Lecture 8 – Pandas, matplotlib

Mr. Yousif G. Arshak

University of Zakho

Computer Science Department

yousif.arshak@uoz.edu.krd

OUTLINES

- Pandas
  - Access rows and columns in DataFrame
  - Dropna and fillna
  - Operations
- Matplotlib

Access DataFrame index

```
import pandas as pd
import numpy as np
from numpy.random import randn
np.random.seed(101) # to get the same random numbers
df = pd.DataFrame(randn(5,4),['A','B','C','D','E'], ['W','X','Y','Z’])
print(df)
```

Access columns

- Access one column
  - df[‘W’]
- Access list of Columns
  - df[['W','Y']]

Access rows

- Access one row
  - df.loc['A']
- Access list of rows
  - df.loc[['A','B']]

Access rows and columns by index

- Access one column
  - df.iloc[:,0]
- Access one row
  - df.iloc[0,:]
- Access one cell
  - df.iloc[0,0]
- Access number of rows and columns
  - df.iloc[0:2,1:3]

Missing Data

- In pandas we use dropna() to remove any rows or columns with non values
- d={'A':[1,2,np.nan],'B':[3,np.nan,np.nan],'C':[1,2,3]}
- df=pd.DataFrame(d)
- df.dropna()

	A	B	C	
0	1.0	3.0	1	

Df.dropna() By default any rows with non value will be removed, we can remove columns:
Df.dropna(axis=1)

Drop any rows with two or more missing values

- df.dropna(thresh=2)

	A	B	C	
0	1.0	3.0	1	
1	2.0	NaN	2	

Fill in non values

- df.fillna(999)

	A	B	C	
0	1.0	3.0	1	
1	2.0	999.0	2	
2	999.0	999.0	3	

- df.fillna(df.mean())

	A	B	C	
0	1.0	3.0	1	
1	2.0	3.0	2	
2	1.5	3.0	3	

Operations

```
df=pd.DataFrame({'col1':[1,2,3,4],
                 'col2':[44,55,66,55],
                 'col3':['abc','def','ghi','jkl']})
```

- df.head(2)

	col1	col2	col3
0	1	44	abc
1	2	55	def

- df['col2'].unique() #array([44, 55, 66], dtype=int64)
- df['col2'].nunique() #3
- df['col2'].value_counts()

```
#55 2
 66 1
 44 1 Name: col2, dtype: int64
```

- df['col1'].sum() #10
- def times(x):

  return x*2

- df['col1'].apply(times)
- df['col1'].apply(lambda x: x*2)

```
0 2
1 4
2 6
3 8 Name: col1, dtype: int64
```

Drop Rows and Columns

- Drop row by inded
  - df.drop(0)
- Drop Column
  - df.drop('col1',axis=1) #use inplace = True to remove from its source

Sorting

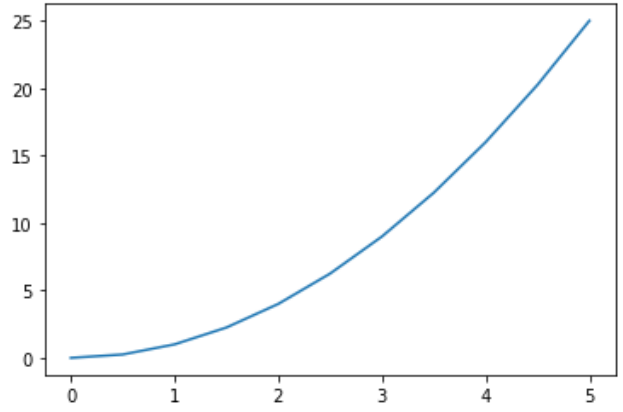
- df.sort_values('col2')

Matplotlib

- It’s the most famous plotting library for python
- It gives you control over every aspect of a figure
- To install it use the below code
  - Python –m pip install matplotlib
  - Or
  - Coda install matplotlib
- To know more check out matplotlib.org

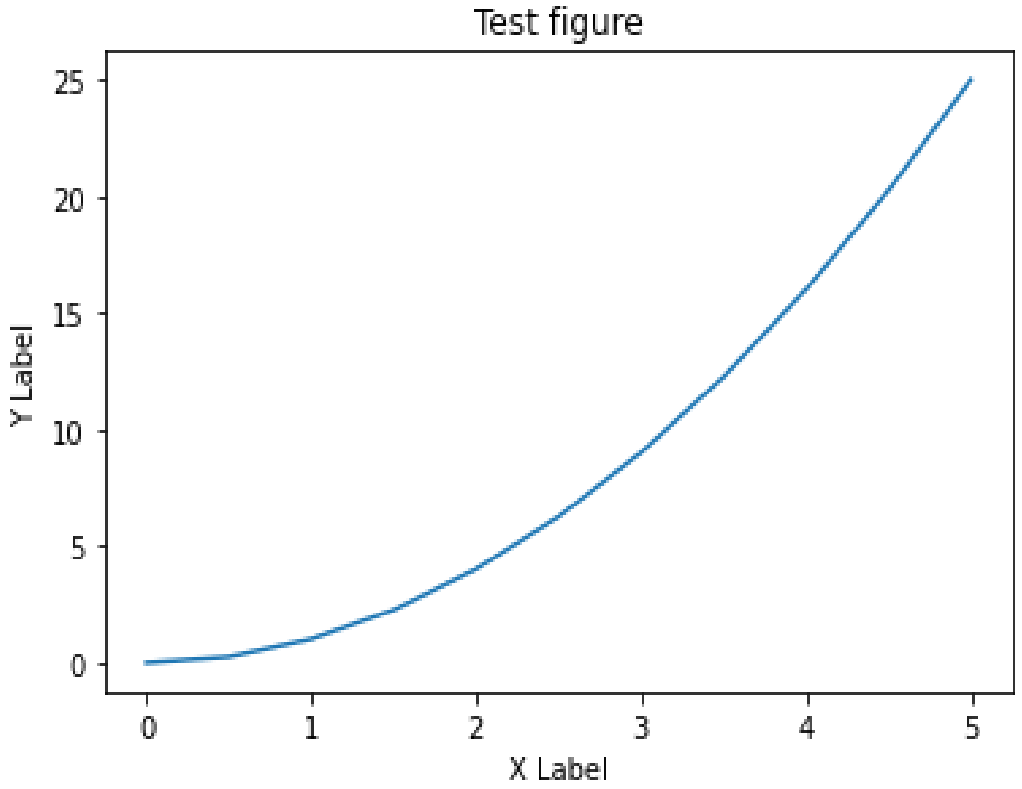
example

- import matplotlib.pyplot as plt
- %matplotlib inline # to show the figures inside jupyter notebook
- import numpy as np
- x=np.linspace(0,5,11)
- y=x**2
- #Functional method to show figures
- plt.plot(x,y)
- plt.show() # to printing out figure



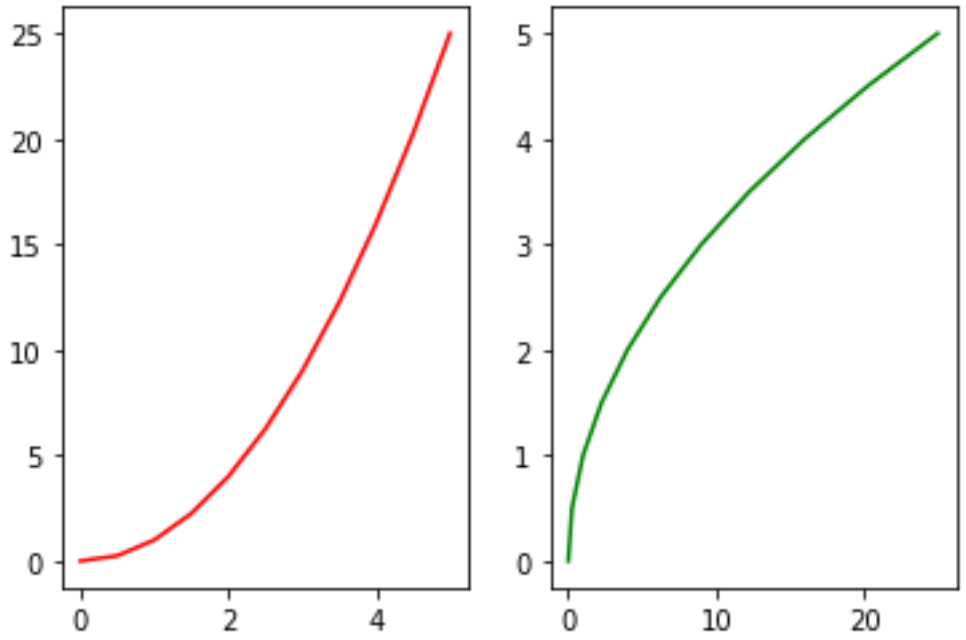
#Functional method to show figures

```
plt.plot(x,y)
plt.xlabel('X Label')
plt.ylabel('Y Label')
plt.title('Test figure')
```



Showing multiple plot

- plt.subplot(1,2,1)
- plt.plot(x,y,'r')
- plt.subplot(1,2,2)
- plt.plot(y,x,'g')



plt.subplot(number of row, number of column, plot position)

Object Oriented method to show figures

- fig = plt.figure()
- axes = fig.add_axes([0.1,0.1,0.8,0.8])

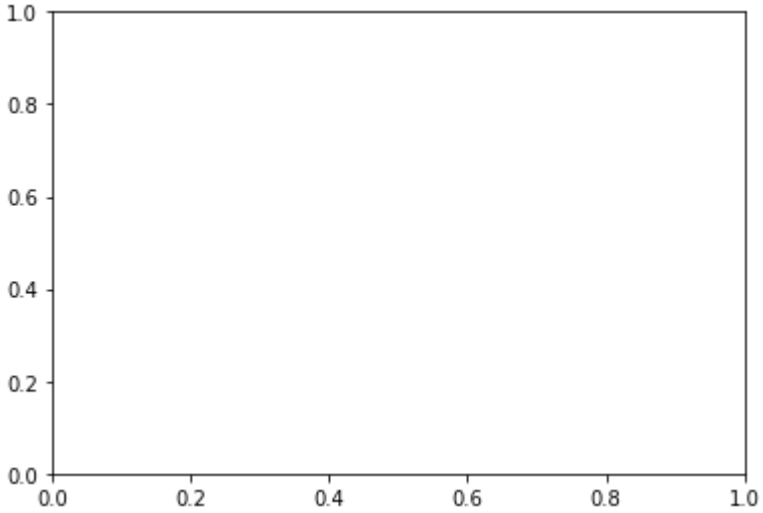
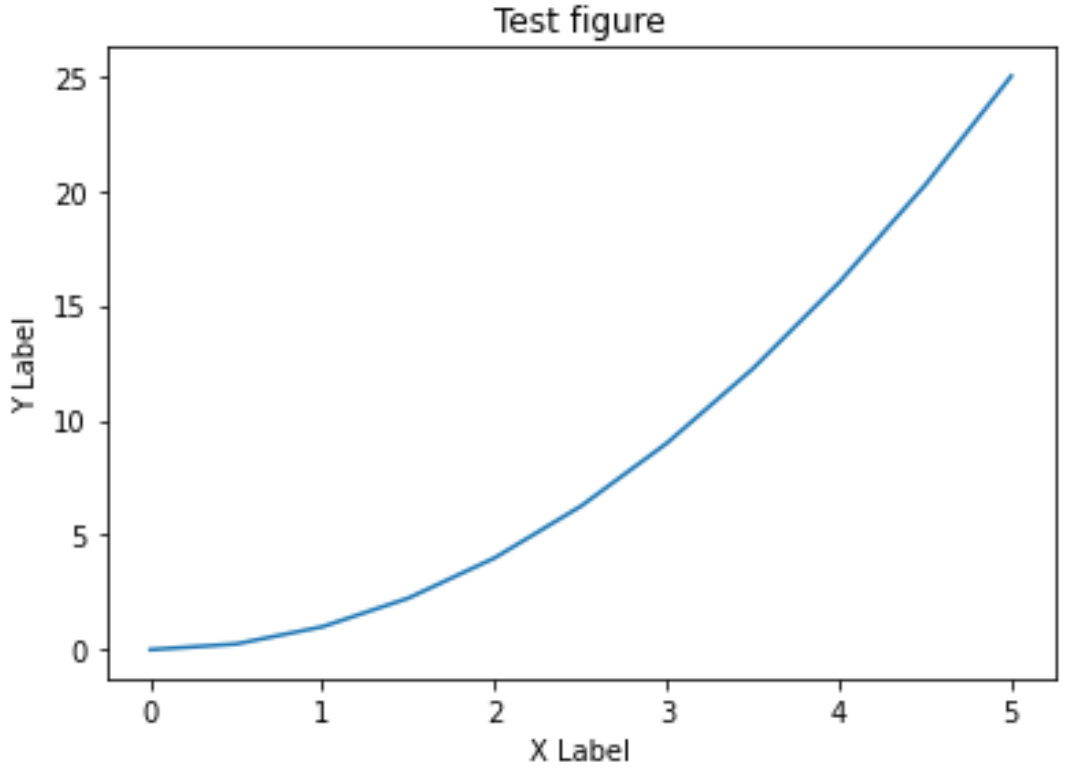


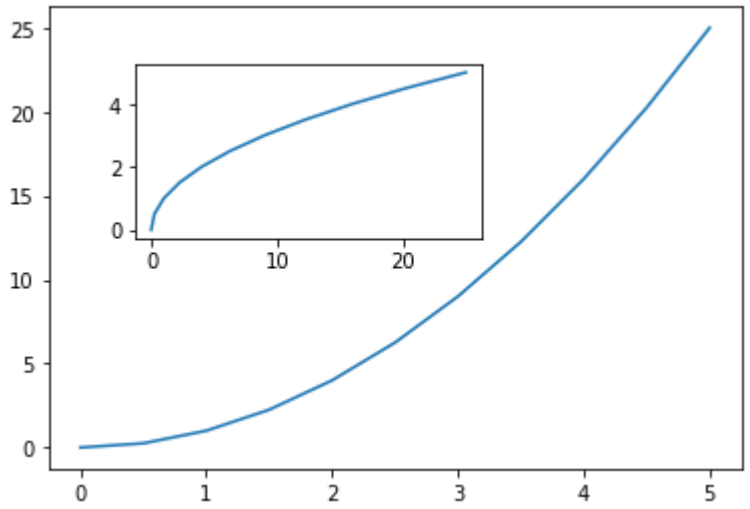
fig.add_axes([left,bottom,width,height])

- fig = plt.figure()
- axes = fig.add_axes([0.1,0.1,0.8,0.8])
- axes.plot(x,y)
- axes.set_xlabel('X Label')
- axes.set_ylabel('Y Label')
- axes.set_title('Test figure')



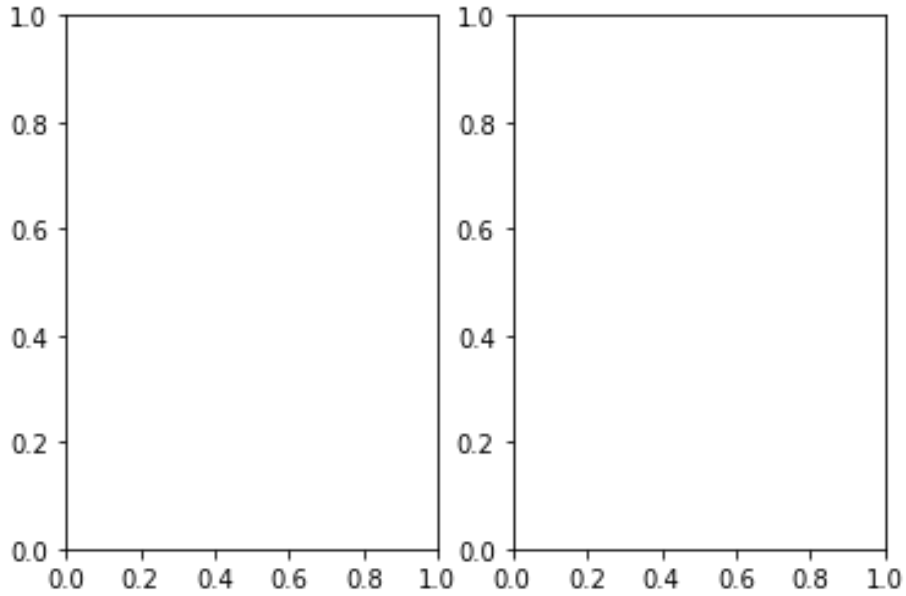
Nested figures

- fig = plt.figure()
- axes1 = fig.add_axes([0.1,0.1,0.8,0.8])
- axes1.plot(x,y)
- axes2 = fig.add_axes([0.2,0.5,0.4,0.3])
- axes2.plot(y,x)



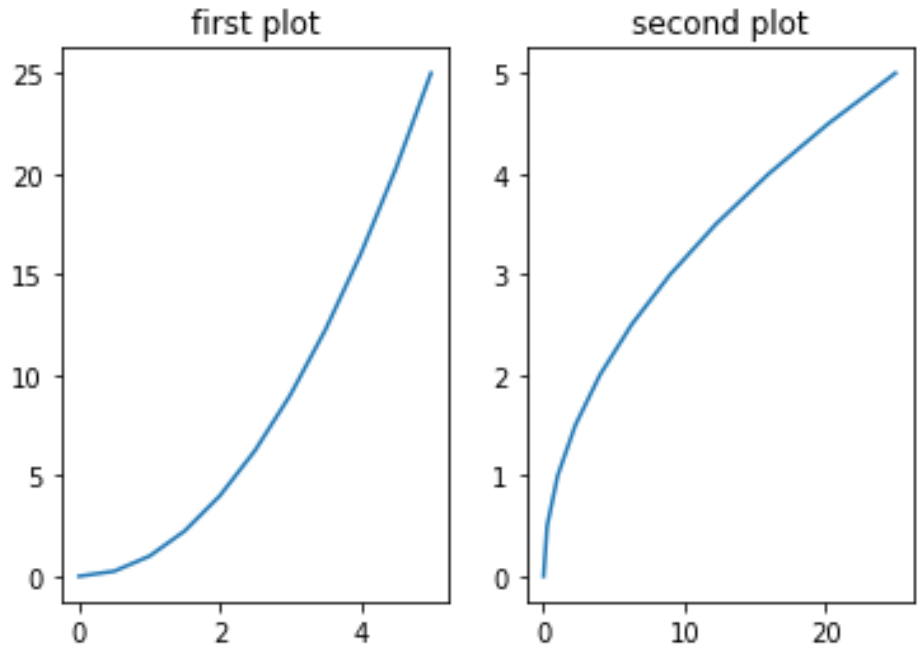
Rows and columns

- fig,axes = plt.subplots(nrows=1,ncols=2)



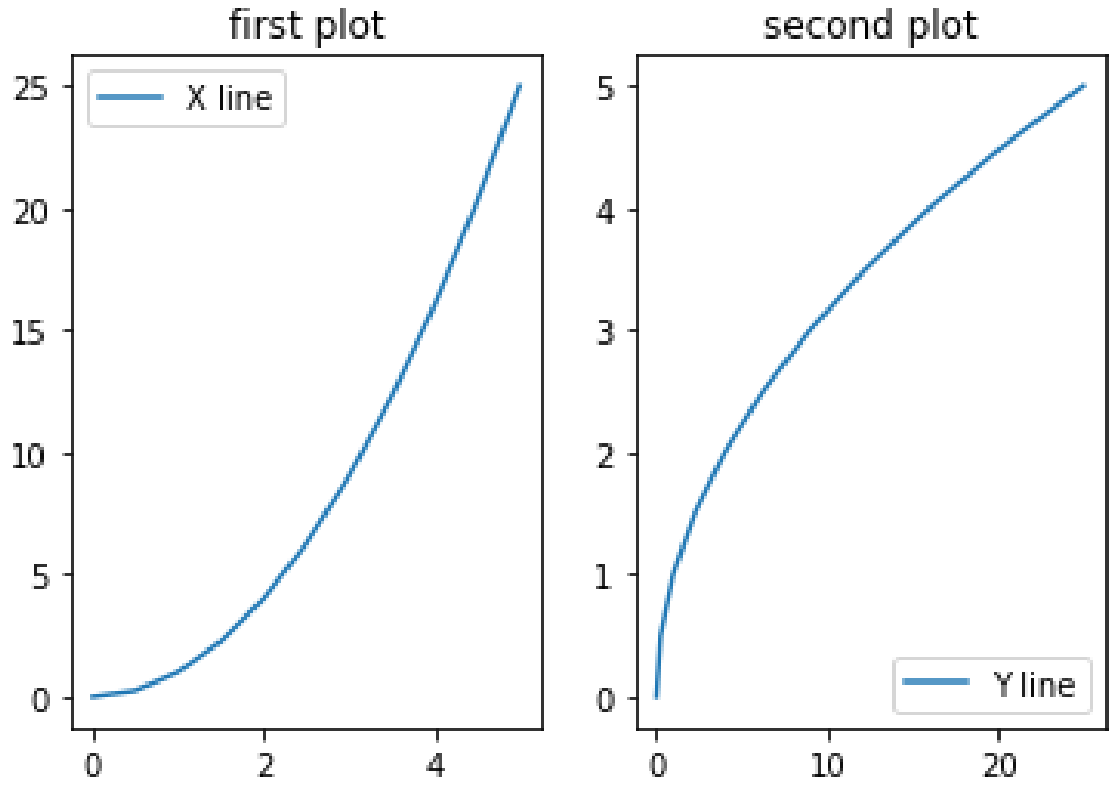
```
fig,axes = plt.subplots(nrows=1,ncols=2)
axes[0].plot(x,y)
axes[0].set_title('first plot')

axes[1].plot(y,x)
axes[1].set_title('second plot')
```

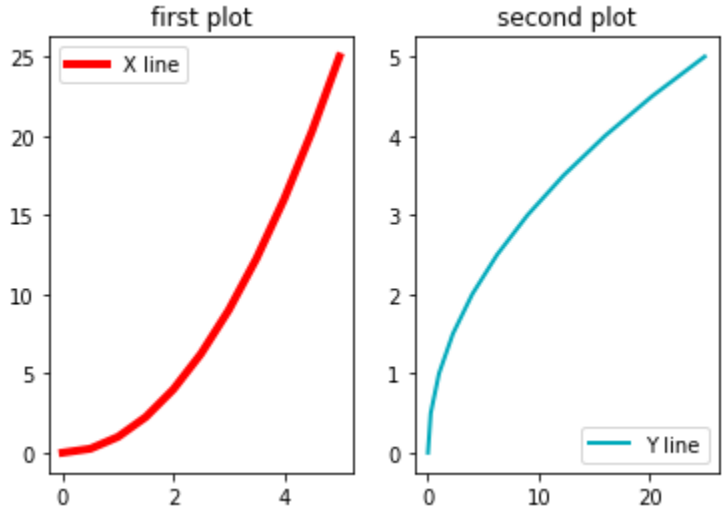


- fig,axes = plt.subplots(nrows=1,ncols=2)
- axes[0].plot(x,y, label='X line')
- axes[0].set_title('first plot')
- axes[0].legend(loc=0)

- axes[1].plot(y,x, label='Y line')
- axes[1].set_title('second plot')
- axes[1].legend(loc=4)



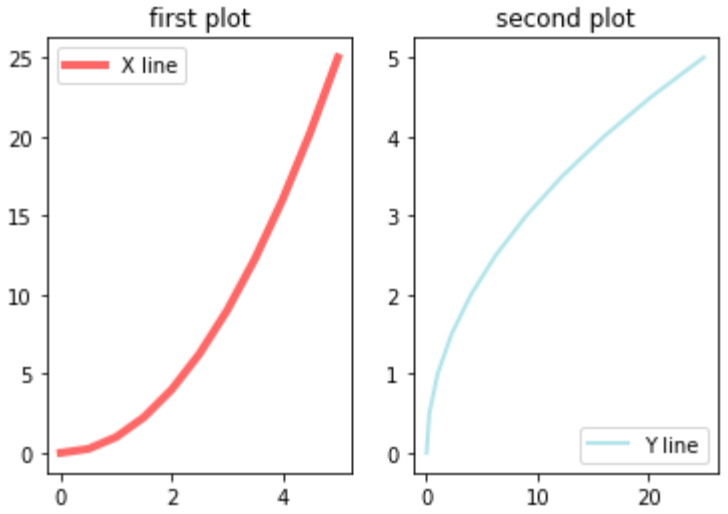
- fig,axes = plt.subplots(nrows=1,ncols=2)
- axes[0].plot(x,y, label='X line', color='red',linewidth=4)
- axes[0].set_title('first plot')
- axes[0].legend(loc=0)



- axes[1].plot(y,x, label='Y line', color='#00AABB', linewidth=2)
- axes[1].set_title('second plot')
- axes[1].legend(loc=4)

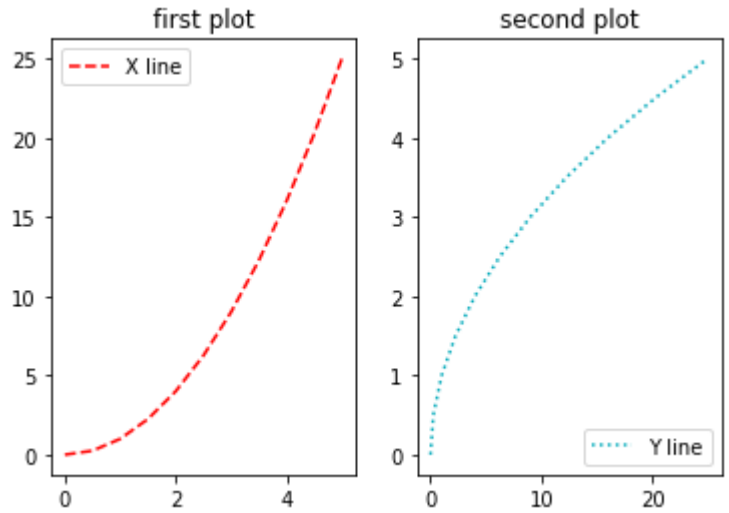
- fig,axes = plt.subplots(nrows=1,ncols=2)
- axes[0].plot(x,y, label='X line', color='red',linewidth=4, alpha=0.6)
- axes[0].set_title('first plot')
- axes[0].legend(loc=0)

- axes[1].plot(y,x, label='Y line', color='#00AABB', linewidth=2, alpha=0.3)
- axes[1].set_title('second plot')
- axes[1].legend(loc=4)



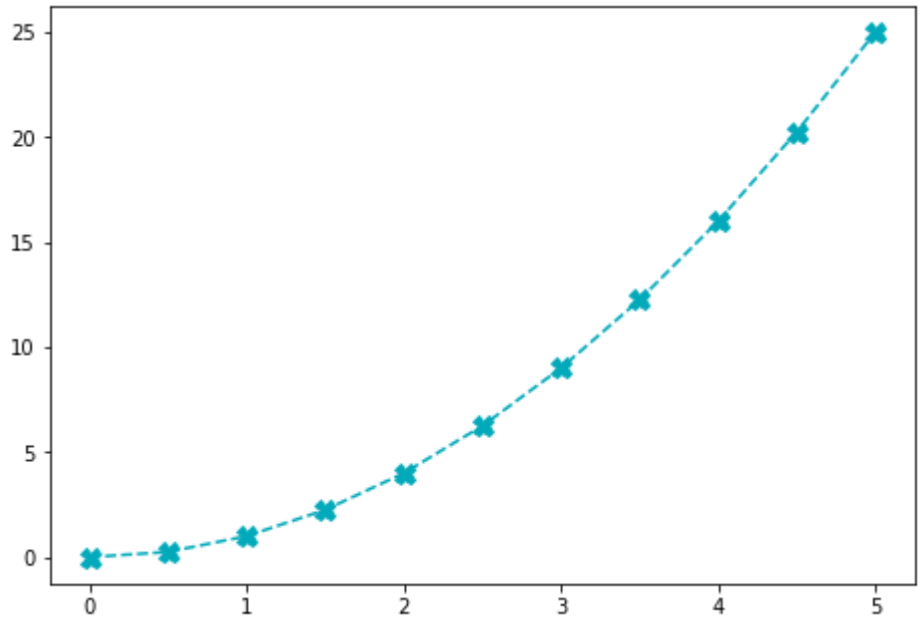
- fig,axes = plt.subplots(nrows=1,ncols=2)
- axes[0].plot(x,y, label='X line', color='red', linestyle='--')
- axes[0].set_title('first plot')
- axes[0].legend(loc=0)

- axes[1].plot(y,x, label='Y line', color='#00AABB’, ls='dotted')
- axes[1].set_title('second plot')
- axes[1].legend(loc=4)

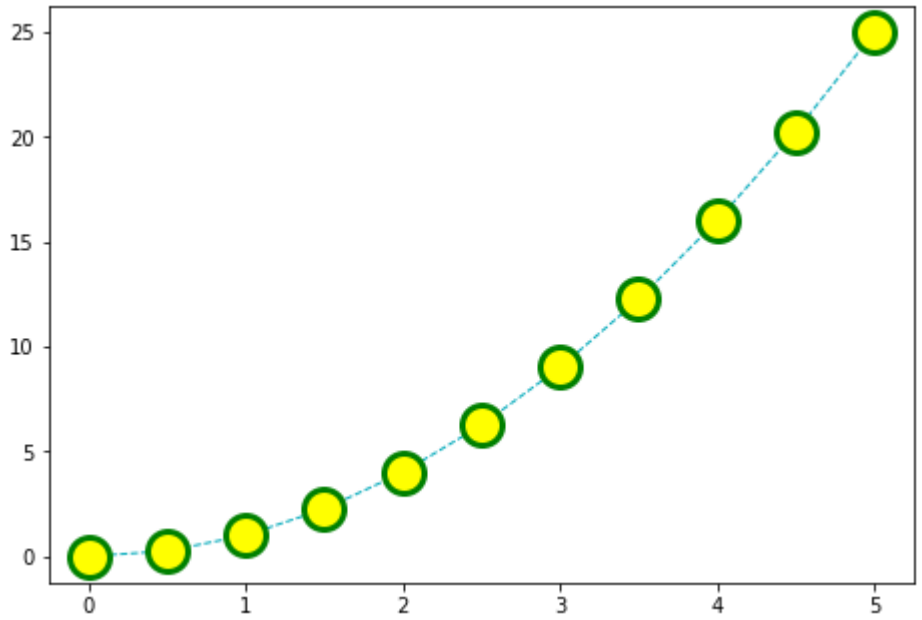


Marker attribute

- fig = plt.figure()
- ax=fig.add_axes([0,0,1,1])
- ax.plot(x,y, ls='dashed' , label='Y line', color='#00AABB', marker='X',markersize=10)



```
fig = plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y, ls='dashed' , label='Y line',lw=1 ,color='#00AABB',
marker='o',
        markersize=20, markerfacecolor='yellow', markeredgewidth=3,
markeredgecolor='green')
```



Scatter

```
fig = plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(x,y)
```