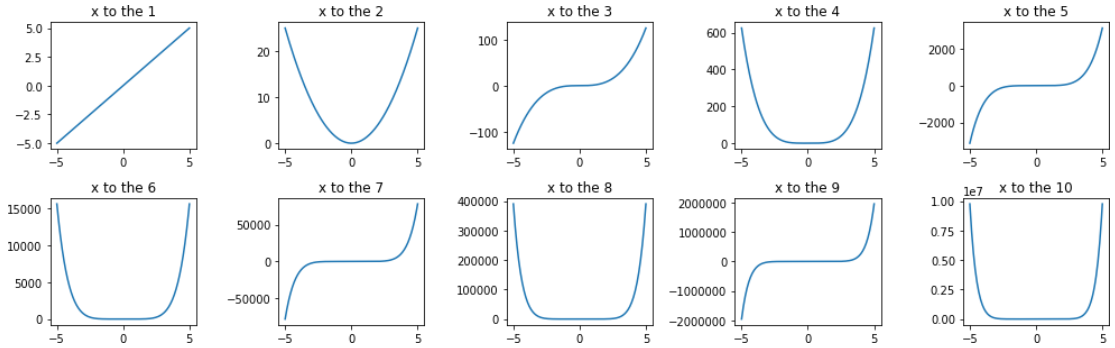
```
axes[1,1].plot(x, x**4)
fig.tight_layout();
```



Example: 10 subplots¶

In [28]:

```
# A more elegant approach
fig, axes = plt.subplots(nrows=2, ncols=5, figsize = (14,4.5))
for i, ax in enumerate(axes.flatten()):
    ax.set_title("x to the {:d}".format(i+1))
    ax.plot(x, x**(i+1))

fig.tight_layout();
```

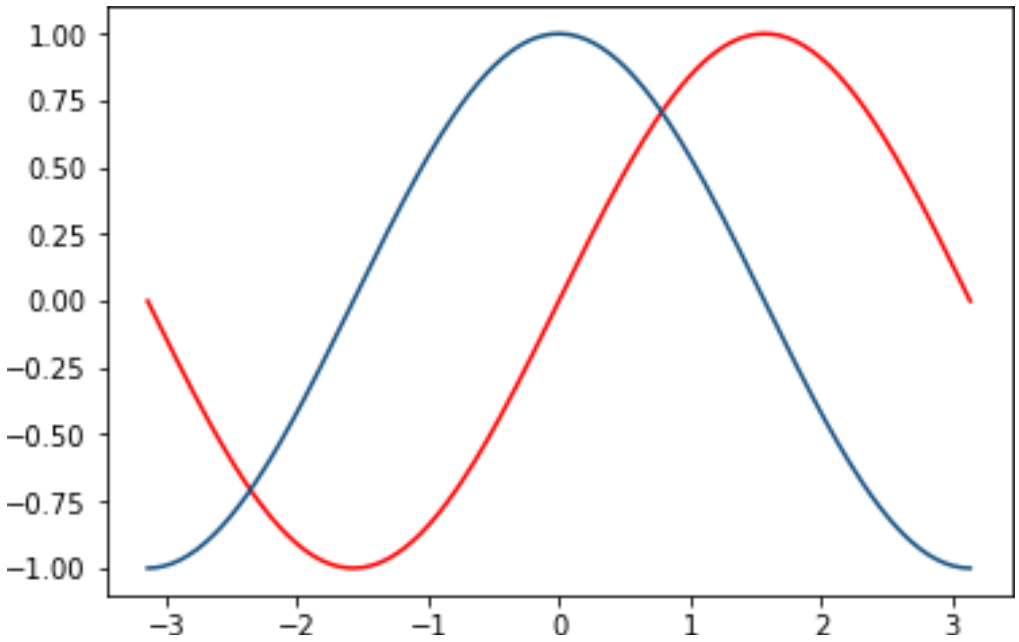


Colors¶

- b: blue
- g: green
- r: red
- c: cyan
- m: magenta
- y: yellow
- k: black
- w: white

In [37]:

```
fig, ax = plt.subplots()
ax.plot(x, sine, color='red')
ax.plot(x, cosine, color='#165181');
```

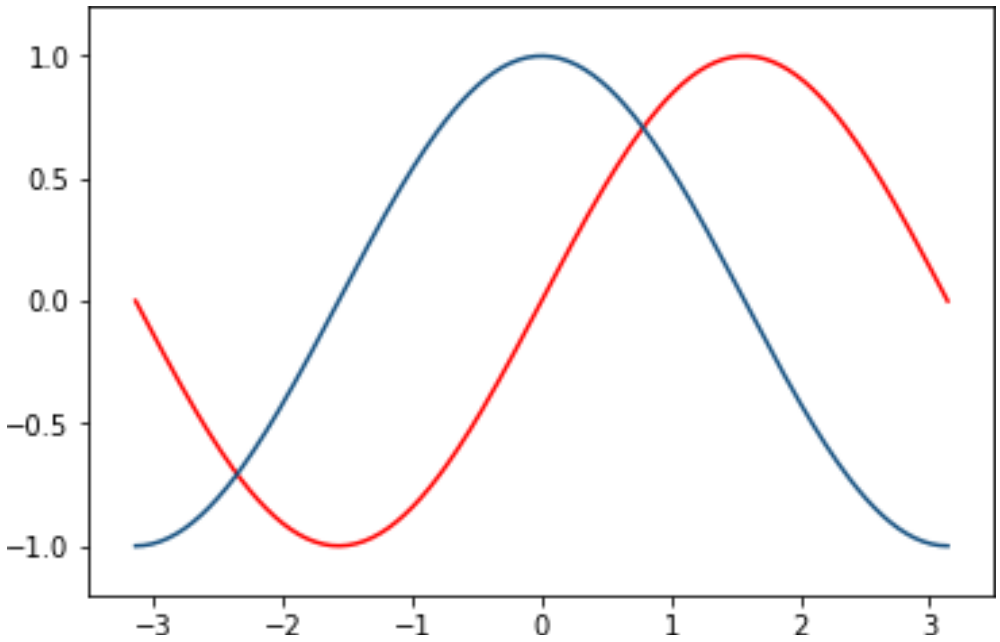


¶

Limites of axis¶

In [39]:

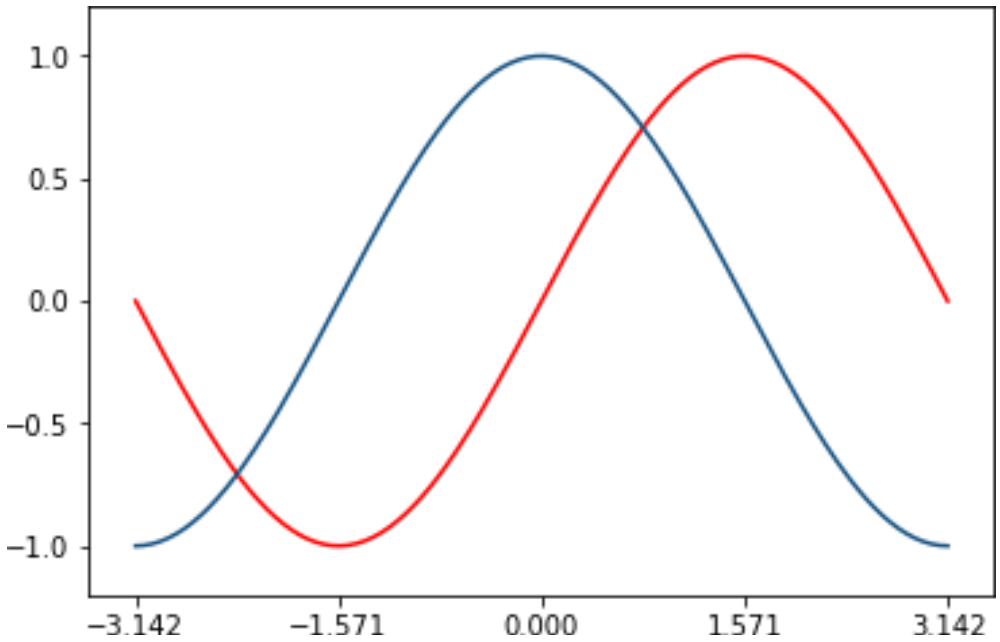
```
fig, ax =  plt.subplots()
ax.plot(x, sine, color='red')
ax.plot(x, cosine, color='#165181')
ax.set_xlim(-3.5,3.5)
ax.set_ylim(-1.2,1.2);
```



Value intervales on axis¶In

[40]:

```
fig, ax =  plt.subplots()
ax.plot(x, sine, color='red')
ax.plot(x, cosine, color='#165181')
ax.set_xlim(-3.5,3.5)
ax.set_ylim(-1.2,1.2);

ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks(np.arange(-1,1.1,0.5));
```



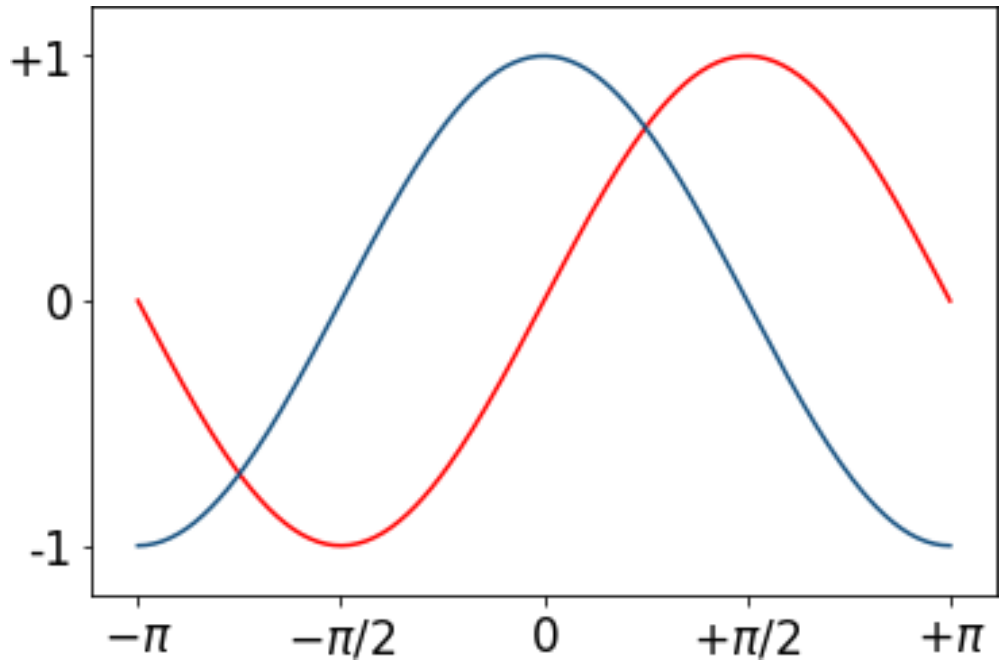
Some Customizations ¶

Range of values on the ¶ axes

in [38]:

```
fig, ax =  plt.subplots()
ax.plot(x, sine, color='red')
ax.plot(x, sew, color='#165181')

ax.set_xlim(-3.5,3.5)
ax.set_ylim(-1.2,1.2)

ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks([-1,0,1])
```

```
ax.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'], size=17)
ax.set_yticklabels(['-1','0','+1'], size=17);
```



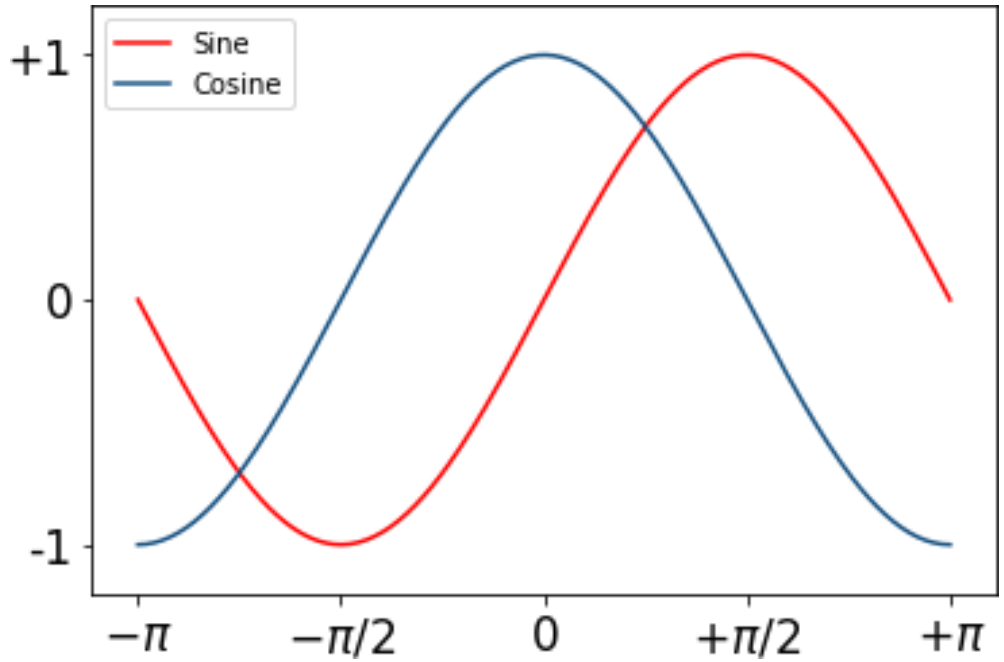
Personalize your graphs¶

Legends¶

In [41]:

```
fig, ax = plt.subplots()
ax.plot(x, sine, color='red', label='Sine')
ax.plot(x, cosine, color='#165181', label='Cosine')

ax.set_xlim(-3.5,3.5)
```

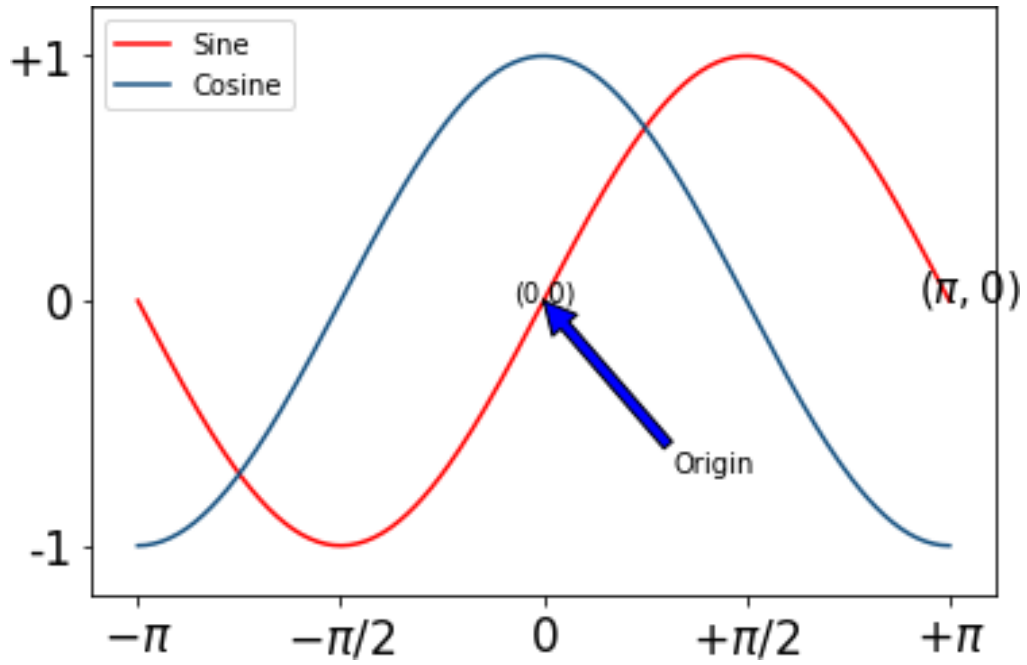
```
ax.set_ylim(-1.2,1.2)

ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks([-1,0,1])

ax.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'], size=17)
ax.set_yticklabels(['-1','0','+1'], size=17)

ax.legend(loc='upper left');
```



Annotations¶

In [43]:

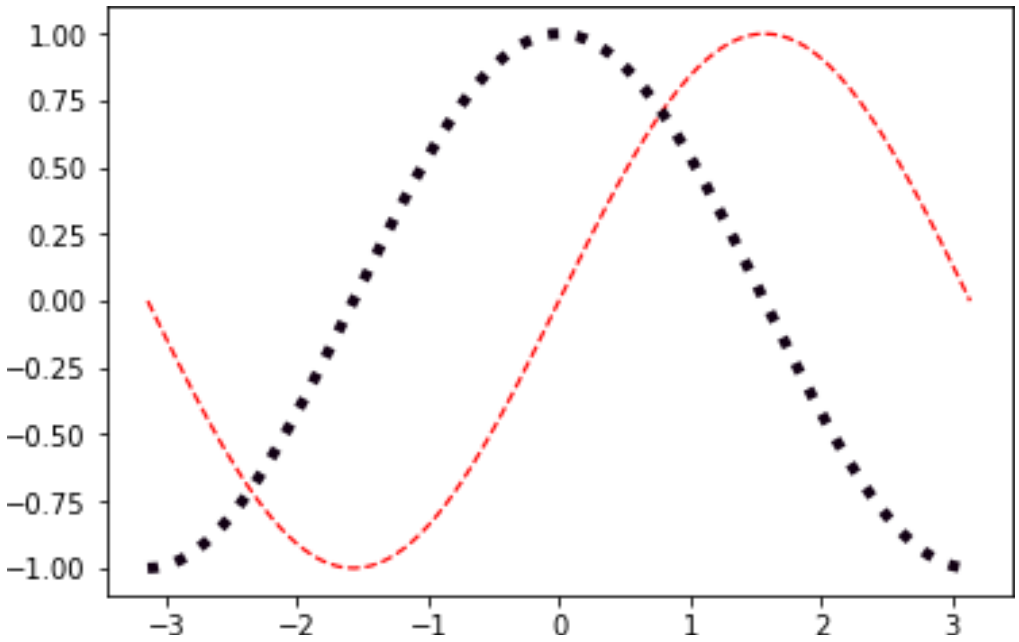
```
fig, ax = plt.subplots()
ax.plot(x, sine, color='red', label='Sine')
ax.plot(x, cosine, color='#165181', label='Cosine')

ax.set_xlim(-3.5,3.5)
ax.set_ylim(-1.2,1.2)

ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks([-1,0,1])

ax.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'], size=17)
ax.set_yticklabels(['-1','0','+1'], size=17)

ax.legend(loc='upper left')

ax.text(-0.25,0,'(0,0)') # x coord, y coord,
ax.text(np.pi-0.25,0, r'$(\pi,0)$', size=15)

ax.annotate('Origin',xy=(0, 0), xytext=(1, -0.7), arrowprops=dict(facecolor='blue'));
```



Line graphs

In [35]:

```
x = np.linspace(-np.pi, np.pi, 200)
sine, cosine = np.sin(x), np.cos(x)

fig, ax = plt.subplots()
ax.plot(x, sine, color='red', linestyle='--', linewidth=1.2)
ax.plot(x, cousin, color='#110013', linestyle=':', linewidth=4);
```



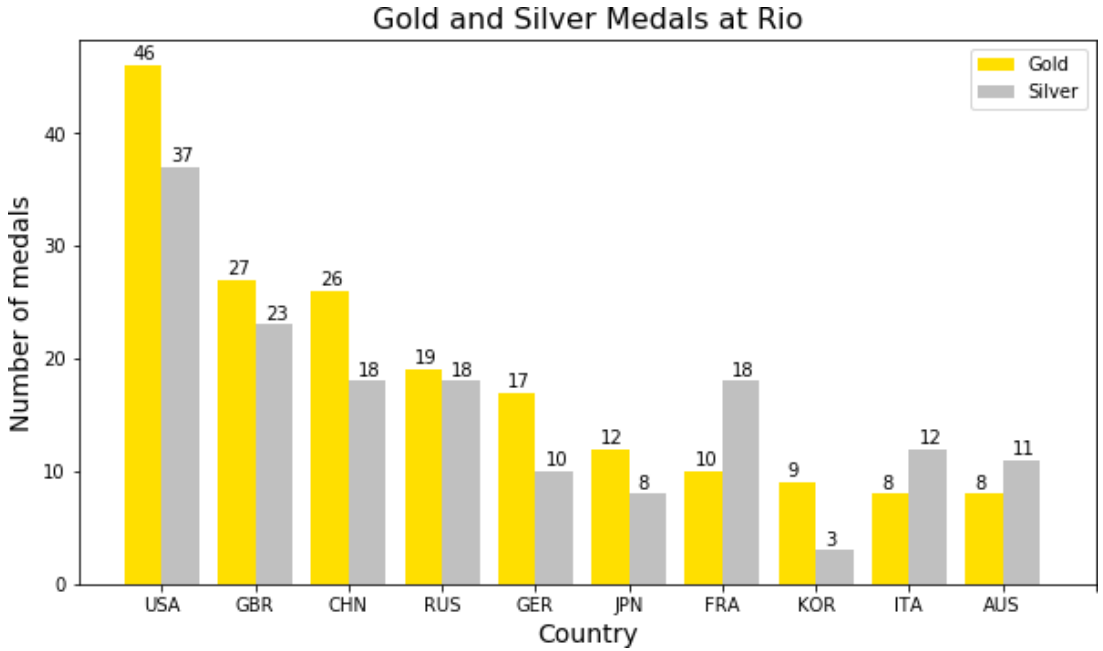
¶Bar graph ¶ In [ ]:

```
# Top 10 countries in the Rio Olympics
countries = ['USA','GBR','CHN','RUS','GER','JPN','FRA','KOR','ITA','AUS']
gold = [46,27,26,19,17,12,10,9,8,8]
silver = [37,23,18,18,10,8,18,3,12,11]
bronze = [38,17,26,19,15,21,14,9,8,10]
```

Bar plot

In [29]:

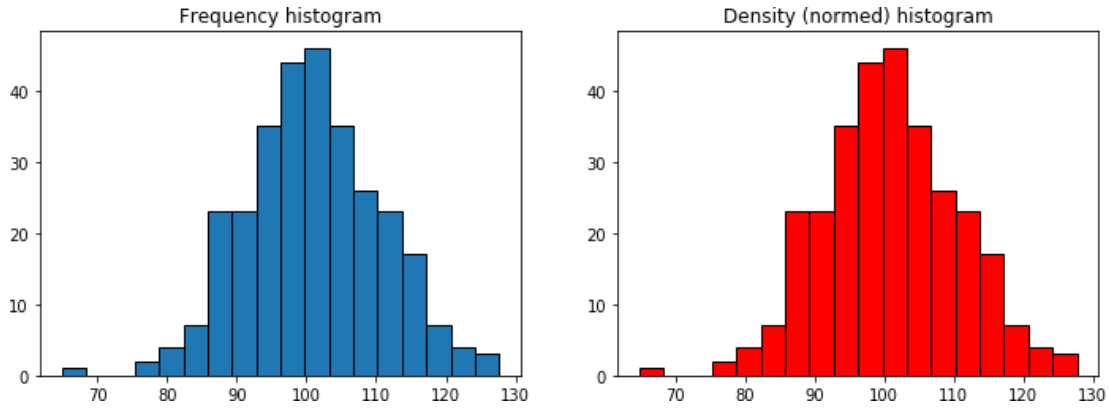
```
fig, ax = plt.subplots(figsize=(10,5.5))
ax.bar(np.arange(10), gold, color="#FFDF00", width=0.4, label='Gold')
ax.bar(np.arange(10)+0.4, silver, color="#C0C0C0", width=0.4, label='Silver')

ax.set_xticks(np.arange(0.2,10.4, 1))
ax.set_xticklabels(countries);

for x,g,s in zip(np.arange(10), gold, silver):
    ax.text(x-0.1, g+0.5, g) # annotating the golds
    ax.text(x+0.3, s+0.5, s) # annotating the silvers

ax.set_title('Gold and Silver Medals at Rio', size=16)
ax.set_xlabel('Country', size=14)
ax.set_ylabel('Number of medals', size=14)
ax.legend(loc='upper right');
```



Histogram

In [32]:

```
iqs = np.random.normal(loc=100, scale=10, size=300)
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,4))

ax[0].hist(iqs, bins=18, edgecolor='k')
ax[0].set_title('Frequency histogram')

ax[1].hist(iqs, bins=18, color = 'red', edgecolor='k')
ax[1].set_title('Density (normed) histogram');
```

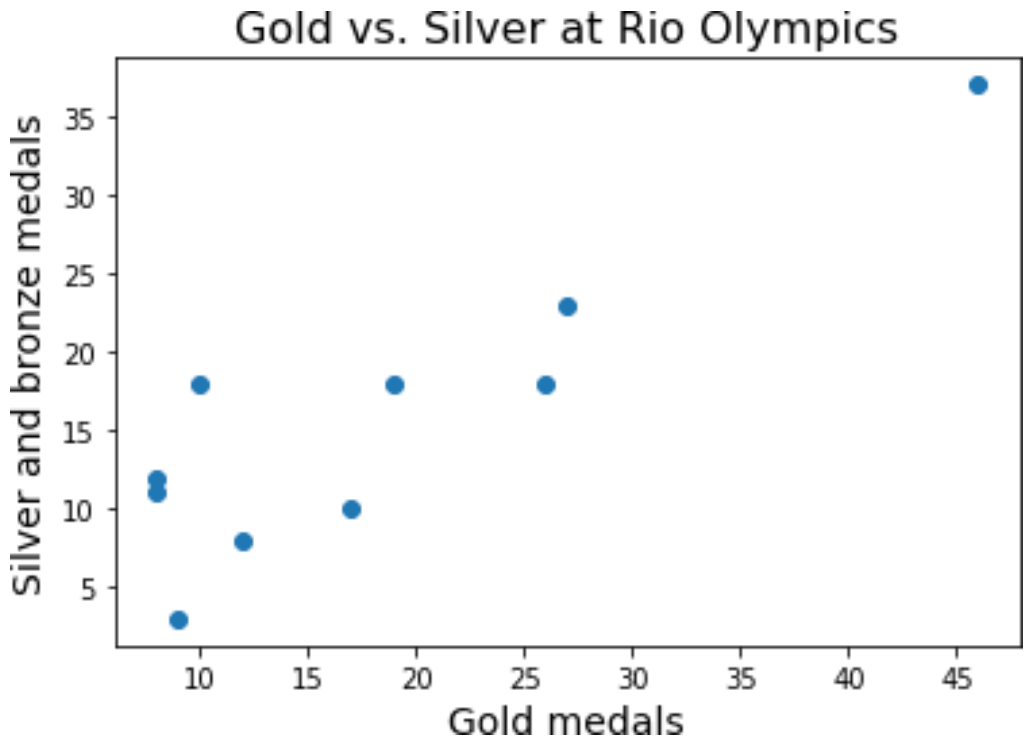


Scatter plot

In [33]:

```
fig, ax = plt.subplots()
ax.scatter(gold, silver, marker='o')
```

```
ax.set_title('Gold vs. Silver at Rio Olympics', size=16)
ax.set_xlabel('Gold medals', size=14)
ax.set_ylabel('Silver and bronze medals', size=14);
```

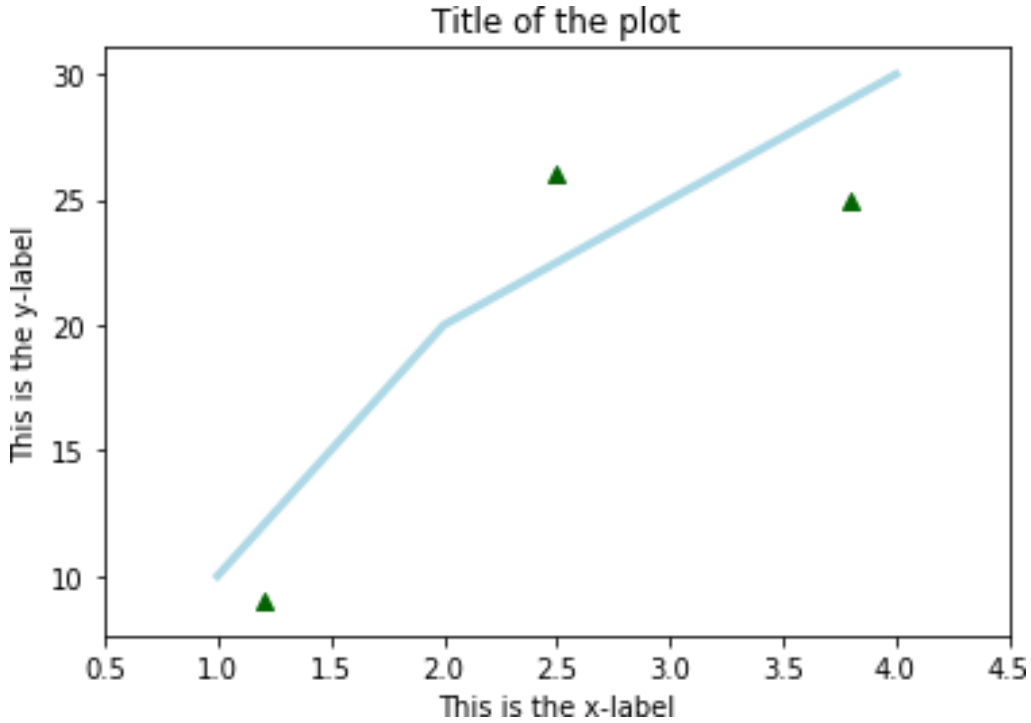


Mixing graphic styles¶

you can mix different types of graphics. Per example, mixing the `plot` what draw lines with the `scatter` in space.

in [17]:

```
plt.plot([1, 2, 3, 4], [10, 20, 25, 30], color='lightblue', linewidth=3)
plt.scatter([0.3, 3.8, 1.2, 2.5], [11, 25, 9, 26], color='darkgreen', marker='^')
plt.xlim(0.5, 4.5)
plt.title("Title of the plot")
plt.xlabel("This is the x-label")
plt.ylabel("This is the y-label");
```



Creating Pie Charts

With Pyplot, you can use the `pie()` function to draw pie charts:

Example

A simple pie chart:

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

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

Result:

Try it Yourself »