ÖZYEĞİN UNIVERSITY

EE393 Python for Engineers

Dr. Orhan Gökçöl

orhan.gokcol@ozyegin.edu.tr

online

07.12.2020

2020-2021 Fall Semester

1

ICE BREAKER!!!!

Reading a web resource using **urllib**

URL ➔ Uniform Resource Locator

Urllib is a package that collects several modules for working with URLs, such as:

- ✓ **urllib.request** for opening and reading.
- ✓ urllib.parse for parsing URLs into parts
- ✓ urllib.error for the exceptions raised
- ✓ urllib.robotparser for parsing robot.txt files

```
#-------
#urllib
```

```
import urllib.request
request_url = urllib.request.urlopen('https://www.ozyegin.edu.tr/')
print(request_url.read())
```

```
b'<!DOCTYPE html>\n<html lang="tr">\n<head>\n\n  <meta charset="utf-8" /
<meta about="/tr/taxonomy/term/127" typeof="skos:Concept" property="rdfs
el skos:prefLabel" content="Anasayfa" />\n<link rel="shortcut icon" href
tps://www.ozyegin.edu.tr/sites/default/files/96.png" type="image/png" />
```

url.ipynb

2

ICE BREAKER!!!!

Creating user interactions in Jupyter using interact

Open it using Jupyter Notebook. **Do not use Jupyter Lab**

```
: def f(x):
      return x

: interact(f, x=10);
```

x 10

10

```
: interact(f, x=True);
```

x

interaction.ipynb

3

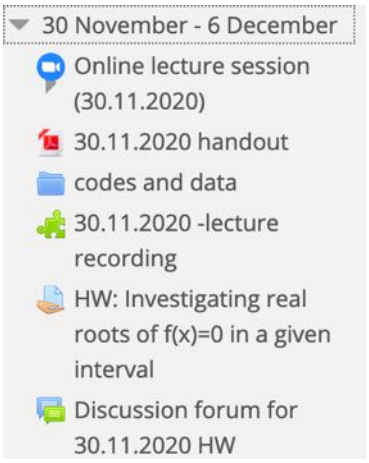
Agenda

- Matplotlib – figure management
- Working with different plot types
- Working with fundamental graphics objects

(Download to your local)

4

07.12.2020 | LMS resources



5

Learning objectives for 07.12.2020

- Knows how to generate (x,y), histogram, pie, bar, scatter and line plots
- Knows how to manage multi-plots using matplotlib

6

MIDTERM EXAM

DECEMBER 21, 2020; 08:40

Online
(Details will be announced later)

FINAL EXAM

JANUARY 15, 2021; 09:00

Online
(Details will be announced later)

7

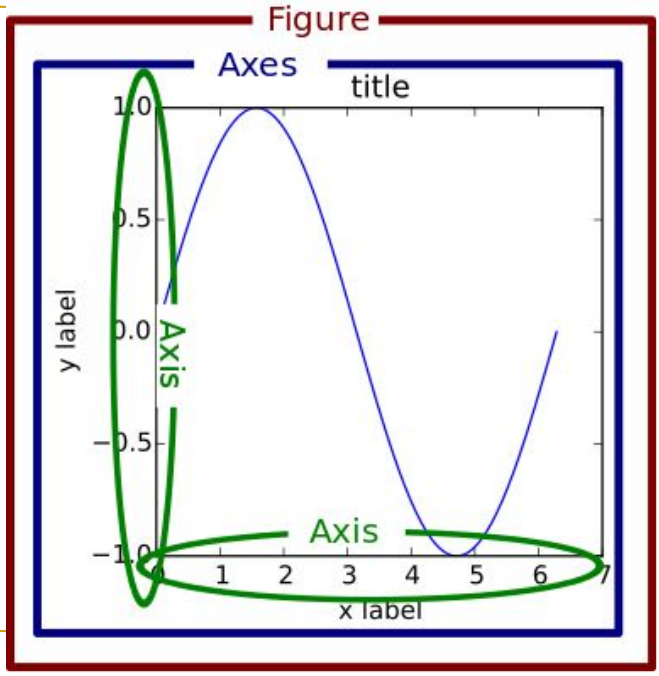
Matplotlib

- We'll first investigate an important python library: **matplotlib**

figure_v2.ipynb

8

What is a 2D Plot?



9

Matplotlib plots

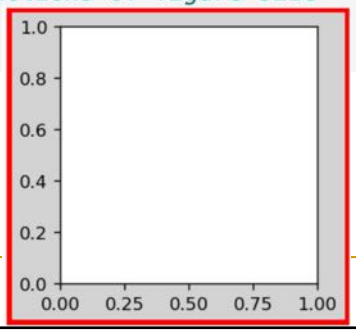
- Matplotlib has two interfaces. The first is an object-oriented (OO) interface. In this case, we utilize an instance of **axes.Axes()** in order to render visualizations on an instance of **figure.Figure()**.
- The second is based on MATLAB and uses a state-based interface. This is encapsulated in the **pyplot** module. (last week)
- The Figure is the final image that may contain one or more Axes.
- The Axes represent an individual plot (don't confuse this with the word "axis", which refers to the x/y axis of a plot).
- We call methods that do the plotting directly from the Axes, which gives us much more flexibility and power in customizing our plot.

10

BASIC SETTING OF THE PLOT AREA

```
#matplotlib figure
import matplotlib.pyplot as plt

#create figure object with given properties
#figure size is in inches, given as tuble
fig = plt.figure(figsize=(2,2), dpi=120,
                 facecolor="lightgrey",
                 edgecolor="red", linewidth=5)
#fig is like an empty sheet of paper on which we plot!
#add axis set : x-y. Name it
#A 4-length sequence of [left, bottom, width, height] quantities
#all units are given as fractions of figure size
ax=fig.add_axes([0,0,1,1])
plt.show()
```

figure_v2.ipynb

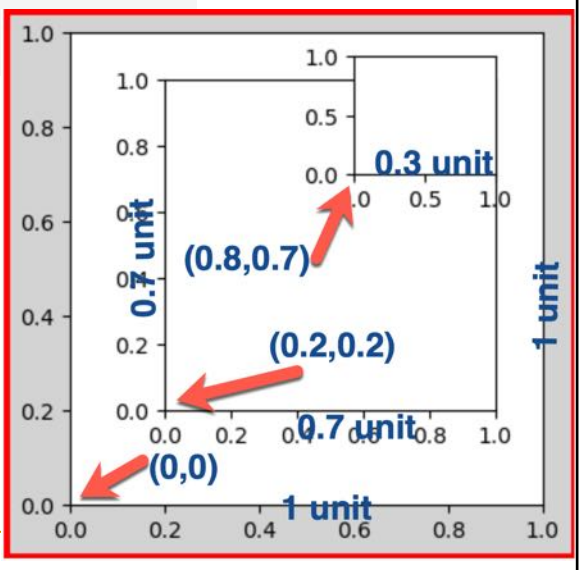


11

```
#if you would like, you can add more than one axis set.
fig = plt.figure(figsize=(3,3), dpi=120,
                 facecolor="lightgrey",
                 edgecolor="red", linewidth=5)
ax0=fig.add_axes([0,0,1,1])
ax1=fig.add_axes([0.2,0.2,0.7,0.7])
ax2=fig.add_axes([0.6,0.7,0.3,0.25])
plt.show()
```

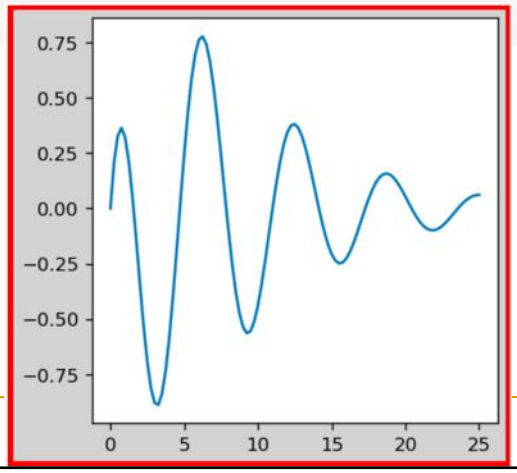
More than one axes

figure_v2.ipynb



12

```
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure(figsize=(3,3), dpi=120,
                 facecolor="lightgrey", edgecolor="red", linewidth=5)
ax=fig.add_axes([0,0,1,1])
#---produce some data
x = np.arange(0,25.01,0.25)
y =x*(np.exp(-0.175*x)*np.cos(x))/np.sqrt(x+1)
ax.plot(x,y)
plt.show()
```

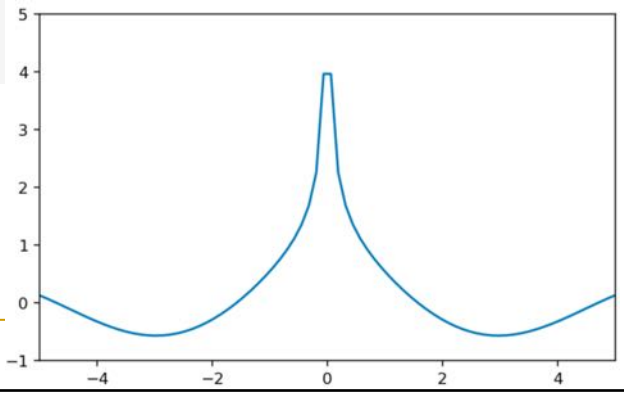


figure_v2.ipynb

13

```
#Adjusting plots
#axes limits
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(5,3), dpi=120)
ax=fig.add_axes([0,0,1,1])

def f(x):
    return np.cos(x)/np.sqrt(np.abs(x))

x = np.linspace(-2*np.pi,2*np.pi,100)
ax.plot(x, f(x));
ax.set_xlim(-5, 5)
ax.set_ylim(-1, 5);
plt.show()
```

Adjusting axes limits



figure_v2.ipynb

14

Walk through

figure_v2.ipynb

15

Matplotlib patches

- They are used to make drawings on a figure area. Possible options are
  - Polygons
  - Curves
  - Lines
  - Circle
  - Rectangle
  - Text
  - Path

patches.ipynb

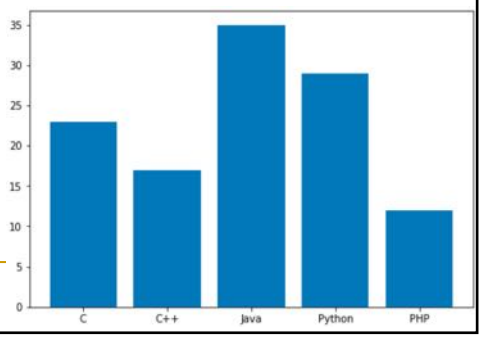
16

BAR PLOT: A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.
A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.

Matplotlib API provides the **bar()** function that can be used in the MATLAB style use as well as object oriented API. The signature of bar() function to be used with axes object is as follows : **ax.bar (x, height, width, bottom, align)**

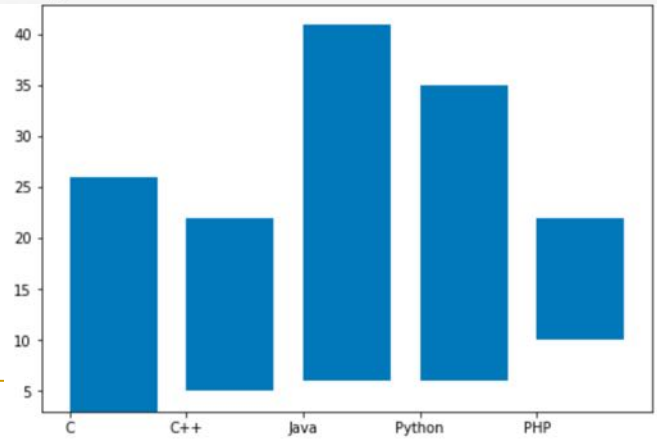
```
#bar plot example
#taken from tutorial point -modified
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(4,2), dpi=120)
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java',
         'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students, width=0.5)
plt.show()
```

Plot2_v2.ipynb



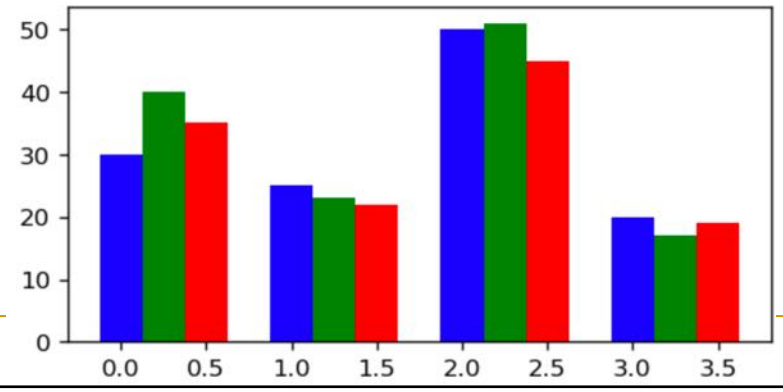
17

```
#bar plot example
#taken from tutorial point -modified
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(4,2), dpi=120)
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java',
         'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students, width=0.75, bottom=[3,5,6,6,10], align="edge" )
plt.show()
```



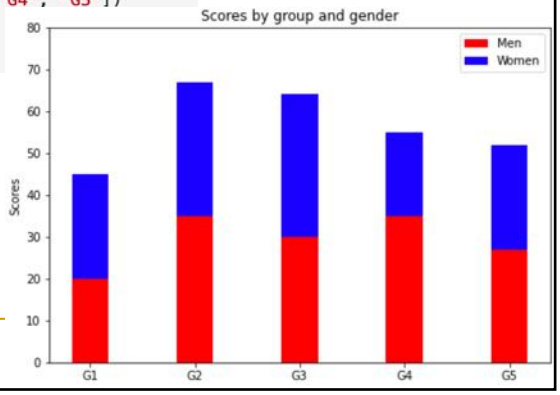
18

```python
#bar chart series
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(figsize=(4,2), dpi=120)
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)
plt.show()
```



19

```python
#stacked bar plots
import numpy as np
import matplotlib.pyplot as plt
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
ind = np.arange(N) # the x locations for the groups
width = 0.35
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, menMeans, width, color='r')
ax.bar(ind, womenMeans, width,bottom=menMeans, color='b')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks([0,1,2,3,4])
ax.set_xticklabels(['G1', 'G2', 'G3', 'G4', 'G5'])
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['Men', 'Women'])
plt.show()
```

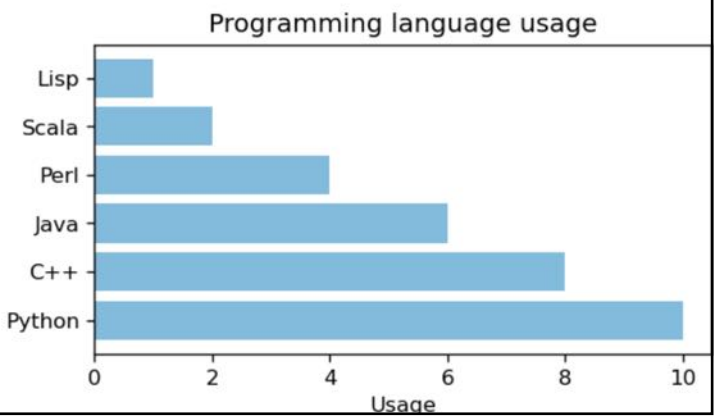


20

Horizontal bar plot

```
import matplotlib.pyplot as plt
import numpy as np
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

fig = plt.figure(figsize=(4,2), dpi=120)
ax = fig.add_axes([0,0,1,1])
plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Usage')
plt.title('Programming language usage')
plt.show()
```

plot2_v2.ipynb

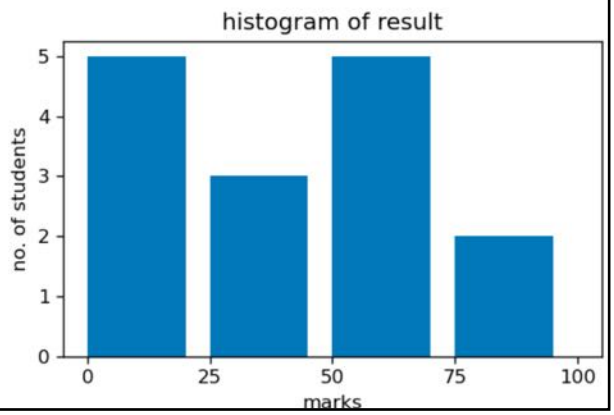


21

HISTOGRAM

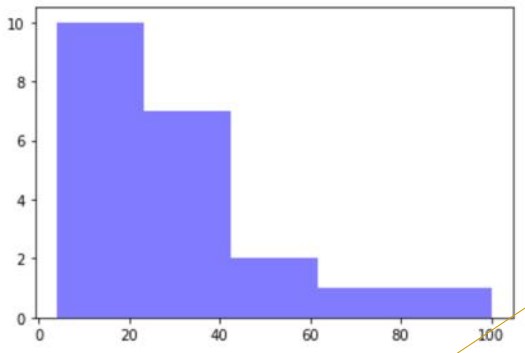
A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.
To construct a histogram, follow these steps ➔➔➔➔

```
#histogram
from matplotlib import pyplot as plt
import numpy as np

#fig = plt.figure(figsize=(4,2), dpi=120)
#ax = fig.add_axes([0,0,1,1])

fig,ax = plt.subplots(1,1, figsize=(5,3), dpi=120)
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], width = 20)
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()
```



22

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

x = [21,22,23,4,5,6,77,8,9,10,31,32,33,34,35,36,37,18,49,50,100]
num_bins = 5
n, bins, patches = plt.hist(x, num_bins, facecolor='blue', alpha=0.5)
plt.show()
print (n)
print (bins)
print (patches)
```



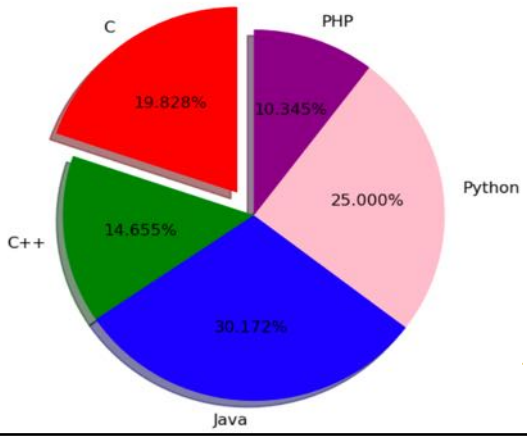
Data on the histogram plot

```
[10.  7.  2.  1.  1.]
[  4.   23.2  42.4  61.6  80.8 100. ]
<BarContainer object of 5 artists>
```

23

A **Pie Chart** can only display one series of data. Pie charts show the size of items (called wedge) in one data series, proportional to the sum of the items. The data points in a pie chart are shown as a percentage of the whole pie.

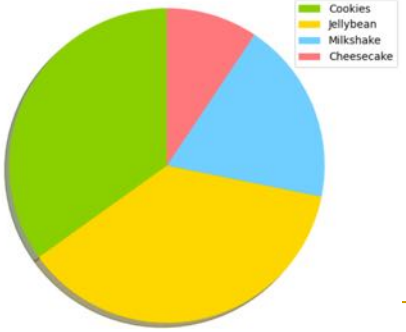
```
#pie chart
from matplotlib import pyplot as plt
import numpy as np
fig = plt.figure(figsize=(4,4), dpi=120)
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
cols = ["red", "green", "blue", "pink", "purple"]

explode = (0.15, 0, 0, 0,0)  # explode 1st slice
ax.pie(students, explode, labels = langs,autopct='%1.3f%%',
       shadow=True, startangle=90, colors=cols,
      )
plt.show()
```



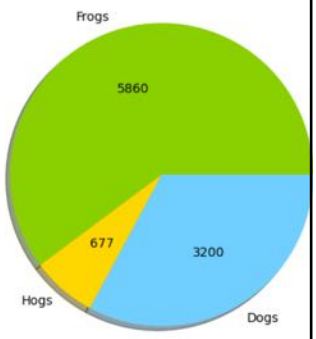
24

Examples

```
import matplotlib.pyplot as plt

labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']
sizes = [38.4, 40.6, 20.7, 10.3]
colors = ['yellowgreen', 'gold', 'lightskyblue',
          'lightcoral']
patches, texts = plt.pie(sizes, colors=colors,
                         shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy

labels = ['Frogs', 'Hogs', 'Dogs']
sizes = numpy.array([5860, 677, 3200])
colors = ['yellowgreen', 'gold', 'lightskyblue']

def absolute_value(val):
    a  = numpy.round(val/100.*sizes.sum(), 0)
    return int(a)

plt.pie(sizes, labels=labels, colors=colors,
        autopct=absolute_value, shadow=True)

plt.axis('equal')
plt.show()
```



25

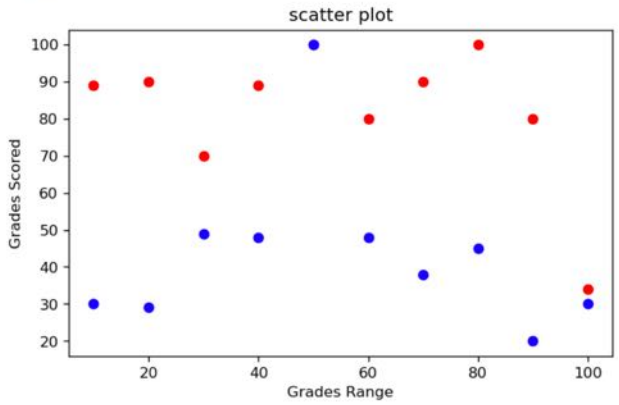
Scatter plots

are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another.

Each row in the data table is represented by a marker the position depends on its values in the columns set on the X and Y axes.

A third variable can be set to correspond to the color or size of the markers, thus adding yet another dimension to the plot.

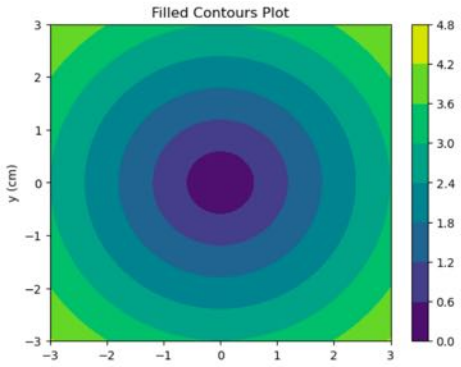
```
import matplotlib.pyplot as plt
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
fig = plt.figure(figsize=(5,3), dpi=120)
ax = fig.add_axes([0,0,1,1])
ax.scatter(grades_range, 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()
```



26

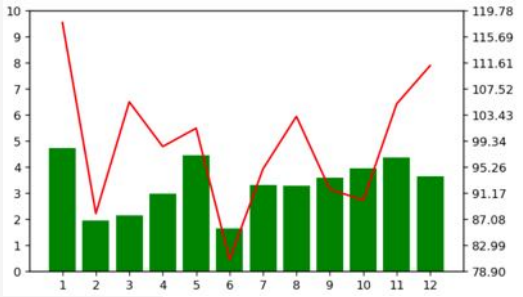
Contour plots (sometimes called Level Plots) are a way to show a three-dimensional surface on a two-dimensional plane. It graphs two predictor variables X Y on the y-axis and a response variable Z as contours.
A contour plot is appropriate if you want to see how value Z changes as a function of two inputs X and Y, such that **Z = f(X,Y).** A contour line or isoline of a function of two variables is a curve along which the function has a constant value.
The independent variables x and y are usually restricted to a regular grid called meshgrid. The **numpy.meshgrid** creates a rectangular grid out of an array of x values and an array of y values.

```
#contour plot example
import numpy as np
import matplotlib.pyplot as plt
xlist = np.linspace(-3.0, 3.0, 100)
ylist = np.linspace(-3.0, 3.0, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = np.sqrt(X**2 + Y**2)
fig,ax=plt.subplots(1,1)
cp = ax.contourf(X, Y, Z)
fig.colorbar(cp) # Add a colorbar to a plot
ax.set_title('Filled Contours Plot')
#ax.set_xlabel('x (cm)')
ax.set_ylabel('y (cm)')
plt.show()
```



27

```
#------------
#ref: https://queirozf.com/entries/matplotlib-pyplot-by-example
#(modified)
#------------------
import matplotlib.pyplot as plt
import numpy as np

# generate sample data for this example
xs = [1,2,3,4,5,6,7,8,9,10,11,12]
ys_bars = np.random.normal(loc=3.0,size=12)
ys_lines = np.random.normal(loc=5.0,size=12,scale=0.5)

fig = plt.figure(figsize=(5,3), dpi=120)
# this is the axis on the left
ax1 = fig.add_axes([0,0,1,1])
ax1.bar(xs,ys_bars,color='green')

# order is important when setting ticks.
# Ticks must be set after the plot has been drawn
ax1.set_yticks(np.arange(0,11,1))

# define the number of ticks
NUM_TICKS=11

# change the tick locator for this axis and set the desired number of ticks
ax1.yaxis.set_major_locator(plt.LinearLocator(numticks=NUM_TICKS))

# create the 'twin' axis on the right
ax2=ax1.twinx()

# plot the same numbers but multiplied by 20
ax2.plot(xs,ys_lines*20,color='red')

# set the ticks for the twin axis
ax2.set_yticks(np.arange(0,101,10))

# change the tick locator for this axis and set the desired number of ticks
ax2.yaxis.set_major_locator(plt.LinearLocator(numticks=NUM_TICKS))

# set ticks for the common x axis (bottom)
ax2.xaxis.set_ticks(xs)
plt.plot();
```



28

SEE YOU NEXT WEEK!!!

DR. ORHAN GÖKÇÖL
gokcol@gmail.com
orhan.gokcol@ozyegin.edu.tr

29