In [2]:

```
# The crosstab() function is used to plot cross-tabulation
# between two columns. Let's import the Titanic dataset from
# the Seaborn library and plot a cross tab matrix between
# passenger class and age columns for the Titanic dataset.
# Look at the following two scripts on how to do that:

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# sets the default style for plotting
sns.set_style("darkgrid")
titanic_data = sns.load_dataset('titanic')
```

In [3]:

```
pd.crosstab(titanic_data["class"], titanic_data["age"], margins=True)
```

Out[3]:

age	0.42	0.67	0.75	0.83	0.92	1.0	2.0	3.0	4.0	5.0	...	63.0	64.0	65.0	66.0	70.0	70
class																	
First	0	0	0	0	1	0	1	0	1	0	...	1	2	2	0	1	
Second	0	1	0	2	0	2	2	3	2	1	...	0	0	0	1	1	
Third	1	0	2	0	0	5	7	3	7	3	...	1	0	1	0	0	
All	1	1	2	2	1	7	10	6	10	4	...	2	2	3	1	2	

4 rows × 89 columns

In [4]:

```
# Discretization and Binning
# Discretization or binning refers to creating categories or bins
# using numeric data. For instance, based on age, you may want
# to assign categories such as toddler, young, adult, and senior
# to the passengers in the Titanic dataset. You can do this using
# binning.
# Let's see an example. The following script imports the Titanic
# dataset.

import matplotlib.pyplot as plt
import seaborn as sns
titanic_data = sns.load_dataset('titanic')
```

In [5]:

```
# You can use the cut method from the Pandas dataframe to
# perform binning. First, the column name is passed to the “x”
# attribute. Next, a list containing ranges for bin values is passed
# to the “bins” attribute, while the bin or category names are
# passed to the “labels” parameter.
# The following script assigns a category toddler to passengers
# between the ages 0-5, young to passengers aged 5-20, adult
# to passengers aged 20-60, and senior to passengers aged 60-
# 100.

titanic_data['age_group']=pd.cut(x = titanic_data['age'],
                                 bins = [0,5,20,60,100],
                                 labels = ["toddler", "young", "adult","senior"])
titanic_data['age_group'].value_counts()
```

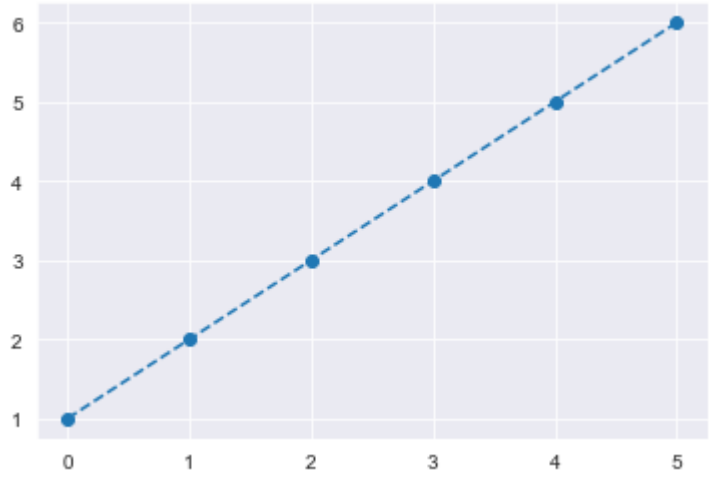
Out[5]:

```
adult      513
young      135
toddler     44
senior      22
Name: age_group, dtype: int64
```

In [13]:

```
# Visualization with NumPy and Matplotlib
# You are now going to learn how to create NumPy Ndarrays with Ndarray creation
# routines and then use Matplotlib to visualize them.
# The first routine is arange(). It creates evenly spaced values with the given interval
# A stop value argument is compulsory. The start value and interval parameters have the
# default arguments 0 and 1, respectively. Here's an example:

import numpy as np
import matplotlib.pyplot as plt
x = np.arange(6)
print(x)
print(type(x))

# Let's go ahead and plot these numbers. For plotting in 2D, we need x-y pairs. Let's
# keep it simple and say y = f(x) = x by running the following statement:

y=x+1

# Now, let's use the function plot() to visualize this. It needs the values of x and y a
# the plotting options.

plt.plot(x, y, 'o--')
plt.show()

# The function show() displays the plot.
```

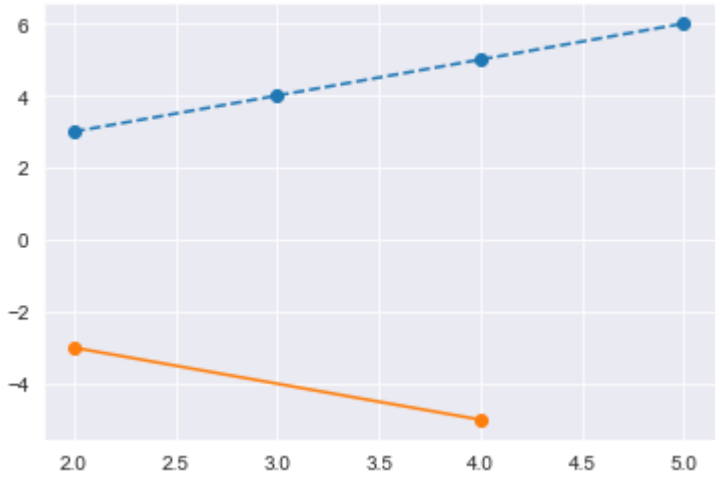
```
[0 1 2 3 4 5]
<class 'numpy.ndarray'>
```



In [14]:

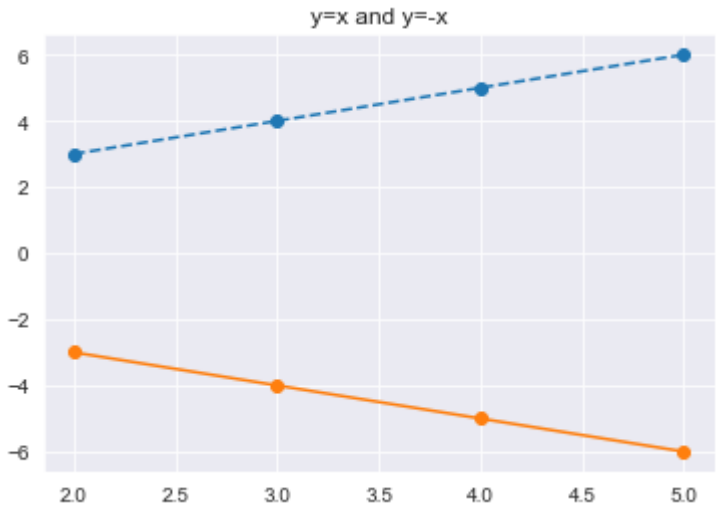
```
# Here's an example of the function call for the function arange() with the start and
# stop arguments:

x = np.arange(2, 6)
print(x)
y = x + 1

# We can even add an argument for the interval as follows:

a= np.arange(2, 6, 2)
print(a)
b = a + 1

# We can draw multiple graphs as follows:

plt.plot(x, y, 'o--')
plt.plot(a, -b, 'o-')
plt.show()

# The output will have one line and another dashed line.
```

```
[2 3 4 5]
[2 4]
```



In [15]:

```
 # You can even add a title to the graph as follows:

plt.plot(x, y, 'o--')
plt.plot(x, -y, 'o-')
plt.title('y=x and y=-x')
plt.show()

# The output will have a title as shown.
```



In [16]:

```
# The function linspace(start, stop, number) returns an array of evenly spaced
# numbers over a specified interval. You must pass it the starting value, the end value,
# the number of values as follows:

N = 16
x = np.linspace(0, 15, N)
print(x)

# The previous code creates 11 numbers (0 to 10, both inclusive) as follows:
# [ 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.]
# Let's visualize this as follows:

y = x
plt.plot(x, y, 'o--')
plt.axis('off')
plt.show()

# The output will be:
```

```
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15.]
```