Nama : Verren Angelina Saputra

Kelas : XII.IPA

Mata Pelajaran : Ekskul Fisika (Grafik 2D dan 3D)

Hari / Tanggal : Jumat, 19 November 2021

Analisa Hasil dan Penulisan Kode Grafik 2D - 3D Dengan 2 *Library* (*Library Matplotlib* dan *Library Seaborn*)

I. Pendahuluan

Gambar I.1 Jupyter Notebook

(*Sumber : https://dev.to/bagasn/memulai-dan-menjalankan-jupyter-notebook-di-vs-code-185j , Diakses pada tanggal 10 Oktober 2018*)

Grafik 2 Dimensi (2D) merupakan sebuah grafik dengan teknik penggambaran yang hanya memiliki 2 titik koordinat yaitu sumbu x (datar) dan sumbu y (tegak) sebagai acuannya. Dimana grafik yang akan ditampilkan dengan teknik ini memiliki nilai koordinat x dan y minimum sebesar 0 dan maksimum bergantung sebesar resolusi yang digunakan. Sedangkan Grafik 3 Dimensi (3D) merupakan sebuah grafik yang memiliki panjang, lebar dan tinggi. Sehingga penggambaran grafik 3D memiliki acuan titik koordinat sumbu x (datar), sumbu y (tegak) dan sumbu z (miring). Representasi grafik geometrik 3D dikatakan sebagai hasil dari pemrosesan dan pemberian efek cahaya (*lighting*) terhadap grafik 2D.

Grafik 2 Dimensi dan 3 Dimensi dapat dibuat secara manual namun di zaman kemajuan teknologi seperti sekarang ini, muncul banyak media *digital* yang dapat digunakan sebagai alat bantu untuk menciptakan grafik secara efisien dan tervisualisasi dengan baik. Ada beragam aplikasi dan *platform* manapun yang dapat kita gunakan untuk memvisualisasikan sebuah *data*

menjadi grafik yang menarik, salah satunya dengan menggunakan aplikasi *Jupyter Notebook*. Keberadaan *Jupyter Notebook* memberikan penggunanya / para *user* untuk menciptakan sebuah grafik dengan menginput *library-library* yang dibutuhkan untuk menunjang proses visualisasi. Ada banyak *library* yang dapat kita gunakan di *platform* ini, namun secara spesifik pembuatan grafik 2D dan 3D hanya perlu membutuhkan *library matplotlib, seaborn* dan *NumPy* (apabila diperlukan).

Gambar I.2 *NumPy Python*

(*Sumber : https://id.wikipedia.org/wiki/NumPy , Diakses pada tanggal 11 Agustus 2021*)

Library NumPy atau singkatan dari *Numerical Python* merupakan salah satu *library* yang dapat kita gunakan di *Jupyter Notebook* untuk membantu proses komputasi numerik yang dapat mengolah sekumpulan variabel acak dan mengelompokkannya menjadi tipe *data* yang sama sehingga memudahkan operasi komputasi *data* yang ada. Dalam pembuatan grafik, *NumPy* biasanya digunakan untuk mengelompokkan variabel acak dalam x,y supaya terstruktur saat grafiknya terbentuk.

Gambar I.3 *Matplotlib Python*

(*Sumber : https://towardsdatascience.com/what-library-can-load-image-in-python-and-what-are-their-difference-d1628c6623ad , Diakses pada tanggal 30 Maret 2019*)

Selanjutnya, *library Matlotlib* merupakan sebuah *library* di dalam *python* yang dimanfaatkan dalam proses visualisasi *data* yang pertama kali diciptakan oleh John D. Hunter untuk menghasilkan *plot* grafik sesuai publikasi jurnal dan artikel ilmiah. Visualisasi pada *matplotlib* terpacu pada satu sumbu atau lebih baik sumbu *horizontal* (x), sumbu *vertikal* (y), dan *plot* lainnya yang direpresentasikan menjadi *glyphys* seperti bentuk pie (lingkaran) ataupun *lines* (garis) serta poligon.

Gambar I.4 *Seaborn Python*

(*Sumber : https://ai-pool.com/a/s/visualization-with-seaborn , Diakses pada tanggal 15 May 2021*)

Sedangkan *Library Seaborn* didefinisikan sebagai salah satu *library* di *python* yang digunakan untuk membantu visualisasi *data* baik secara sederhana atau yang lebih kompleks. *Seaborn* sangat berorientasi pada fungsi *plotting* berdasarkan sebuah *dataset* yang beroperasi untuk pemetaan statistik yang diperlukan dalam menghasilkan grafik informatif. *Seaborn* merupakan *library* yang digunakan untuk memvisualisasikan *data* yang dibangun diatas *library matplotlib*. Keunggulannya yaitu *seaborn* memiliki lebih banyak fungsi / kegunaan untuk melakukan visualisasi *data* dan lebih mudah digunakan *user*.

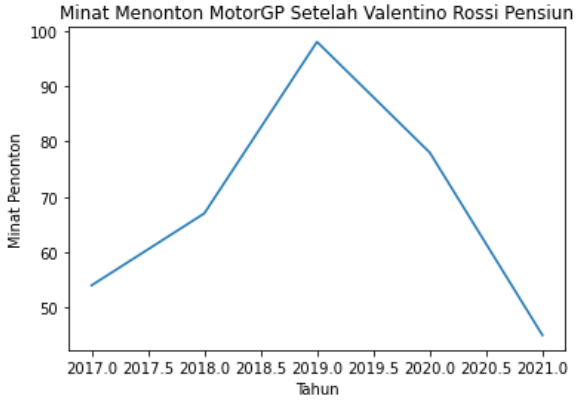
Kedua *library* ini dapat kita gunakan untuk membuat berbagai macam bentuk grafik baik *line chart, bar chart, polygon chart, pie chart, histogram chart*, dan masih banyak lainnya secara mudah dan tanpa memerlukan banyak *code* yang sulit. Meskipun bila diperhatikan keduanya sama-sama dimanfaatkan untuk visualisasi, namun *library Matplotlib* hanya sebatas digunakan untuk visualisasi dasar. Sedangkan untuk menghasilkan visualisasi yang lebih menarik dan lebih dinamis maka kita bisa meneruskannya dengan menggunakan *library Seaborn.*

Maka dari itu, diharapkan pendahuluan ini bisa menjadi dasar dan acuan agar bisa melakukan komparasi antara kedua *library* yakni *Library Seaborn* dan juga *Library NumPy* dengan setiap grafik yang akan ditampilkan di bawah ini. Sehingga didapat sebuah kesimpulan yang akurat dan sesuai dengan setiap teori-teori yang telah dikaji secara runtut di bab ini.

II. Hasil Grafik, Kode, dan Deskripsi

Berikut saya tampilkan hasil grafik 2D dan 3D yang telah saya buat dengan menggunakan bantuan *library Seaborn, library Matplotlib* dan juga *library NumPy* beserta dengan penulisan kode dan deskripsi mengenai kumpulan grafik-grafik tersebut.

1. Membuat Grafik 2D = *Line Chart*

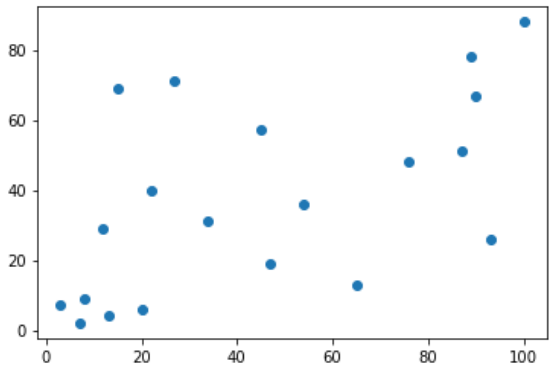


Kode =

```
# 1. Membuat Grafik Garis 2D

plt.plot([2017,2018,2019,2020,2021],[54, 67, 98, 78, 45])
plt.title('Minat Menonton MotorGP Setelah Valentino Rossi Pensiun')
plt.xlabel('Tahun')
plt.ylabel('Minat Penonton')
plt.show()
```

Deskripsi = Pada grafik ini, sumbu x memuat *data* tahun dan sumbu y memuat *data* frekuensi. *Line chart* di atas merupakan hasil visualisasi dasar dari *matplotlib*, *library matplotlib* mengubah serangkaian *data* menjadi grafik garis yang kontinu dari tahun 2017-2021 mengenai "Minat Menonton MotorGP Setelah Valentino Rossi Pensiun".

2. Membuat Grafik 2D = *Scatter Chart*

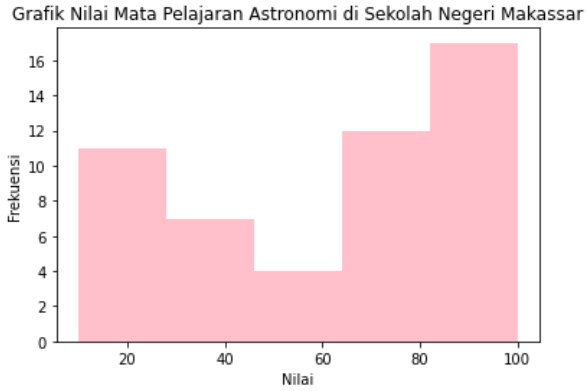


Kode =

```
# 2. Membuat Plot Sebaran 2D

x = [7,13,20,3,8,65,47,93,12,34,54,22,76,87,45,90,15,27,89,100]
y = [2,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
plt.scatter(x,y)
plt.show()
```

Deskripsi = Pada grafik ini, sumbu x dan y memuat angka random dengan jumlah yang sama. Dimana kumpulan angka ini akan divisualisasikan oleh *library matplotlib* menjadi grafik sederhana dengan bentuk titik-titik yang dikenal dengan sebutan *scatter chart*.

3. Membuat Grafik 2D = *Histogram Chart*

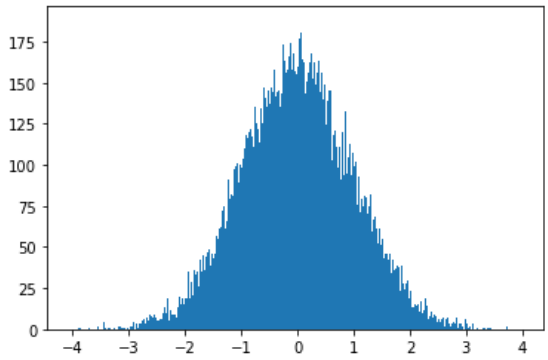


Kode =

```
# 3. Membuat Plot Histogram 2D

x = [100,10,30,20,15,70,75,80,80,90,95,90,100,100,40,30,70,50,60,100,90,20,10,10,40,70,80,90,95,45,65,60,75,80,95,25,10,45,75,85
num_bins = 5
n, bins, patches = plt.hist(x, num_bins, facecolor = 'pink')
plt.title('Grafik Nilai Mata Pelajaran Astronomi di Sekolah Negeri Makassar')
plt.xlabel('Nilai')
plt.ylabel('Frekuensi')
plt.show()
```

Deskripsi = Pada grafik di atas, terdapat sekumpulan angka acak dalam rentang 10-100 yang dikumpulkan dalam grup x. Sekumpulan *data* ini akan dikelompokkan dalam 5 *bar* yang berdekatan dikenal sebagai *histogram* oleh *library Matplotlib* dengan warna pink. 100 dibagi 5 menjadi 20, jadi setiap *histogram* berisikan angka yang terdiri dari rentang 0-20, 20-40, 40-60, 60-80, dan 80-100. Sekumpulan angka tersebut bertopik “Grafik Nilai Mata Pelajaran Astronomi di Sekolah Negeri Makassar”.

4. **Membuat Grafik 2D = *Histogram Chart***

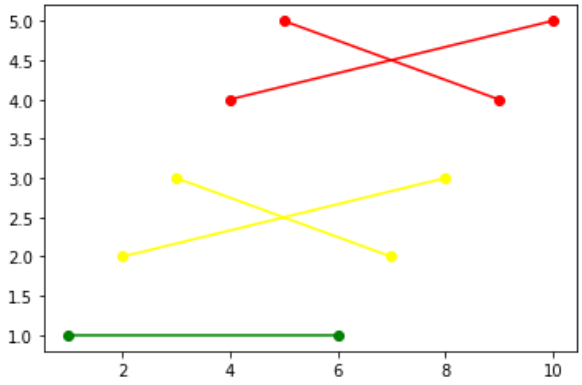


Kode =

```
# 4. Membuat Plot histogram 2D

import matplotlib.pyplot as plt
import numpy as np

x = np.random.randn(50000)

plt.hist(x, 1000)
plt.show()
```

Deskripsi = Pada grafik di atas, *library matplotlib* memvisualisasikan sebuah komputasi numerik acak menjadi bentuk *histogram*. Karena rentang datanya sangat luas maka bentuk garis histogramnya tipis dan berlapis hingga gambar grafiknya seperti yang ada di atas.

5. **Membuat Grafik 2D = *X line Chart***

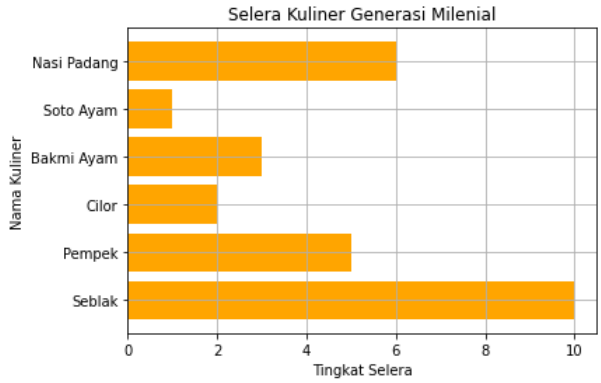


Kode =

```
# 5. Membuat Grafik Garis X 2D

x=(1,2,3,4,5,1,2,3,4,5)
y=(1,2,3,4,5,6,7,8,9,10)
status=[0,0,0,0,0,0,0,0,0,0]
new_status=[3,0,0,2,0,0,0,0,4,1]
ax = plt.subplot()
y16 = (y[0], y[5])
x16 = (x[0], x[5])
ax.plot(y16 ,x16, marker='o', color='green')
y28 = (y[7], y[1])
x28 = (x[7], x[1])
ax.plot(y28, x28, marker='o', color='yellow')
y37 = (y[2], y[6])
x37 = (x[2], x[6])
ax.plot(y37, x37, marker='o', color='yellow')
y410 = (y[3], y[9])
x410 = (x[3], x[9])
ax.plot(y410, x410, marker='o', color='red', )
y59 = (y[4], y[8])
x59 = (x[4], x[8])
ax.plot(y59, x59, marker='o', color='red')
plt.show()
```

Deskripsi = Pada grafik di atas, terdapat *data* angka yang terletak di sumbu x dan y. Kemudian kita kelompokkan *data* supaya saling berpasang-pasangan menjadi garis X yang akan divisualisasikan dengan sebuah *library* yakni *library matplotlib*. Dimana setiap garis-garis X berwarna-warni, ada yang berwarna hijau, kuning dan merah.

6. Membuat Grafik 2D = *Horizontal Bar Chart*

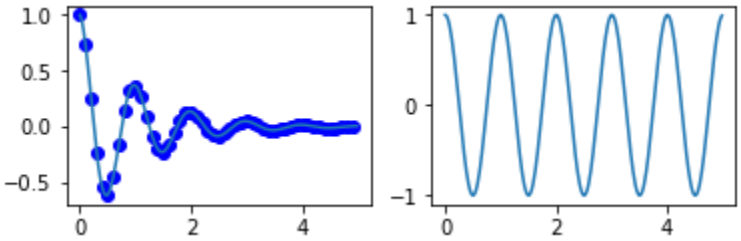


Kode =

```
# 6. Membuat grafik horizontal dengan library pylab

from pylab import *
pos = arange(6) + .5
barh(pos, (10, 5, 2, 3, 1, 6), align='center', color='Orange')
yticks(pos, ('Seblak', 'Pempek', 'Cilor', 'Bakmi Ayam', 'Soto Ayam','Nasi Padang'))
xlabel('Tingkat Selera')
ylabel('Nama Kuliner')
title('Selera Kuliner Generasi Milenial')
grid(True)
show()
```

Deskripsi = Pada grafik di atas, dapat dideskripsikan sebuah *data* mengenai "Selera Kuliner Generasi Millenial" yang divisualisasikan dengan *library* di *python* dengan warna oranye. Sumbu Y diisi dengan 6 variabel / 6 jenis makanan yakni Seblak, Pempek, Cilor, Bakmi Ayam, Soto Ayam dan Nasi Padang. Sumbu Y diisi dengan frekuensi minat generasi milenial. Biasanya *Bar* posisinya *vertikal* namun karena posisinya menyamping makanya dikenal dengan sebutan *horizontal bar chart*.

7. **Membuat Grafik 2D = *Plot and Line Chart***

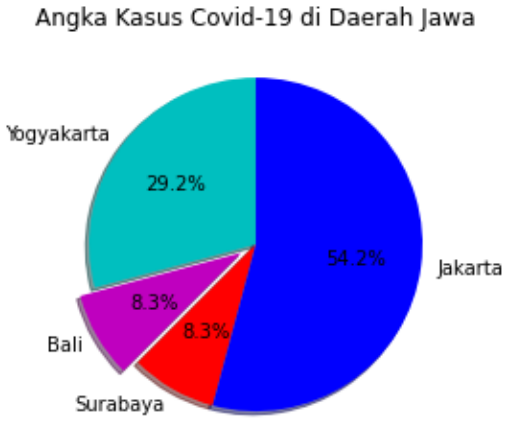


Kode =

```
# 7. Membuat Multigrafik (Plot dan Line) 2D

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)
plt.subplot(221)
plt.plot(t1, f(t1), 'bo', t2, f(t2))
plt.subplot(222)
plt.plot(t2, np.cos(2*np.pi*t2))
plt.show()
```

Deskripsi = Pada grafik di atas, *library matplotlib* memvisualisasikan setiap *data-data* menjadi bentuk perpaduan grafik yaitu grafik *scatter plot* dan juga *line (curve)* yang berwarna biru.

8. **Membuat Grafik 2D = *Pie Chart***

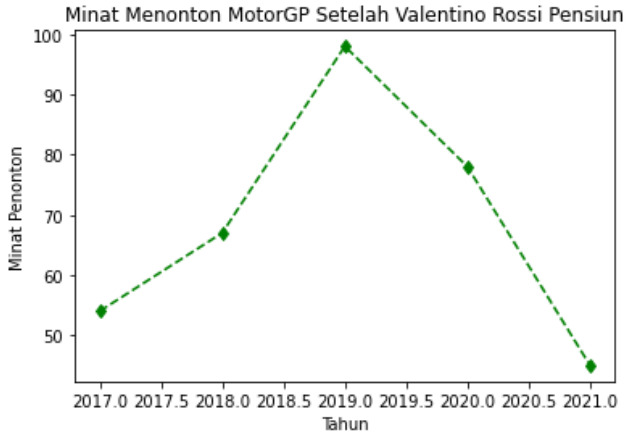


Kode =

```
# 8. Membuat Grafik Pie 2D

days = [1,2,3,4,5]

Yogyakarta =[7,8,6,11,7]
Bali = [2,3,4,3,2]
Surabaya =[7,8,7,2,2]
Jakarta = [8,5,7,8,13]
slices = [7,2,2,13]
activities = ['Yogyakarta','Bali','Surabaya','Jakarta']
cols = ['c','m','r','b']

plt.pie(slices,
  labels=activities,
  colors=cols,
  startangle=90,
  shadow= True,
  explode=(0,0.1,0,0),
  autopct='%1.1f%%')

plt.title('Angka Kasus Covid-19 di Daerah Jawa')
plt.show()
```

Deskripsi = Pada grafik di atas, *library* yang ada dalam *python* yakni *matplotlib* memvisualisasikan sekumpulan *data* yang diterima pada 4 variabel yakni Yogyakarta, Bali, Surabaya, dan Jakarta menjadi *pie chart*. Dimana *pie chart* tersebut digunakan untuk mendeskripsikan grafik mengenai "Angka Kasus Covid-19 Di Daerah Jawa".

9. Membuat Grafik 2D = *Dotted Line Chart*

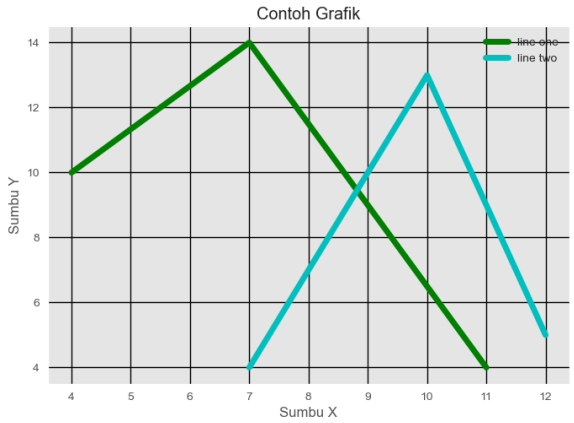


Kode =

```
# 9. Membuat Grafik Garis Putus-putus 2D

plt.plot([2017,2018,2019,2020,2021],[54, 67, 98, 78, 45],'g--d')
plt.title('Minat Menonton MotorGP Setelah Valentino Rossi Pensiun')
plt.xlabel('Tahun')
plt.ylabel('Minat Penonton')
plt.show()
```

Deskripsi = Pada grafik di atas, kurang lebih gambaran datanya hampir mirip dengan grafik paling pertama. Namun yang membedakannya di visual grafik, dimana fungsi "g—d" pada *jupyter notebook* bisa menghasilkan bentuk grafik garis putus-putus / garis titik atau yang dikenal dengan *dotted line chart*.

10. Membuat Grafik 2D = *Multi-line Chart*

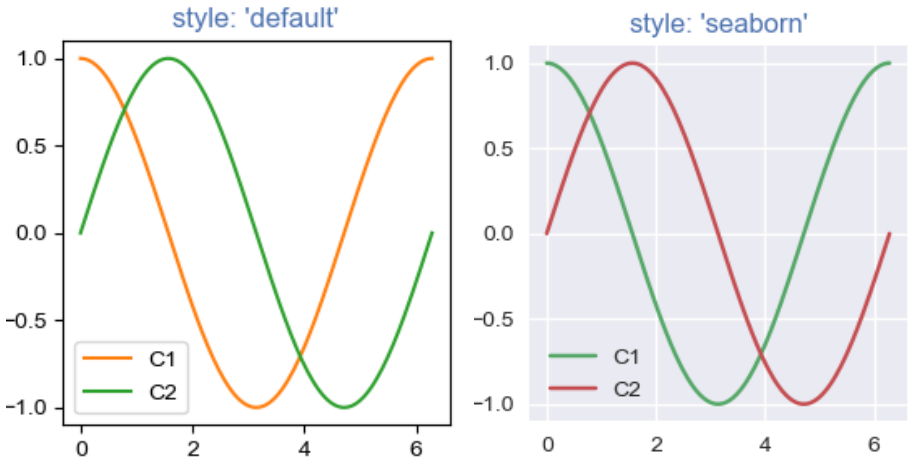


Kode =

```
# 10. Membuat Grafik Multiline 2D

style.use('ggplot')
x = [4,7,11]
y = [10,14,4]
x2 = [7,10,12]
y2 = [4,13,5]
plt.plot(x,y,'g',label='line one', linewidth=5)
plt.plot(x2,y2,'c',label='line two',linewidth=5)
plt.title('Contoh Grafik')
plt.ylabel('Sumbu Y')
plt.xlabel('Sumbu X')
plt.legend()
plt.grid(True,color='k')
plt.show()
```

Deskripsi = Pada grafik di atas, dapat dideskripsikan bahwa *library python* dapat memvisualisasikan sebuah grafik dari *data* grup x dan y menjadi multi line atau grafik yang terdiri lebih dari 1 garis. Untuk memudahkan dalam membedakannya maka kita bisa memberi warna yang berbeda yakni hijau dan biru muda.

11. Membuat Grafik 2D = *Multicurve with 2 style Chart*

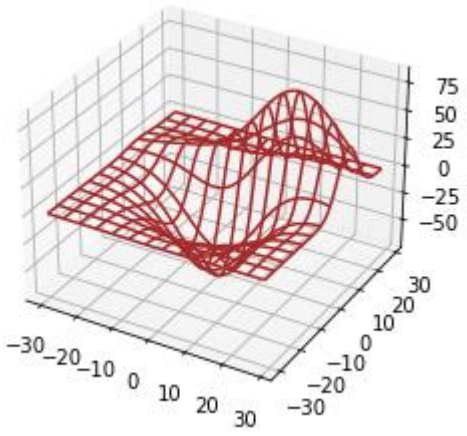


Kode =

```
# Grafik Default dan Grafik dengan Seaborn

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

th = np.linspace(0, 2*np.pi, 128)


def demo(sty):
    mpl.style.use(sty)
    fig, ax = plt.subplots(figsize=(3, 3))

    ax.set_title('style: {!r}'.format(sty), color='C0')

    ax.plot(th, np.cos(th), 'C1', label='C1')
    ax.plot(th, np.sin(th), 'C2', label='C2')
    ax.legend()


demo('default')
demo('seaborn')
```

Deskripsi = Pada grafik di atas, kita bisa melakukan aktivitas komparasi untuk membedakan *style* garis *default* dengan *library seaborn*.

12. Membuat Grafik 3D = *Wireframe Chart*

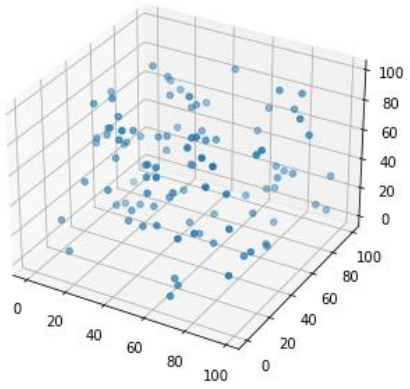


Kode =

```
# 1. Membuat grafik 3D dengan wireframe (lengkungan isi)
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x, y, z = axes3d.get_test_data(0.15)
ax.plot_wireframe(x, y, z, rstride=3, cstride=3, color = "brown")
plt.show()
```

Deskripsi = Pada grafik di atas, kita bisa mengetahui bahwa *jupyter notebook* dapat memvisualisasikan grafik 3 Dimensi. Dengan program yang ada kita bisa membentuk *wireframe chart* yaitu grafik dengan lengkungan isi. Pada kode, terdapat 3 sumbu yang diperlukan yakni sumbu x, sumbu y, dan sumbu z. Berbeda dengan grafik 2D yang hanya membutuhkan 2 sumbu yakni sumbu x dan sumbu y saja.

13. Membuat Grafik 3D = *Scatter 3D Chart*

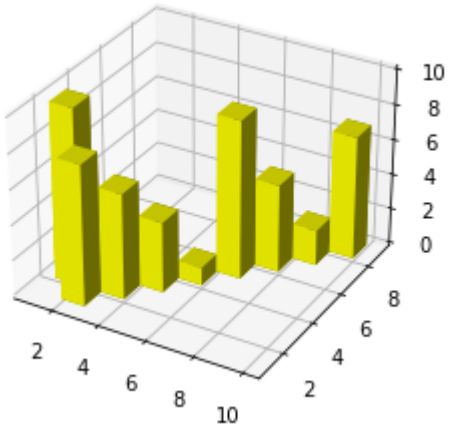


Kode =

```
# 2. Membuat grafik 3D dengan Plot Sebaran

from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
import random


fig = pyplot.figure()
ax = Axes3D(fig)

sequence_containing_x_vals = list(range(0, 100))
sequence_containing_y_vals = list(range(0, 100))
sequence_containing_z_vals = list(range(0, 100))

random.shuffle(sequence_containing_x_vals)
random.shuffle(sequence_containing_y_vals)
random.shuffle(sequence_containing_z_vals)

ax.scatter(sequence_containing_x_vals, sequence_containing_y_vals, sequence_containing_z_vals)
pyplot.show()
```

Deskripsi = Pada grafik di atas, dapat kita perhatikan bahwa *library matplotlib* dapat mengubah serangkaian kode diatas menjadi grafik 3 Dimensi. Dengan kode di atas, kita mampu menciptakan grafik 3 Dimensi yang berisi sekumpulan titik atau *scatter*.

14. Membuat Grafik 3D = *Bar 3D Chart*

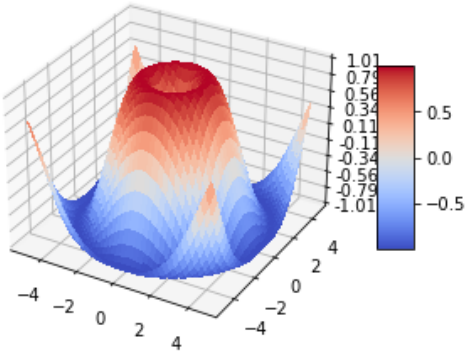


Kode =

```
# 3. Membuat grafik 3D dengan isi histogram

from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from random import randint
fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')
xpos = [5, 8, 1, 4, 7, 3, 9, 2, 6, 1]
ypos = [4, 7, 2, 3, 6, 2, 8, 1, 5, 2]
np.zeros(10)
np.ones(10)
zpos = np.zeros(10)
dx = np.ones(10)
dy = np.ones(10)
dz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ax1.bar3d(xpos, ypos, zpos, dx, dy, dz, color='yellow')
plt.show()
```

Deskripsi = Pada grafik di atas, dapat divisualisasikan sebuah grafik 3 Dimensi dengan bentuk *Bar* yang memiliki panjang, lebar dan juga tinggi. Visualisasi grafik diberi warna yang bisa di *custom*, untuk kode di atas memberikan perintah program untuk mewarnai grafik dengan warna kuning.

15. Membuat Grafik 3D = *Surpace Chart*

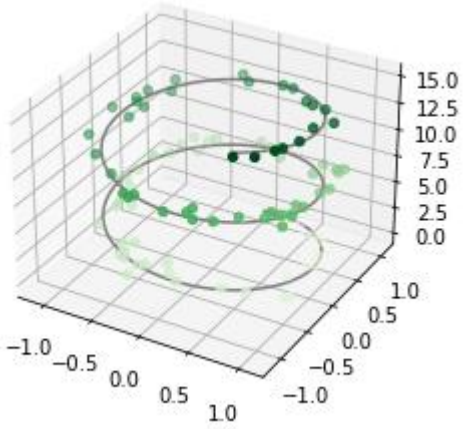


Kode =

```
# 4. Membuat Grafik 3D Surface (Color Map)

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import numpy as np

fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
# A StrMethodFormatter is used automatically
ax.zaxis.set_major_formatter('{x:.02f}')

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

Deskripsi = Pada grafik di atas, dapat kita buat sebuah kode untuk memberikan perintah memvisualisasikan sebuah *data* yang ada menjadi bentuk lempengan grafik yang memiliki volume sehingga disebut dengan *surface chart*. Untuk warna gradasinya kita berikan perintah program untuk mewarnai grafik menjadi warna merah dan biru.

16. Membuat Grafik 3D = *Spiral Line 3D Chart*

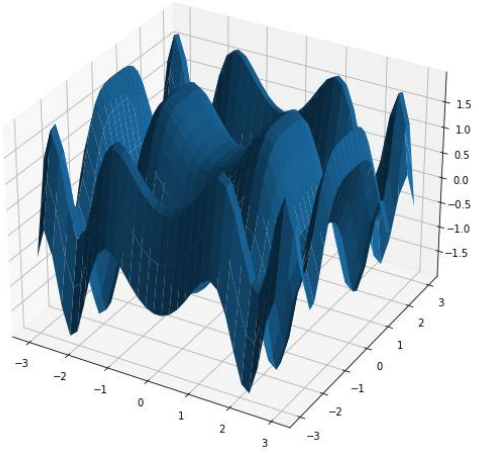


Kode =

```
# 5. Membuat Grafik 3D Plot dan Line

ax = plt.axes(projection='3d')

# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'Gray')

# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```

Deskripsi = Pada grafik di atas, kita dapat memvisualisasikan sebuah grafik 3 Dimensi berupa garis spiral yang dikelilingi dengan titik / *scatter* berwarna hijau.

17. Membuat Grafik 3D = *Surface Chart*

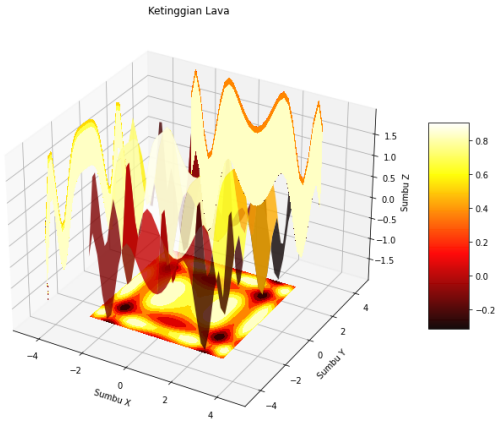


Kode =

```
# 6. Membuat Grafik 3D Surface

from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt


# Creating dataset
x = np.outer(np.linspace(-3, 3, 32), np.ones(32))
y = x.copy().T # transpose
z = (np.sin(x **2) + np.cos(y **2) )

# Creating figure
fig = plt.figure(figsize =(14, 9))
ax = plt.axes(projection ='3d')

# Creating plot
ax.plot_surface(x, y, z)

# show plot
plt.show()
```

Deskripsi = Pada grafik di atas, divisualisasikan sebuah grafik 3 dimensi berupa grafik gelombang yang memiliki panjang, lebar dan juga tinggi berupa volume atau ruang berwarna biru.

18. Membuat Grafik 3D = *Surface Chart*



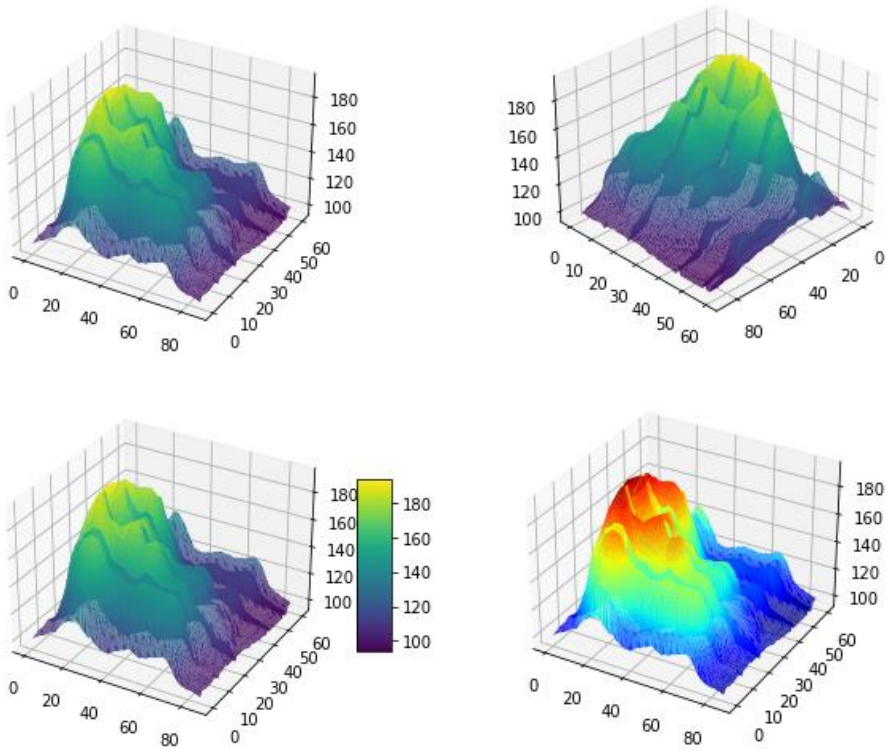
Kode =

```
# 7. Membuat Grafik 3D Surface Dengan Warna Gradient

from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# Creating dataset
x = np.outer(np.linspace(-3, 3, 32), np.ones(32))
y = x.copy().T # transpose
z = (np.sin(x **2) + np.cos(y **2) )

# Creating figure
fig = plt.figure(figsize =(14, 9))
ax = plt.axes(projection ='3d')

# Creating color map
my_cmap = plt.get_cmap('hot')
```

```
surf = ax.plot_surface(x, y, z,
                       rstride = 8,
                       cstride = 8,
                       alpha = 0.8,
                       cmap = my_cmap)
cset = ax.contourf(x, y, z,
                   zdir ='z',
                   offset = np.min(z),
                   cmap = my_cmap)
cset = ax.contourf(x, y, z,
                   zdir ='x',
                   offset =-5,
                   cmap = my_cmap)
cset = ax.contourf(x, y, z,
                   zdir ='y',
                   offset = 5,
                   cmap = my_cmap)
fig.colorbar(surf, ax = ax,
             shrink = 0.5,
             aspect = 5)

# Adding labels
ax.set_xlabel('Sumbu X')
ax.set_xlim(-5, 5)
ax.set_ylabel('Sumbu Y')
ax.set_ylim(-5, 5)
ax.set_zlabel('Sumbu Z')
ax.set_zlim(np.min(z), np.max(z))
ax.set_title('Ketinggian Lava')

# show plot
plt.show()
```

Deskripsi = Pada grafik di atas, gambaran grafiknya hampir mirip dengan yang sebelumnya. Namun untuk pewarnaannya kita program dalam kode *colorbar* yang disetting mulai dari segi *surf, shrink* dan *aspect* supaya menghasilkan gradasi warna seperti lava/magma.

19. Membuat Grafik 3D = *Surface Chart*



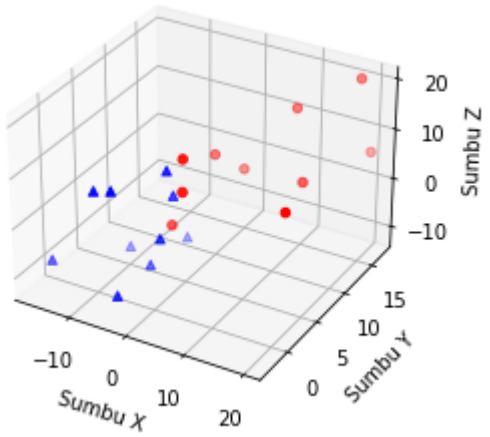
Kode =

```
# 8. Membuat Grafik 3D dengan Surface Plot

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Dapatkan data di file csv
url = 'https://raw.githubusercontent.com/holtzy/The-Python-Graph-Gallery/master/static/data/volcano.csv'
data = pd.read_csv(url)

# Ubah ke format panjang
df=data.unstack().reset_index()
df.columns=["X","Y","Z"]

# Dan ubah nama kolom lama menjadi numerik
df['X']=pd.Categorical(df['X'])
df['X']=df['X'].cat.codes

# Buka plotnya
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_trisurf(df['Y'], df['X'], df['Z'], cmap=plt.cm.viridis, linewidth=0.2)
plt.show()

# untuk Menambahkan bilah warna yang memetakan nilai ke warna.
fig = plt.figure()
ax = fig.gca(projection='3d')
surf=ax.plot_trisurf(df['Y'], df['X'], df['Z'], cmap=plt.cm.viridis, linewidth=0.2)
fig.colorbar( surf, shrink=0.5, aspect=5)
plt.show()
```

```
# Putar Plot
fig = plt.figure()
ax = fig.gca(projection='3d')
surf=ax.plot_trisurf(df['Y'], df['X'], df['Z'], cmap=plt.cm.viridis, linewidth=0.2)
ax.view_init(30, 45)
plt.show()

# Plot Lainnya
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_trisurf(df['Y'], df['X'], df['Z'], cmap=plt.cm.jet, linewidth=0.01)
plt.show()
```

Deskripsi = Pada grafik di atas, dapat kita ketahui bahwa program atau kode kompleks bisa menghasilkan grafik yang kompleks. Visualisasi *data* menghasilkan bentuk grafik seperti dataran yang dipadukan dengan warna kuning, hijau, dan ungu terang serta warna Pelangi pada grafik paling bawah. Bahkan kita membuat kode untuk mengukur ketinggian warna / rentang warna di samping grafik dengan kode *df.colums*.

20. Membuat Grafik 3D = *Multi-scatter 3D Chart*

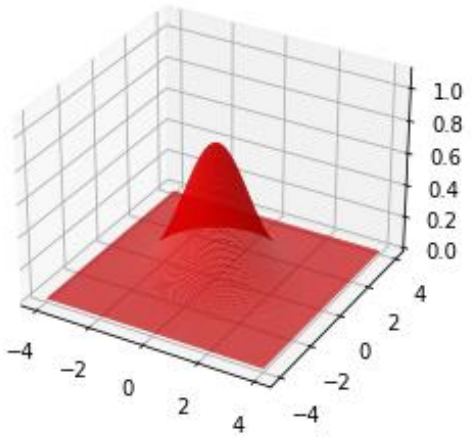


Kode =

```
# 9. Membuat Grafik 3D Dengan 2 Variabel

from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from random import randint
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
X = [randint(0, 20) for i in range(10)]
Y = [randint(0, 20) for i in range(10)]
Z = [randint(0, 20) for i in range(10)]
Xs = [randint(-20, 0) for i in range(10)]
Ys = [randint(-7, 9) for i in range(10)]
Zs = [randint(-16, 12) for i in range(10)]
ax.scatter(X,Y,Z, c='r', marker='o')
ax.scatter(Xs,Ys,Zs, c='b', marker='^')
ax.set_xlabel('Sumbu X')
ax.set_ylabel('Sumbu Y')
ax.set_zlabel('Sumbu Z')
plt.show()
```

Deskripsi = Pada grafik di atas, *library* yang kita gunakan dapat memvisualisasikan sebuah grafik 3 dimensi dengan isian titik-titik halus (jenis bentuk ada yang segitiga dan lingkaran) yang posisinya tersebar luas sesuai kode yang kita sudah buat.

21. Membuat Grafik 3D = *Surface Chart*



Kode =

```
# 10.Membuat Grafik 3D dengan Surface Plot

import numpy as np
print('numpy: '+np.version.full_version)
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
import matplotlib
print('matplotlib: '+matplotlib.__version__)

N = 150 # Meshsize
fps = 10 # frame per sec
frn = 50 # frame number of the animation

x = np.linspace(-4,4,N+1)
x, y = np.meshgrid(x, x)
zarray = np.zeros((N+1, N+1, frn))

f = lambda x,y,sig : 1/np.sqrt(sig)*np.exp(-(x**2+y**2)/sig**2)

for i in range(frn):
    zarray[:,:,i] = f(x,y,1.5+np.sin(i*2*np.pi/frn))

def update_plot(frame_number, zarray, plot):
    plot[0].remove()
    plot[0] = ax.plot_surface(x, y, zarray[:,:,frame_number], cmap="magma", color ="red")
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

plot = [ax.plot_surface(x, y, zarray[:,:,0], color='red', rstride=1, cstride=1)]
ax.set_zlim(0,1.1)
ani = animation.FuncAnimation(fig, update_plot, frn, fargs=(zarray, plot), interval=1000/fps)
```

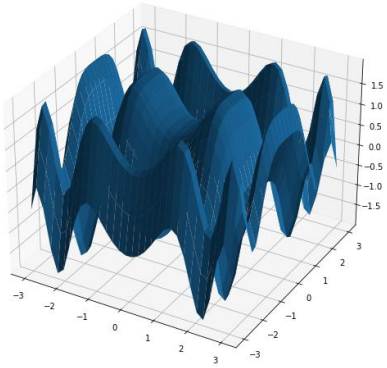
Deskripsi = Pada grafik di atas, grafik 3 dimensi membutuhkan 3 jenis sumbu mulai dari sumbu x, sumbu y dan sumbu z untuk memvisualisasikan grafik yang memiliki ruang / volume dengan warna merah.

III. Analisa

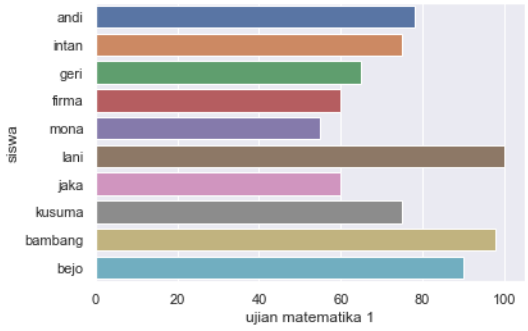
Berdasarkan setiap grafik-grafik yang telah dibuat dengan bantuan beberapa *library*, bisa ditemukan sebuah hasil analisa untuk melakukan komparasi terhadap *library Seaborn* dan juga *library Matplotlib*. Keduanya sangat bermanfaat untuk proses visualisasi *data* menjadi bentuk grafik, namun ada beberapa aspek yang bisa dijadikan perbandingan mulai dari segi fungsionalitas, komputasi angka, kode program, visualisasi *data*, *data frame* dan *array*, fleksibilitas dan kemampuan *problem solve*. Berikut adalah penjabarannya :

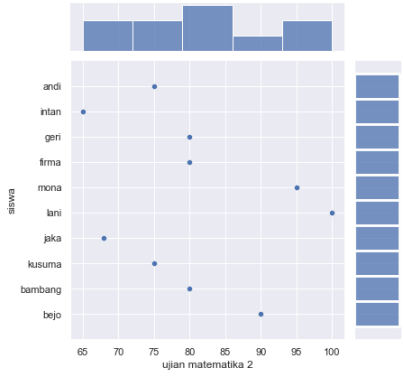
1. Fungsionalitas

- ***Matplotlib*** = *Library matplotlib* digunakan untuk *plot* dasar saja. Biasanya digunakan untuk keperluan memvisualisasi *bar, pie, lines, scatter plot* dan masih banyak lainnya baik untuk grafik berbentuk 2 Dimensi ataupun 3 Dimensi.



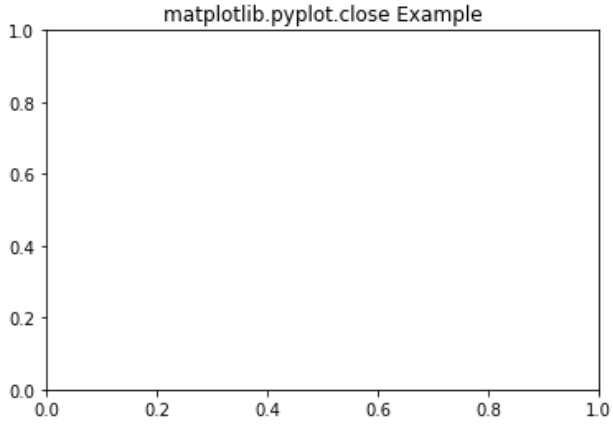
- ***Seaborn*** = *Library seaborn* lebih lengkap dibandingkan *library matplotlib* karena menyediakan berbagai pola visualisasi, lebih sedikit sintaks, dan memiliki tema *default* yang sangat menarik. Namun *library* ini lebih spesifik digunakan untuk visualisasi statistik dan digunakan apabila seseorang harus meringkas *data* dalam visualisasi serta menunjukkan distribusi dalam *data*.





2. Komputasi Angka

- ***Matplotlib*** = *Library matplotlib* memiliki beberapa angka yang dapat dibuka, namun perlu ditutup secara eksplisit dengan kode *plt.close()* untuk menutup gambar / visualisasi saat ini sedangkan apabila kita ketik "*all*" di dalam kurung menjadi *plt.close(all)* maka kode akan menutup semuanya.

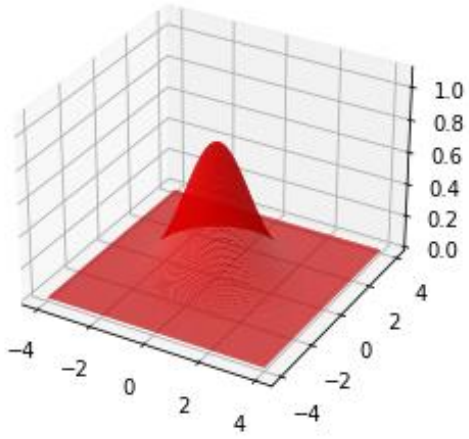



```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LogNorm


dx, dy = 0.015, 0.05
x = np.arange(-4.0, 4.0, dx)
y = np.arange(-4.0, 4.0, dy)
X, Y = np.meshgrid(x, y)

extent = np.min(x), np.max(x), np.min(y), np.max(y)

Z1 = np.add.outer(range(8), range(8)) % 2
plt.imshow(Z1, cmap ="binary_r",
           interpolation ='nearest',
           extent = extent,
           alpha = 1)

def geeks(x, y):
    return (1 - x / 2 + x**5 + y**6) * np.exp(-(x**2 + y**2))

Z2 = geeks(X, Y)

x = plt.imshow(Z2, cmap ="Greens",
               alpha = 0.7,
               interpolation ='bilinear',
               extent = extent)
plt.close()
plt.title('matplotlib.pyplot.close Example')
plt.show()
```

- ***Seaborn*** = *Library seaborn* lebih ringkas dan praktis dalam mengkomputasi angka karena semuanya dilakukan secara otomatis meski terhadap menyebabkan masalah *OOM* / Kehabisan Memori.

3. Kode program

- ***Matplotlib*** = Kode program untuk menghasilkan visualisasi *data* pada *library matplotlib* lebih kompleks. Misalnya terdapat modul *pylot* mencerminkan perintah *plot MATLAB* dengan cermat sehingga penggunaan *MATLAB* dapat dengan mudah *transit* untuk membuat *plot* dengan *python*. Jadi bisa dikatakan kurang efisien karena harus melakukan perpindahan / *transit* program dengan berbagai *syntax* atau kode programnya harus melalui serangkaian proses tidak langsung untuk menciptakan visualisasi *data* yang diinginkan

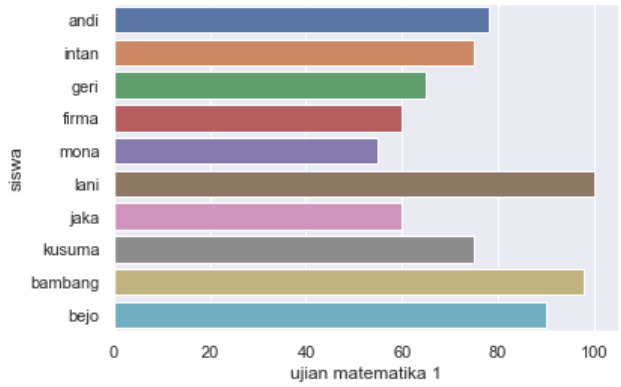



```
# 10.Membuat Grafik 3D dengan Surface Plot

import numpy as np
print('numpy: '+np.version.full_version)
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
import matplotlib
print('matplotlib: '+matplotlib.__version__)

N = 150 # Meshsize
fps = 10 # frame per sec
frn = 50 # frame number of the animation

x = np.linspace(-4,4,N+1)
x, y = np.meshgrid(x, x)
zarray = np.zeros((N+1, N+1, frn))

f = lambda x,y,sig : 1/np.sqrt(sig)*np.exp(-(x**2+y**2)/sig**2)

for i in range(frn):
    zarray[:,:,i] = f(x,y,1.5+np.sin(i*2*np.pi/frn))

def update_plot(frame_number, zarray, plot):
    plot[0].remove()
    plot[0] = ax.plot_surface(x, y, zarray[:,:,frame_number], cmap="magma", color ="red")
```

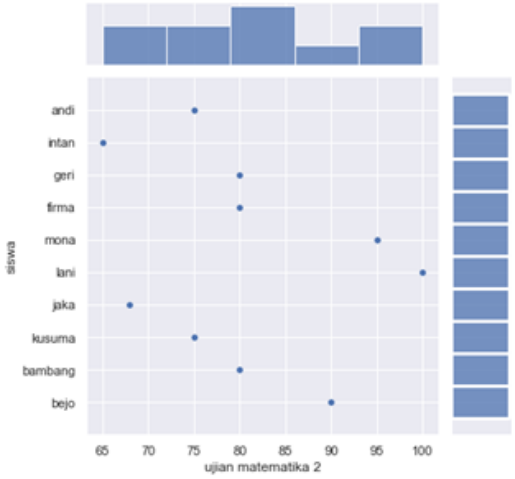
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

plot = [ax.plot_surface(x, y, zarray[:,:,0], color='red', rstride=1, cstride=1)]
ax.set_zlim(0,1.1)
ani = animation.FuncAnimation(fig, update_plot, frn, fargs=(zarray, plot), interval=1000/fps)
```

- ***Seaborn*** = Kode program untuk menghasilkan visualisasi *data* pada *library matplotlib* lebih singkat dan lebih mudah karena tidak berbelit-belit *syntax* nya dan lebih mengacu pada *direct code*.

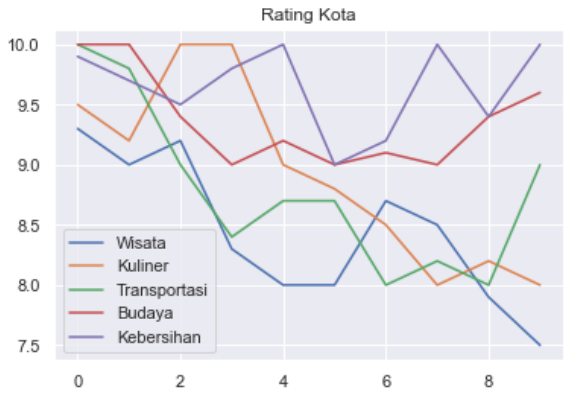
```
sns.barplot(x='ujian matematika 1', y='siswa', data=tabel)
plt.show()
```




```
sns.jointplot(x='ujian matematika 2', y= 'siswa',data=tabel);
plt.show()
```




```
dataku_rating_destinasi.plot()
plt.title('Rating Kota')
```



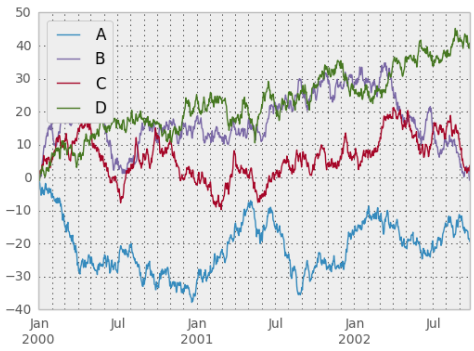
4. Visualisasi

- ***Matplotlib*** = Dalam proses visualisasi, *library matplotlib* merupakan *library* yang sangat terintegrasi saat baik dengan *library NumPy* dan *Pandas*.
- ***Seaborn*** = Dalam proses visualisasi, *library seaborn* merupakan *library* yang terintergrasi hanya dengan bingkai *data Pandas*.

5. *Data Frame* dan *Array*

- ***Matplotlib*** = *Library matplotlib* bekerja dengan bingkai dan larik *data* karena memiliki *API stateful* yang berbeda untuk merencanakan. Biasanya dalam pemanggilan angka dan objek lainnya diawali dengan *plot()* tanpa harus mengelola *parameter*.

```
df = DataFrame(randn(1000, 4), index=ts.index, columns=list('ABCD'))

df = df.cumsum()

plt.figure(); df.plot(); plt.legend(loc='best')
<matplotlib.legend.Legend at 0x12f2b590>
```



- ***Seaborn*** = *Library seaborn* secara keseluruhan lebih bekerja pada pengelolaan *dataset* dan jauh lebih intuitif daripada *Matplotlib*. Untuk *library seaborn*, *replot()* adalah *API entry* dengan *parameter* '*kind'* untuk menentukan jenis *plot* yang bisa berupa garis, batang, dan masih banyak lainnya. Bisa dikatakan bahwa *library seaborn* lebih cocok untuk menangani jenis *data* yang lebih banyak.

```
#Loading dataset harga rumah
dataku = pd.read_csv('reported.csv')
```

	Year	crimes.total	crimes.penal.code	crimes.person	murder	assault	sexual.offenses	rape	stealing.general	burglary	...	vehicle.theft	out.of.vehicle.theft
0	1950	2784	2306	120	1	105	40	5	1578	295	...	NaN	NaN
1	1951	3284	2754	125	1	109	45	6	1899	342	...	NaN	NaN
2	1952	3160	2608	119	1	104	39	4	1846	372	...	NaN	NaN
3	1953	2909	2689	119	1	105	45	5	1929	361	...	NaN	NaN
4	1954	3028	2791	126	1	107	41	5	1981	393	...	NaN	NaN
5	1955	3357	3101	135	1	118	44	5	2254	459	...	NaN	NaN
6	1956	3488	3215	133	1	116	38	5	2363	470	...	NaN	NaN
7	1957	3774	3520	133	1	116	36	5	2635	580	...	245.0	NaN
8	1958	4064	3791	127	1	113	40	6	2880	724	...	279.0	NaN
9	1959	4033	3733	125	1	110	47	6	2793	715	...	238.0	NaN

6. **Fleksibilitas**
   - ***Matplotlib*** **=** *Library matplotlib* lebih mudah disesuaikan atau dikustomisasi dengan bebas sesuai permintaan *user* nya.
   - ***Seaborn*** **=** *Library seaborn* lebih terstruktur dan lebih menyediakan tema *default* yang umum digunakan.

7. ***Problem solve***
   - ***Matplotlib*** **=** Biasanya *library Pandas* menggunakan *library matplotlib* untuk menjadi *cover* rapih disekitar area visualisasi yang telah dibentuk *matplotlib*.
   - ***Seaborn*** **=** *Library seaborn* dapat digunakan untuk penggunaan *data* yang lebih kompleks dan dapat menindaklanjuti atau memproses visualisasi terusan yang telah dibentuk oleh *matplotlib*.

Kedua *library* visualisasi populer yang digunakan dalam *python* memiliki perbedaan dalam kasus penggunaan, skalabilitas dan banyak hal lainnya. Berdasarkan pernyataan Analisa di atas, *user* harus mampu memilih alat visualisasi terbaik untuk pekerjaan tertentu. Jika seseorang ingin memvisualisasi *data* statistik maka *seaborn* adalah pilihan yang terbaik karena memiliki banyak kegunaan yang cocok untuk tugas statistik. Sedangkan visualisasi sederhana di luar kepentingan statistik yang tidak membutuhkan *datasets* bisa menggunakan *library Matplotlib* saja.