8 - Matplotlib

Data Science Practicum 2021/22, Lesson 8

Marko Tkalčič

Univerza na Primorskem

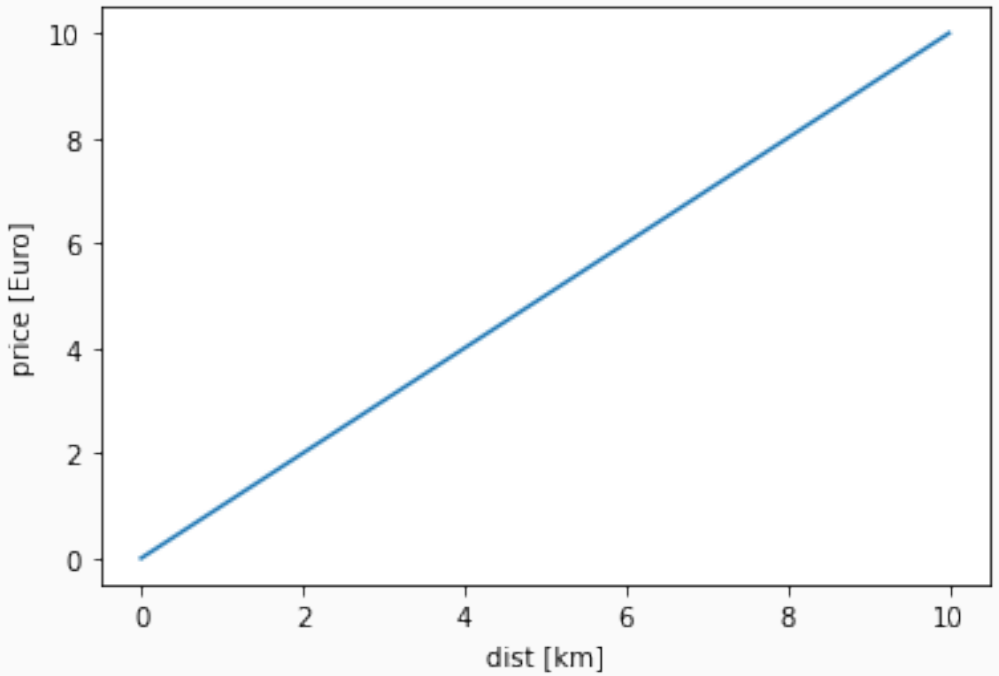
Table of Contents

Matplotlib

- Most used package for 2D plotting
- Produces publication quality figures
- Matlab-like plotting framework

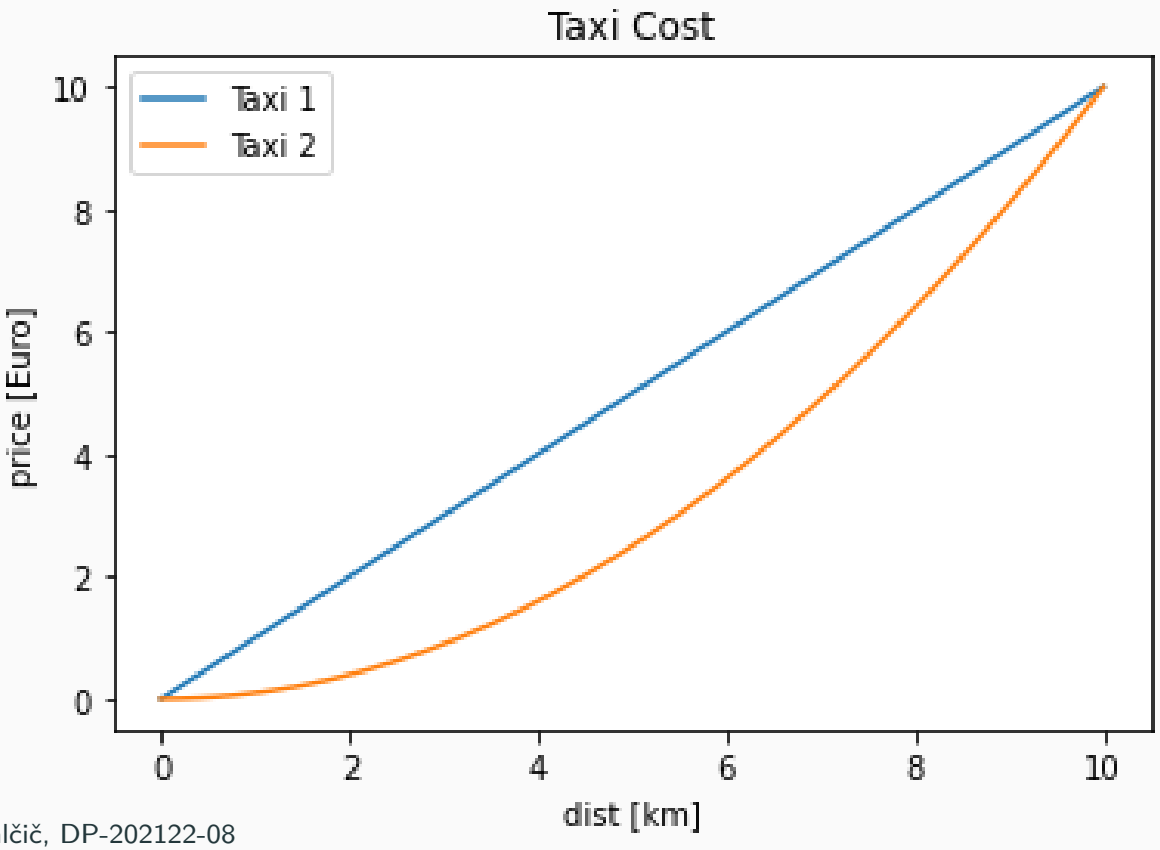
Start

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

dist = np.linspace(0, 10, 100)

taxi_1 = dist.copy()
plt.plot(taxi_1, dist)

plt.xlabel("dist [km]")
plt.ylabel("price [Euro]")
plt.show()    # invokes the Plot viewer window
```



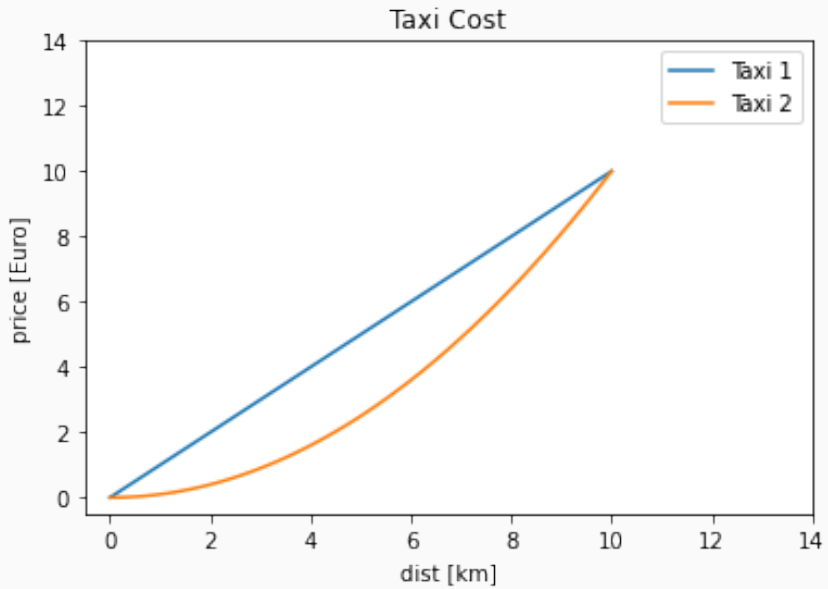
Start

```
taxi_2 = (dist ** 2) / 10
plt.plot(dist, taxi_1)
plt.plot(dist, taxi_2)

plt.legend(["Taxi 1","Taxi 2"])
plt.xlabel("dist [km]")
plt.ylabel("price [Euro]")
plt.title("Taxi Cost")

plt.show()
```



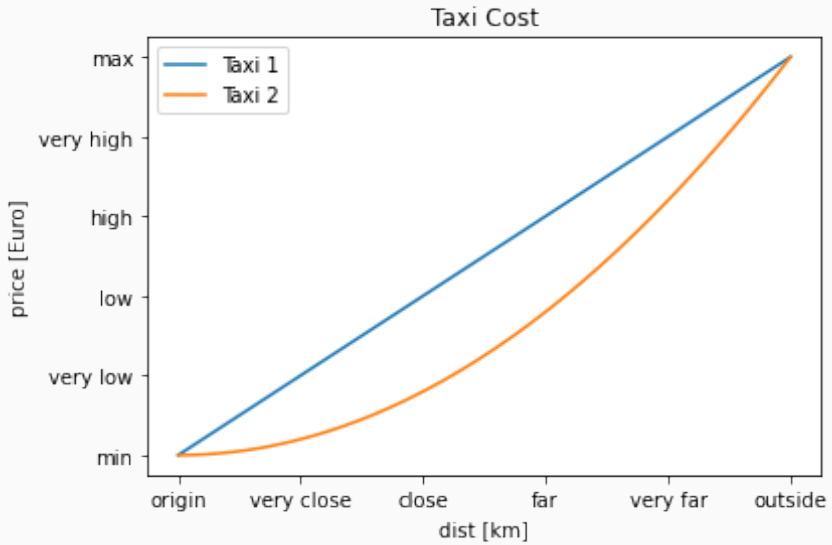
x/y-ticks

```
taxi_2 = (dist ** 2) / 10
plt.plot(dist, taxi_1)
plt.plot(dist, taxi_2)

plt.legend(["Taxi 1","Taxi 2"])
plt.xlabel("dist [km]")
plt.ylabel("price [Euro]")
plt.title("Taxi Cost")
plt.xticks(np.arange(0, 16, step=2))
plt.yticks(np.arange(0, 16, step=2))
plt.show()
```



x/y-ticks

```
taxi_2 = (dist ** 2) / 10
plt.plot(dist, taxi_1)
plt.plot(dist, taxi_2)

plt.legend(["Taxi 1","Taxi 2"])
plt.xlabel("dist [km]")
plt.ylabel("price [Euro]")
plt.title("Taxi Cost")
plt.xticks(np.arange(12,step=2), ('origin','very close', 'close',  'far', 'very far','outside'))
plt.yticks(np.arange(12,step=2), ('min','very low', 'low', 'high', 'very high','max'))

plt.show()
```

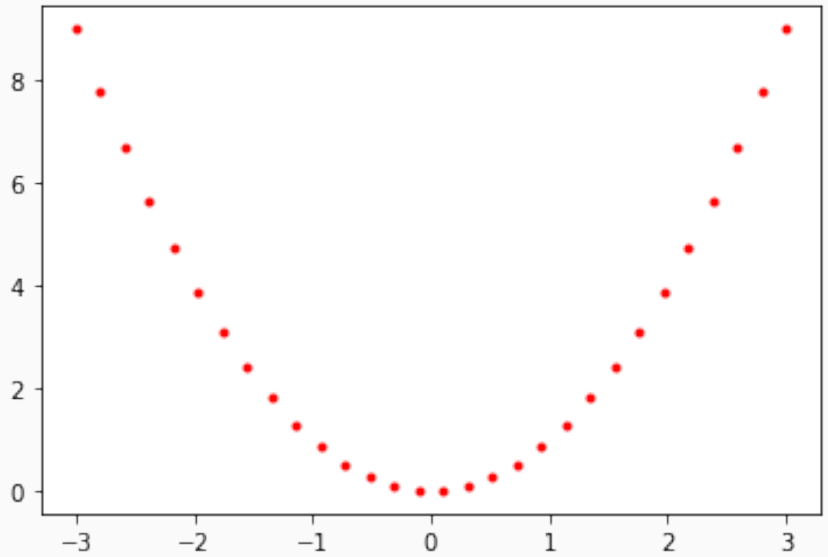


Plotting

- To plot symbols rather than lines, provide an additional string argument.

symbols	- , –, -., , . , , , , o , ^ , v , < , > , s , + , x , D , d , 1 , 2 , 3 , 4 , h , H , p , | , _
colors	b, g, r, c, m, y, k, w

```
x = linspace(-3, 3, 30)
y = x**2
plt.plot(x, y, 'r.')
plt.show()
```



Object-oriented Interface

- Idea: create figure objects and then just call methods or attributes off of that object
  - create new figure
  - create plots through the figure reference

Object-oriented Interface

- Idea: create figure objects and then just call methods or attributes off of that object
  - create new figure
  - create plots through the figure reference

- Create an empty canvas

```
fig = plt.figure()
```

Object-oriented Interface

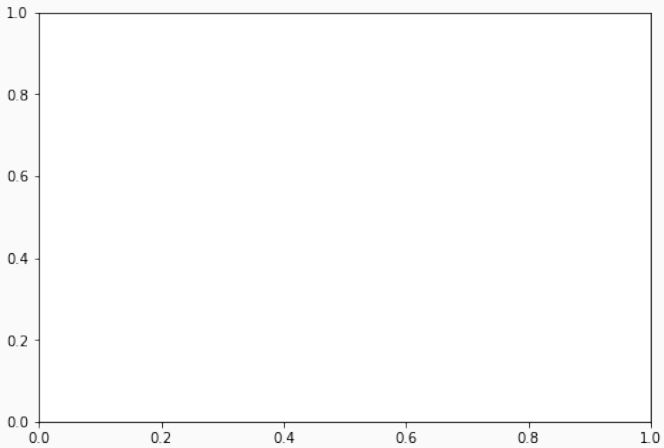
- Idea: create figure objects and then just call methods or attributes off of that object
  - create new figure
  - create plots through the figure reference

- Create an empty canvas

```
fig = plt.figure()
```

- Add axes to figure: a list object of 4 elements corresponding to left, bottom, width and height of the figure. Each number must be between 0 and 1

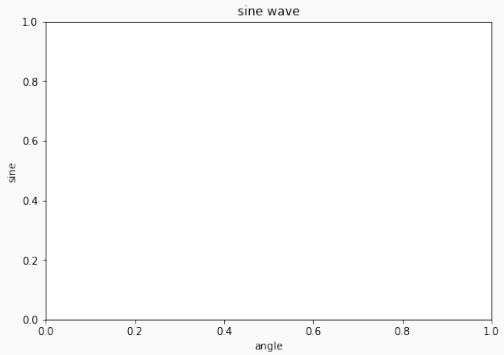
```
ax=fig.add_axes([0,0,1,1])
```



Object-oriented Interface

- Set labels for x and y axis as well as title

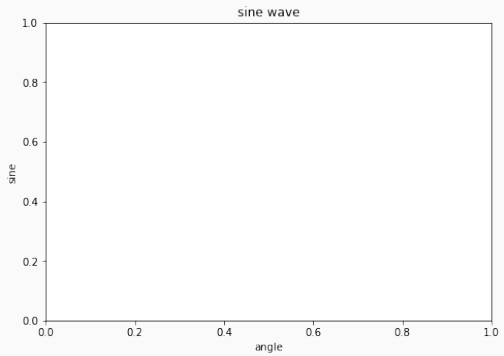
```
ax.set_title("sine wave")
ax.set_xlabel('angle')
ax.set_ylabel('sine')
```



Object-oriented Interface

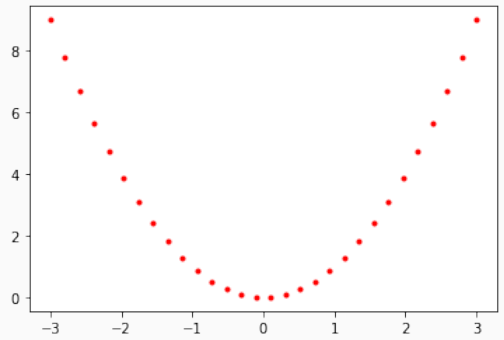
- Set labels for x and y axis as well as title

```
ax.set_title("sine wave")
ax.set_xlabel('angle')
ax.set_ylabel('sine')
```



- invoke the plot() method

```
ax.plot(x,y)
```



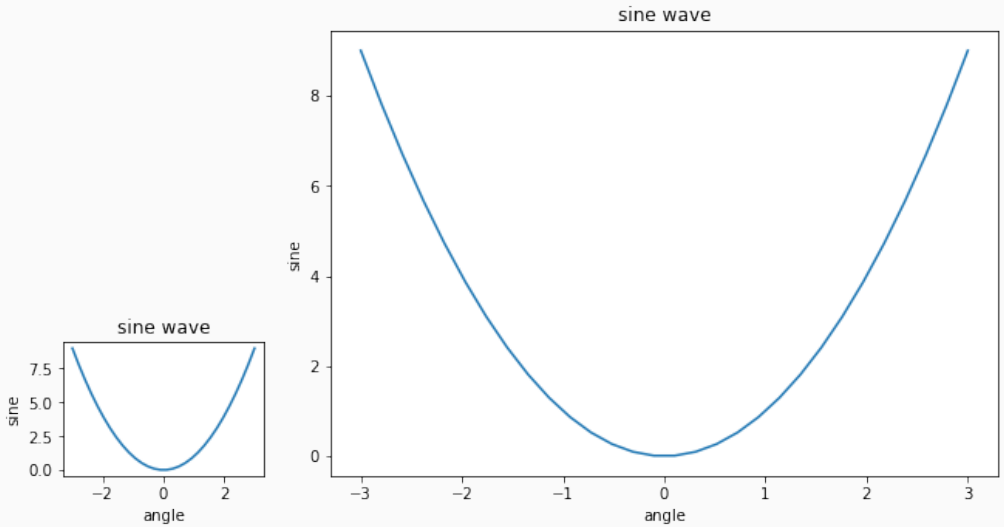
The Figure Class

- Additional figure properties:
  - Figsize (width,height) tuple in inches
  - Dpi Dots per inches
  - Facecolor Figure patch facecolor
  - Edgecolor Figure patch edge color
  - Linewidth Edge line width

The Axes Class

- A given figure can contain many Axes, but a given Axes object can only be in one Figure.
- The Axes contains two (or three in the case of 3D) **Axis** objects.

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

fig = plt.figure()
x = np.linspace(-3, 3, 30)
y = x**2

ax=fig.add_axes([0,0,0.3,0.3])
ax.set_title("sine wave")
ax.set_xlabel('angle')
ax.set_ylabel('sine')
ax.plot(x,y)

ax2=fig.add_axes([0.4,0,1,1])
ax2.set_title("sine wave")
ax2.set_xlabel('angle')
ax2.set_ylabel('sine')
ax2.plot(x,y)
```



Subplots

- subplot() returns the axes object at a given grid position

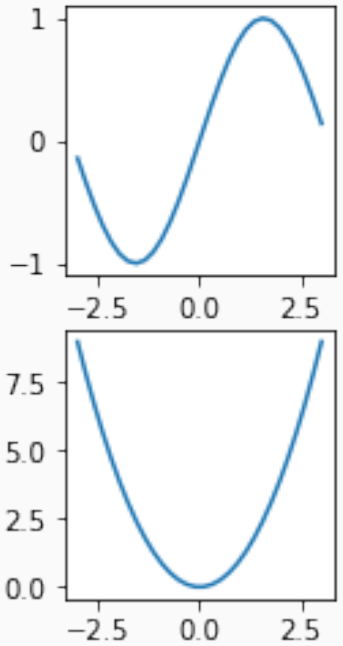
```
plt.subplot(subplot(nrows, ncols, index)
```

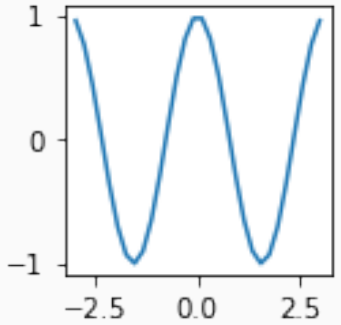
Subplots

- subplot() returns the axes object at a given grid position

```
plt.subplot(subplot(nrows, ncols, index)
```

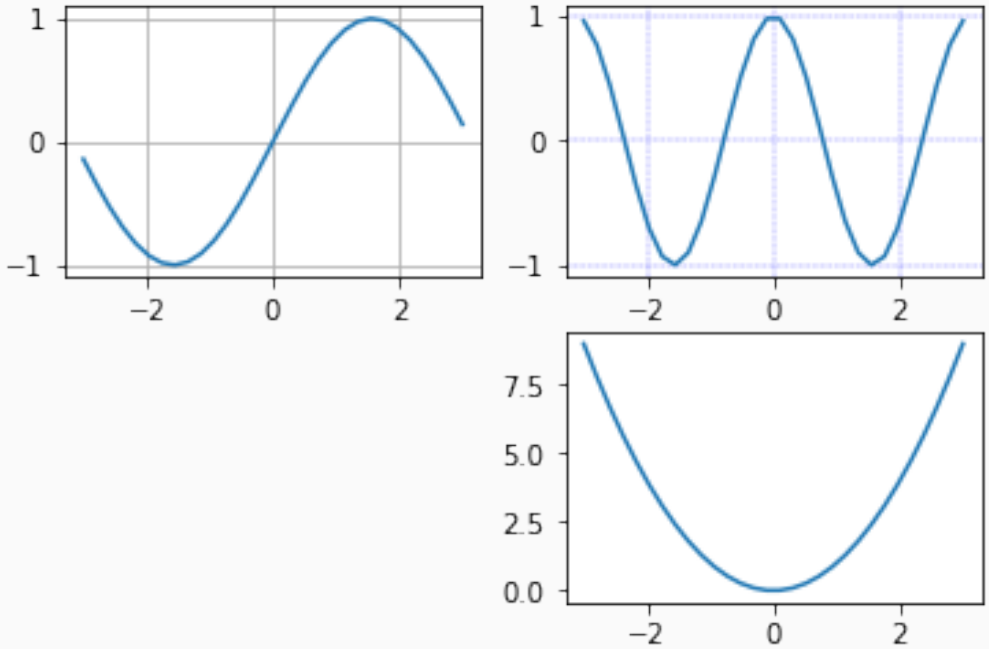
```
x = np.linspace(-3, 3, 30)
xa = np.sin(x)
xb = np.cos(2*x)
xc = x **2

a = plt.subplot(2,3,1)
b = plt.subplot(2,3,3)
c = plt.subplot(2,3,4)

a.plot(x,xa)
b.plot(x,xb)
c.plot(x,xc)
```





Grid

```
x = np.linspace(-3, 3, 30)
xa = np.sin(x)
xb = np.cos(2*x)
xc = x **2

a = plt.subplot(2,2,1)
b = plt.subplot(2,2,2)
c = plt.subplot(2,2,4)

a.plot(x,xa)
a.grid(True)
b.plot(x,xb)
b.grid(color='b', ls = '-.', lw = 0.25)
c.plot(x,xc)
```

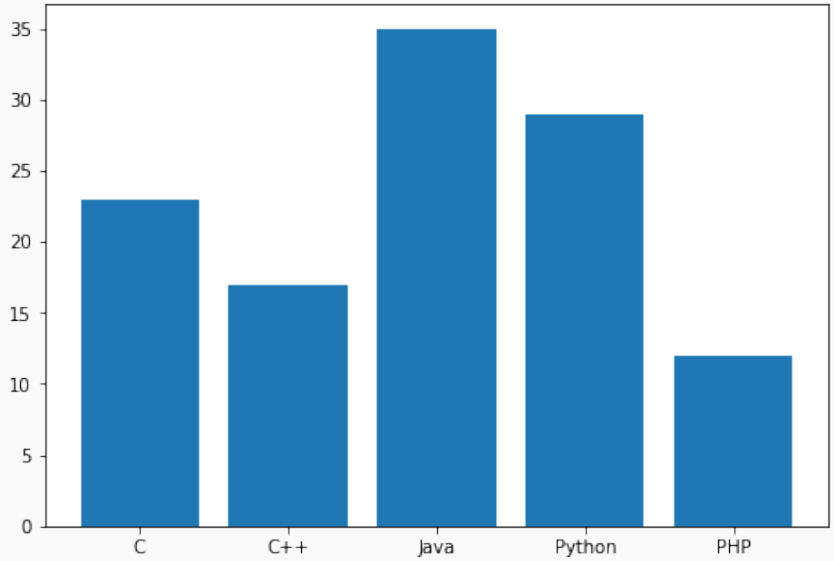


Saving

```
x = np.linspace(-3, 3, 30)
xa = np.sin(x)
xb = np.cos(2*x)
xc = x **2

fig1=plt.figure()
fig2=plt.figure()

a1 = fig1.add_axes([0,0,1,1])
a2 = fig2.add_axes([0,0,1,1])

a1.plot(x,xa)
a2.plot(x,xb)

fig1.savefig("out1.jpg", dpi=200)
fig2.savefig("out2.jpg", dpi=200)
plt.savefig("out_plt.jpg")        # saves last used figure
```

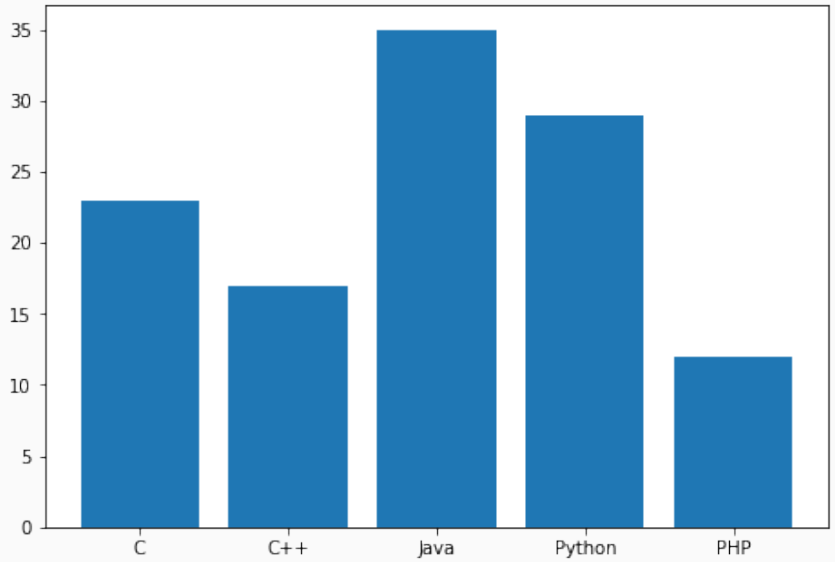
Bar Chart

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```

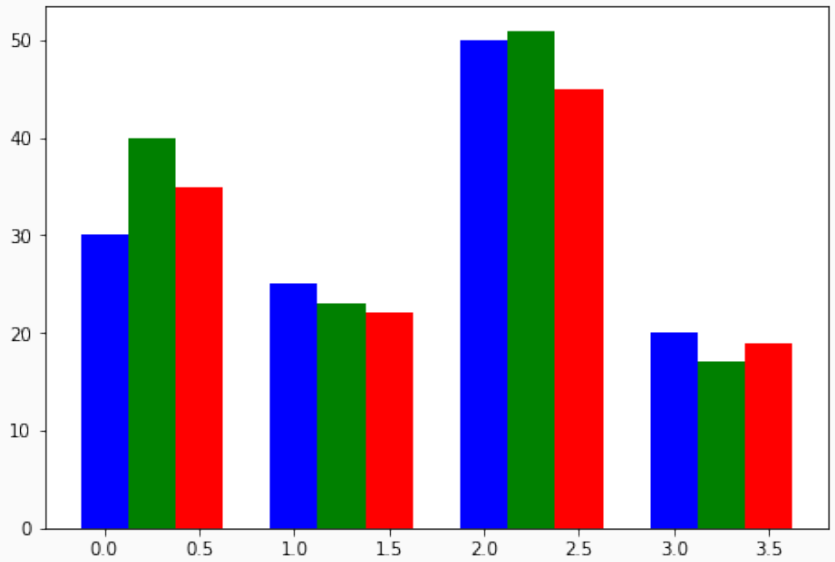


Bar Chart

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```

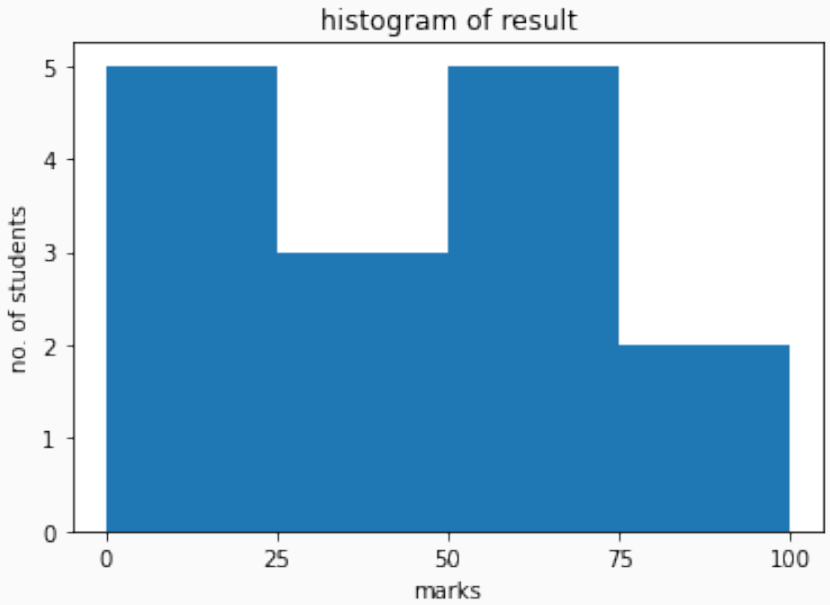



```
import numpy as np
import matplotlib.pyplot as plt
data = [[30, 25, 50, 20],
[40, 23, 51, 17],
[35, 22, 45, 19]]
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, data[0], color = 'b', width = 0.25)
ax.bar(X + 0.25, data[1], color = 'g', width = 0.25)
ax.bar(X + 0.50, data[2], color = 'r', width = 0.25)
```



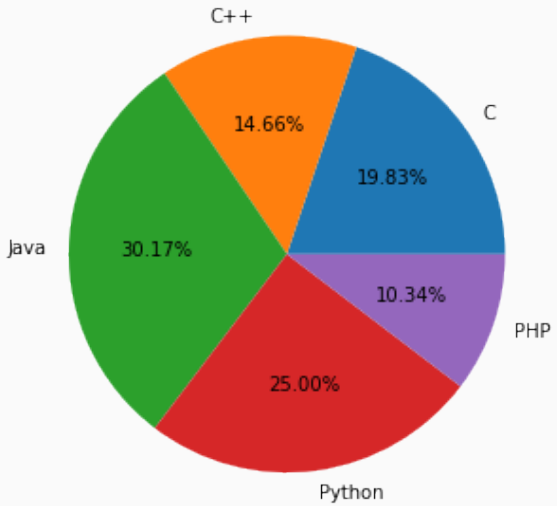
Histogram

```
from matplotlib import pyplot as plt
import numpy as np
fig,ax = plt.subplots(1,1)
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
ax.hist(a, bins = [0,25,50,75,100])
ax.set_title("histogram of result")
ax.set_xticks([0,25,50,75,100])
ax.set_xlabel('marks')
ax.set_ylabel('no. of students')
plt.show()
```



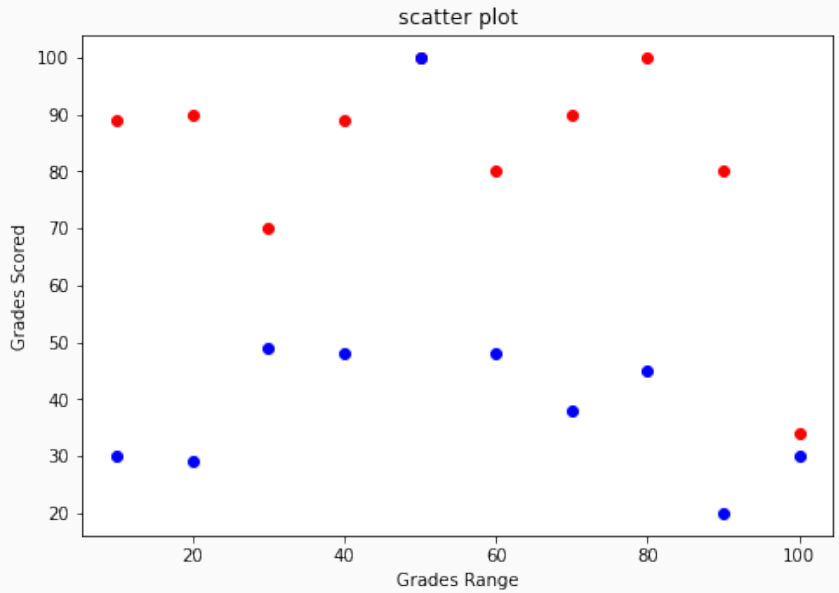
Pie Chart

```
from matplotlib import pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.pie(students, labels = langs,autopct='%1.2f%%')
plt.show()
```



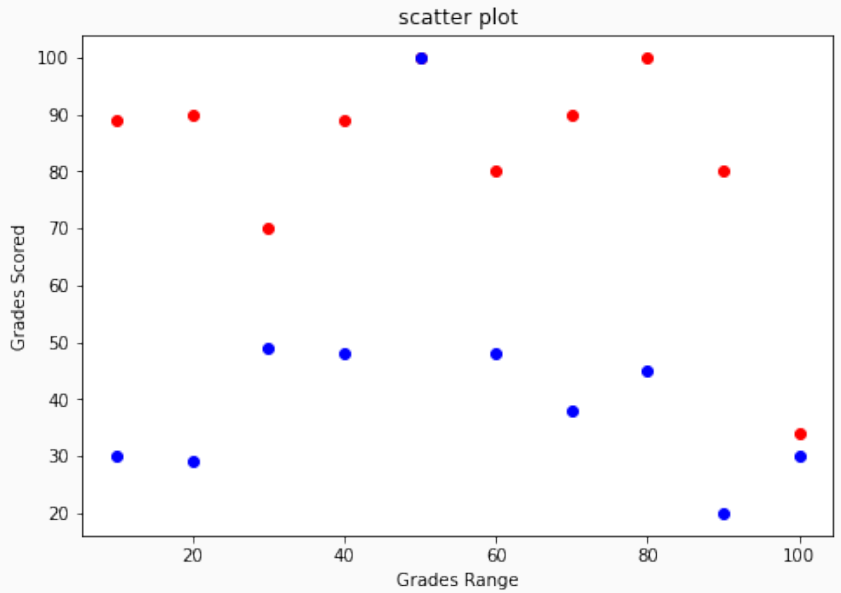
Scatter Plot

```
import matplotlib.pyplot as plt
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(grades_range, girls_grades, color='r')
ax.scatter(grades_range, boys_grades, color='b')
ax.set_xlabel('Grades Range')
ax.set_ylabel('Grades Scored')
ax.set_title('scatter plot')
plt.show()
```

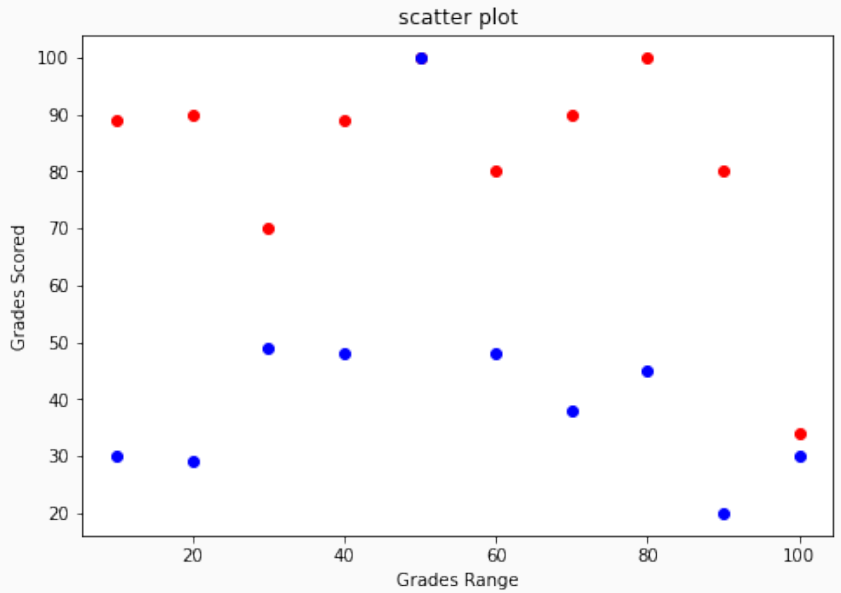


Scatter Plot

```
import matplotlib.pyplot as plt
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(grades_range, girls_grades, color='r')
ax.scatter(grades_range, boys_grades, color='b')
ax.set_xlabel('Grades Range')
ax.set_ylabel('Grades Scored')
ax.set_title('scatter plot')
plt.show()
```

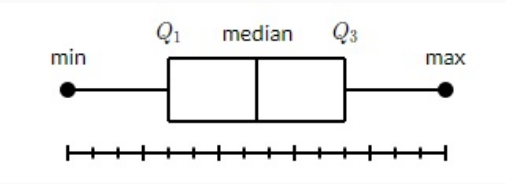



```
import matplotlib.pyplot as plt
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(grades_range, girls_grades,'ro')
ax.plot(grades_range, boys_grades, 'bo')
ax.set_xlabel('Grades Range')
ax.set_ylabel('Grades Scored')
ax.set_title('scatter plot')
plt.show()
```

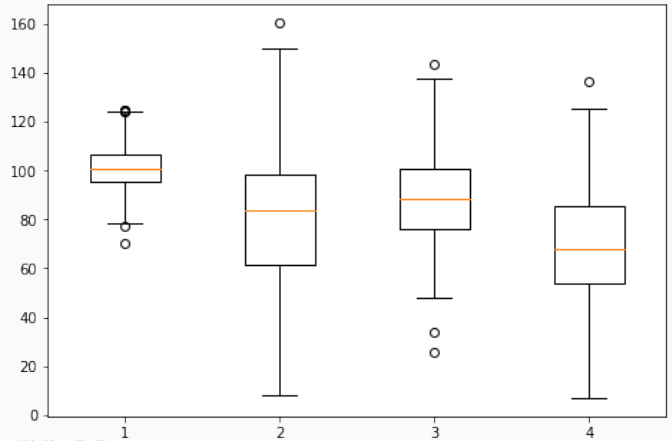


Boxplot (Whisker plot)

- minimum, first quartile, median, third quartile, and maximum




```
np.random.seed(10)
collectn_1 = np.random.normal(100, 10, 200)
collectn_2 = np.random.normal(80, 30, 200)
collectn_3 = np.random.normal(90, 20, 200)
collectn_4 = np.random.normal(70, 25, 200)

fig = plt.figure()
# Create an axes instance
ax = fig.add_axes([0,0,1,1])
# Create the boxplot
bp = ax.boxplot([collectn_1, collectn_2, collectn_3, collectn_4])
plt.show()
```



Violin plot

- Similar to box plots, except that they also show the probability density of the data at different values.

```
np.random.seed(10)
collectn_1 = np.random.normal(100, 10, 200)
collectn_2 = np.random.normal(80, 30, 200)
collectn_3 = np.random.normal(90, 20, 200)
collectn_4 = np.random.normal(70, 25, 200)

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
bp = ax.violinplot([collectn_1, collectn_2, collectn_3, collectn_4])
plt.show()
```

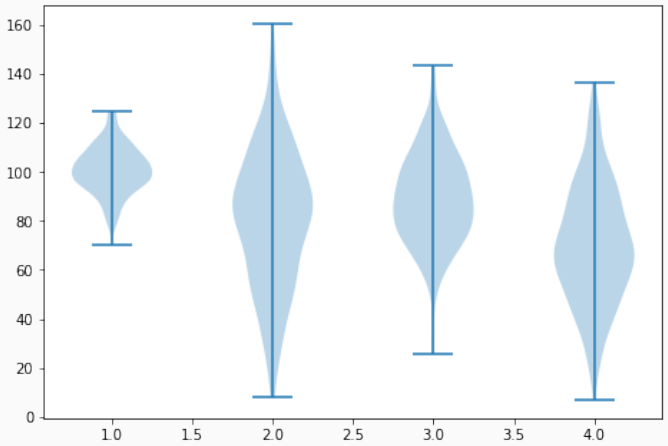


Table of Contents

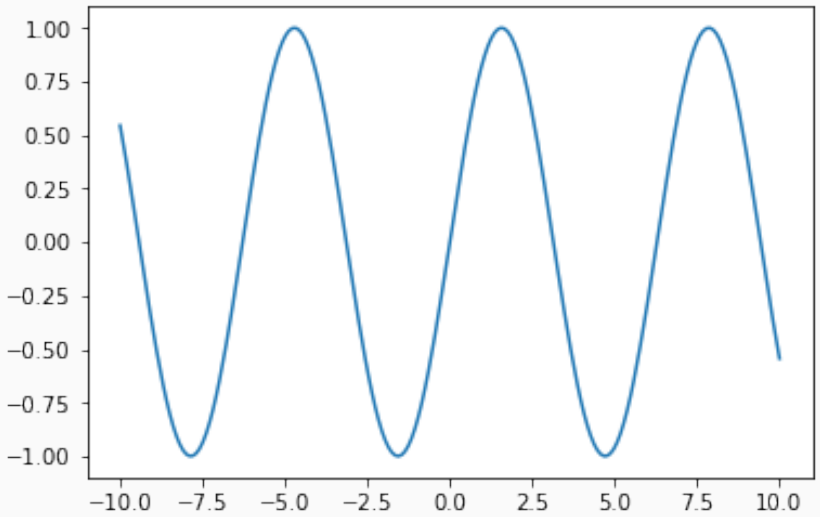
Exercise 1

- create the `x` variable with 200 elements from -10 to 10
- create the `y` variable which has the sine (`np.sin()`) of `x`
- plot the curve

Exercise 1

- create the x variable with 200 elements from -10 to 10
- create the y variable which has the sine (`np.sin()`) of x
- plot the curve

```
from matplotlib import pyplot as plt
import numpy as np
x = np.linspace(-10,10,200)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```



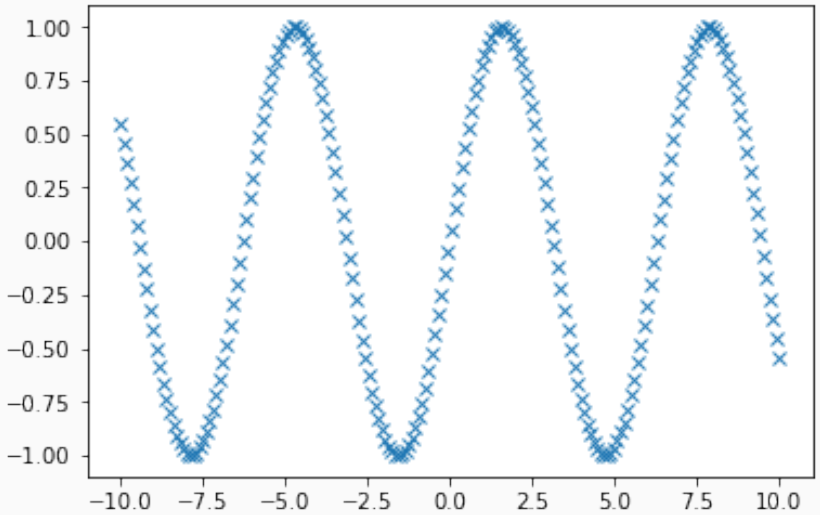
Exercise 2

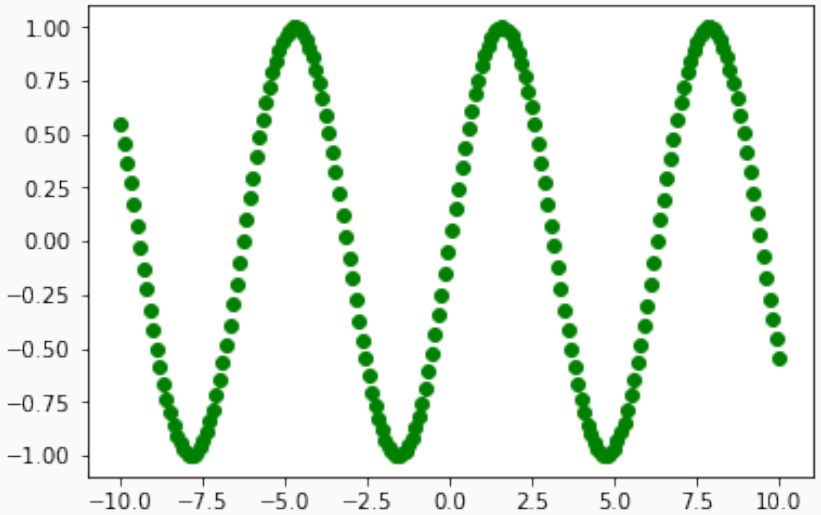
- plot the elements of x (above exercise):
  - with an x
  - with a green o

Exercise 2

- plot the elements of x (above exercise):
  - with an x
  - with a green o

```
plt.plot(x,y,'x')
plt.plot(x,y,'og')
```





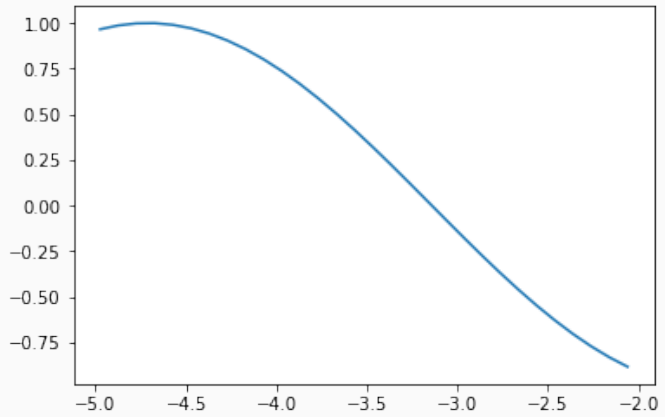
Exercise 3

- plot only the elements from the index 50 to 80

Exercise 3

- plot only the elements from the index 50 to 80

```
plt.plot(x[50:80],y[50:80])
plt.show()
```

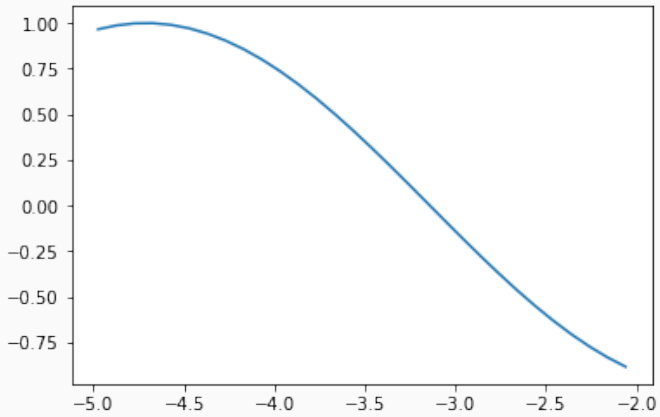


- plot only the elements from the index 51 to 56

Exercise 3

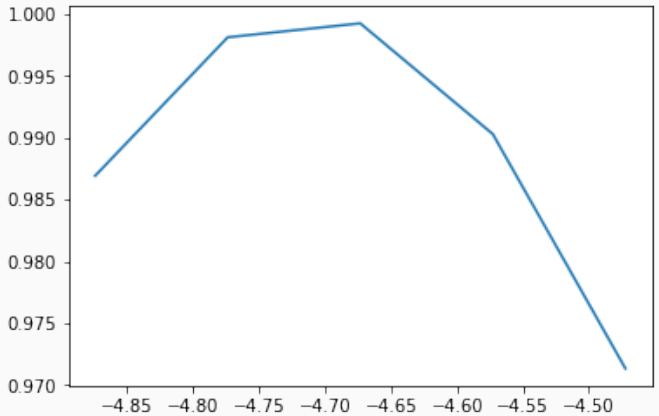
- plot only the elements from the index 50 to 80

```
plt.plot(x[50:80],y[50:80])
plt.show()
```



- plot only the elements from the index 51 to 56

```
plt.plot(x[51:56],y[51:56])
plt.show()
```



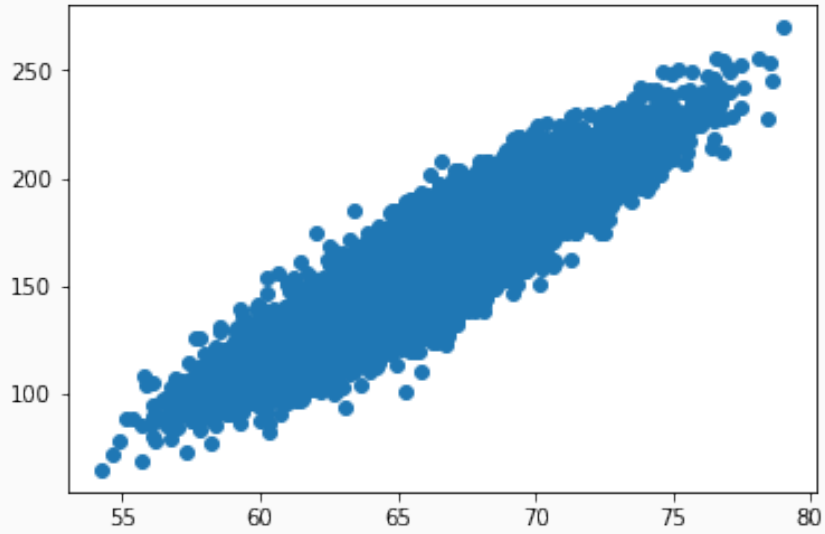
Exercise 4 - Hold

- load the dataset `weight-height.csv`
- use `df[].values` to get a numpy array out of the data-frame columns
- plot the height vs weight

Exercise 4 - Hold

- load the dataset `weight-height.csv`
- use `df[].values` to get a numpy array out of the data-frame columns
- plot the height vs weight

```
import pandas as pd
raw = pd.read_csv("weight-height.csv")
h   = raw["Height"].values
w   = raw["Weight"].values
plt.plot(h,w,'o')
plt.show()
```



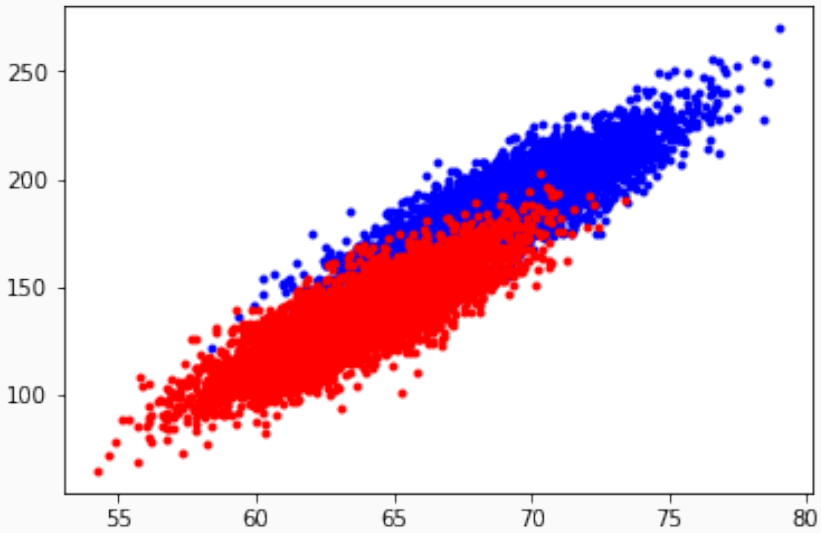
Exercise hold

- plot the male with blue and female with red in the same plot

Exercise hold

- plot the male with blue and female with red in the same plot

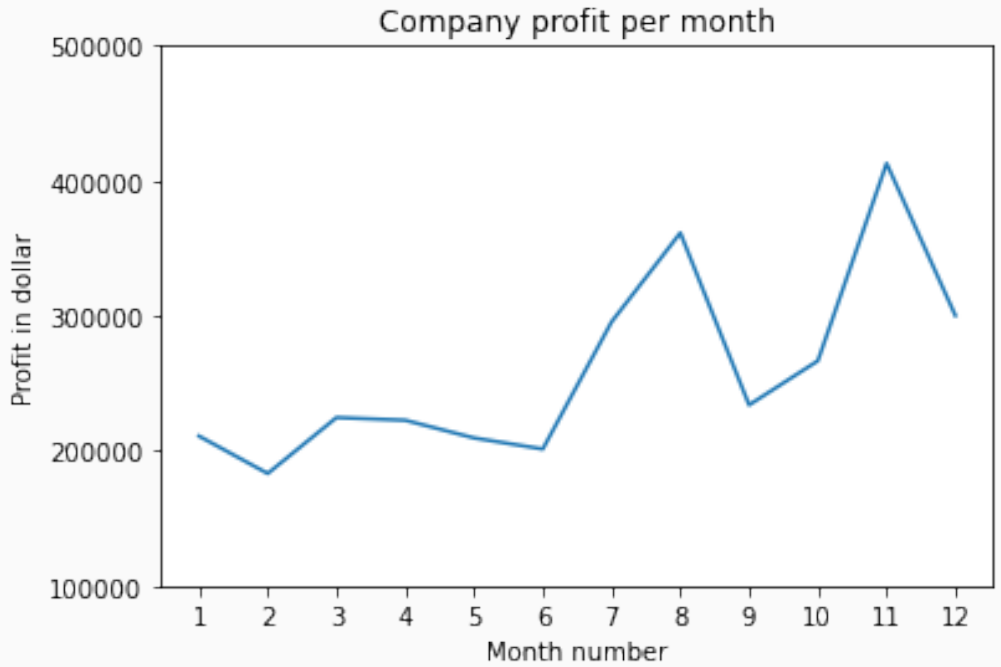
```
i_m = raw["Gender"] == "Male"
plt.plot(h[i_m],w[i_m],'.b')
i_f = raw["Gender"] == "Female"
plt.plot(h[i_f],w[i_f],'.r')
plt.show()
```



Exercise 5

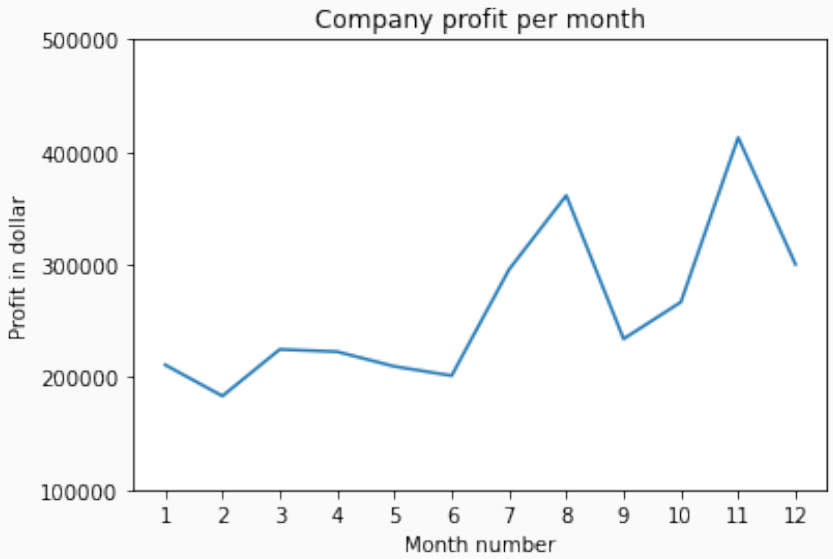
- Read the Total profit of all months and show it using a line plot
- The Total profit data is provided for each month. Generated line plot must include the following properties:
  - X label name = Month Number
  - Y label name = Total profit

The line plot graph should look like this:



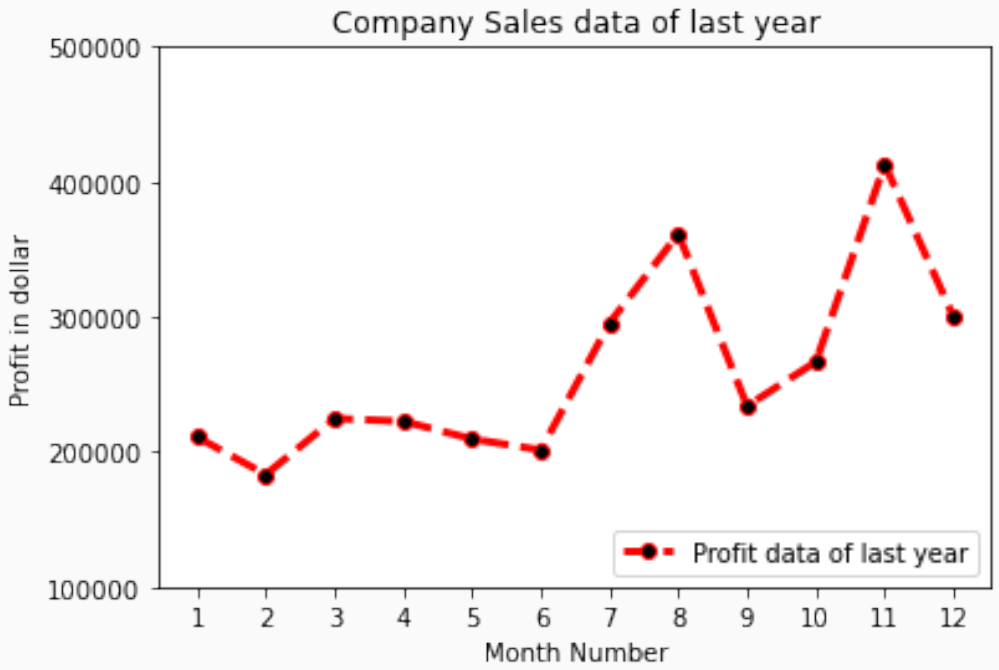
Exercise 5

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
print(df.head())
profitList = df ['total_profit'].values #df ['total_profit'].tolist()
monthList  = df ['month_number'].values #df ['month_number'].tolist()
plt.plot(monthList, profitList, label = 'Month-wise Profit data of last year')
plt.xlabel('Month number')
plt.ylabel('Profit in dollar')
plt.xticks(monthList)
plt.title('Company profit per month')
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



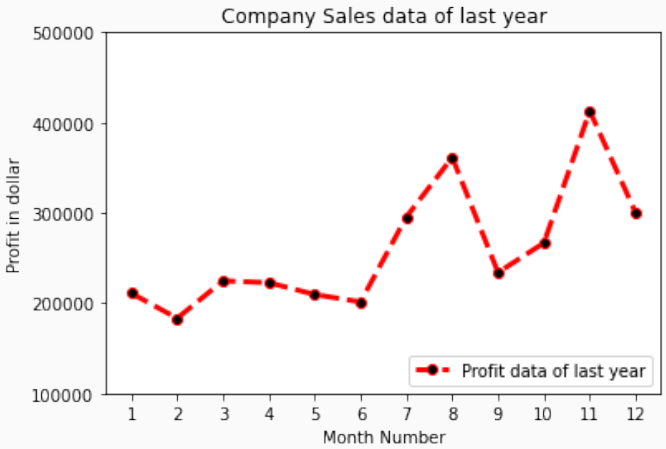
Exercise 6

- Get Total profit of all months and show line plot with the following Style properties
  - Line Style dotted and Line-color should be red
  - Show legend at the lower right location.
  - X label name = Month Number
  - Y label name = Sold units number
  - Add a circle marker.
  - Line marker color as read
  - Line width should be 3



Exercise 6

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
profitList = df ['total_profit'].values
monthList  = df ['month_number'].values

plt.plot(monthList, profitList, label = 'Profit data of last year',
      color='r', marker='o', markerfacecolor='k',
      linestyle='--', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Profit in dollar')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



Exercise 7

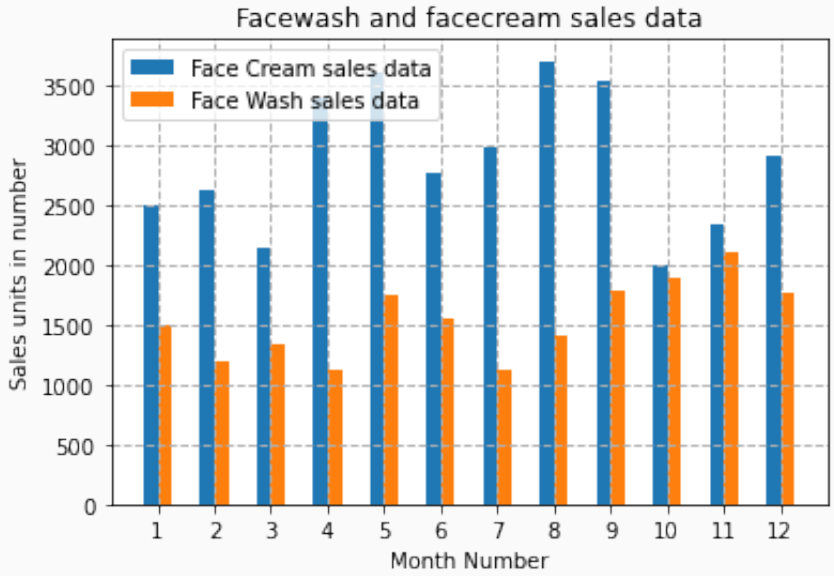
- Read all product sales data and show it using a multiline plot

Exercise 7

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
monthList  = df ['month_number'].values
faceCremSalesData   = df ['facecream'].values
faceWashSalesData   = df ['facewash'].values
toothPasteSalesData = df ['toothpaste'].values
bathingsoapSalesData   = df ['bathingsoap'].values
shampooSalesData   = df ['shampoo'].values
moisturizerSalesData = df ['moisturizer'].values

plt.plot(monthList, faceCremSalesData,   label = 'Face cream Sales Data', marker='o', linewidth=3)
plt.plot(monthList, faceWashSalesData,   label = 'Face Wash Sales Data',  marker='o', linewidth=3)
plt.plot(monthList, toothPasteSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)
plt.plot(monthList, bathingsoapSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)
plt.plot(monthList, shampooSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)
plt.plot(monthList, moisturizerSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.legend(loc='upper left')
plt.xticks(monthList)
plt.yticks([1000, 2000, 4000, 6000, 8000, 10000, 12000, 15000, 18000])
plt.title('Sales data')
plt.show()
```

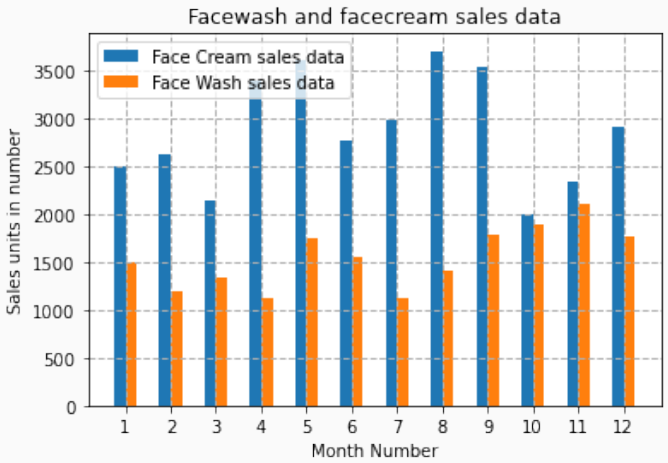
Exercise 8

- Read face cream and facewash product sales data and show it using the bar chart



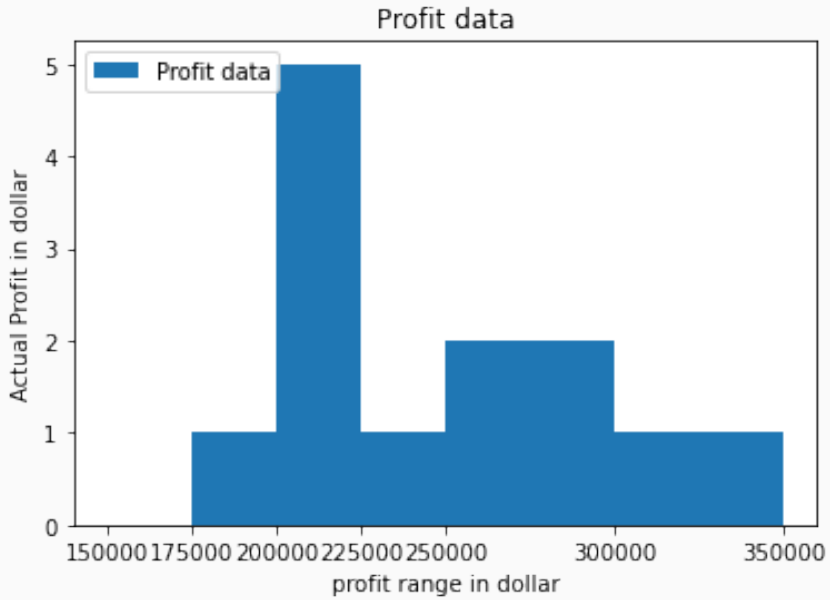
Exercise 8

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
monthList  = df ['month_number'].values
faceCremSalesData   = df ['facecream'].values
faceWashSalesData   = df ['facewash'].values

plt.bar([a-0.25 for a in monthList], faceCremSalesData, width= 0.25, label = 'Face Cream sales data', align='edge')
plt.bar([a+0.25 for a in monthList], faceWashSalesData, width= -0.25, label = 'Face Wash sales data', align='edge')
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.legend(loc='upper left')
plt.title(' Sales data')

plt.xticks(monthList)
plt.grid(True, linewidth= 1, linestyle="--")
plt.title('Facewash and facecream sales data')
plt.show()
```



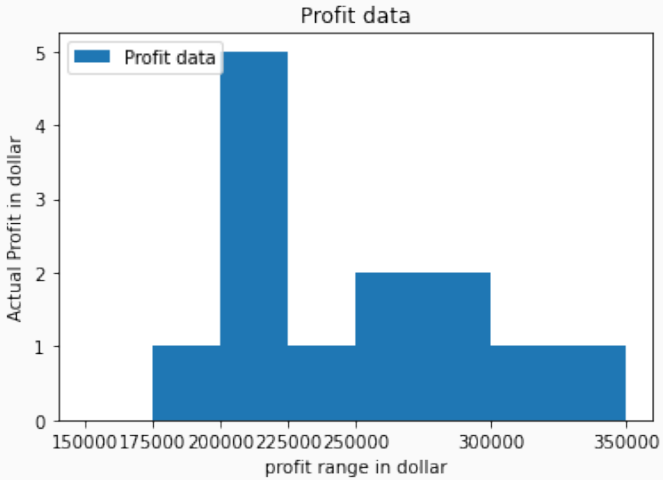
Exercise 9

- Read the total profit of each month and show it using the histogram to see most common profit ranges
- Save it to the disk



Exercise 9

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
profitList = df ['total_profit'].values
labels = ['low', 'average', 'Good', 'Best']
profit_range = [150000, 175000, 200000, 225000, 250000, 300000, 350000]
plt.hist(profitList, profit_range, label = 'Profit data')
plt.xlabel('profit range in dollar')
plt.ylabel('Actual Profit in dollar')
plt.legend(loc='upper left')
plt.xticks(profit_range)
plt.title('Profit data')
plt.show()
plt.savefig('sales_data_of_bathingsoap.png', dpi=150)
```



Exercise 10

- Read Bathing soap facewash of all months and display it using the Subplot

Exercise 10

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("company_sales_data.csv")
monthList  = df ['month_number'].values
bathingsoap   = df ['bathingsoap'].values
faceWashSalesData   = df ['facewash'].values

f, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(monthList, bathingsoap, label = 'Bathingsoap Sales Data', color='k', marker='o', linewidth=3)
axarr[0].set_title('Sales data of  a Bathingsoap')
axarr[1].plot(monthList, faceWashSalesData, label = 'Face Wash Sales Data', color='r', marker='o', linewidth=3)
axarr[1].set_title('Sales data of  a facewash')

plt.xticks(monthList)
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.show()
```

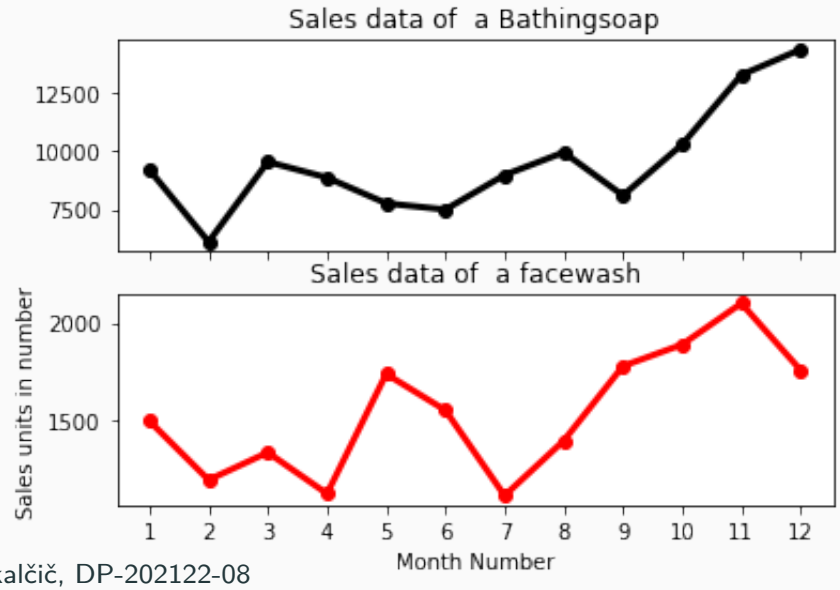


Table of Contents

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://pynative.com/python-matplotlib-exercise/
- https://www.tutorialspoint.com/matplotlib/index.htm