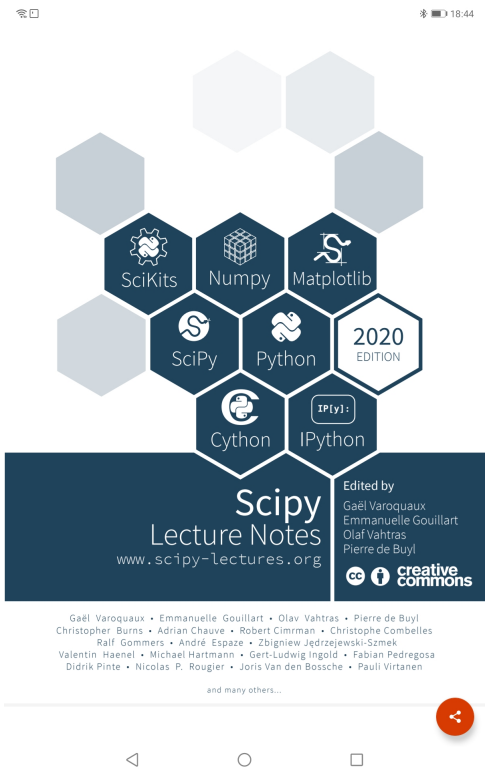

MATPLOTLIB PYTHON 3

Tugas Mata Kuliah SK5003 Pemrograman dalam Sains
Summary Matplotlib

Mohammad Rizka Fadhli (Ikang)
20921004@mahasiswa.itb.ac.id

25 September 2021

Contents

List of Figures

MATPLOTLIB

1 CHAPTER I

1.1 Pendahuluan

`matplotlib` adalah salah satu *package* visualisasi atau grafik yang paling populer di *Python* saat ini. Sedangkan `numpy` adalah *fundamental package for scientific computing* di *Python*.

Kombinasi keduanya sering digunakan untuk membuat visualisasi data atau keperluan matematis lainnya. Kali ini kita akan mengeksplorasi bagaimana proses membuat grafik di *Python*.

Contoh Sederhana Satu *Plot*

Sebagai contoh, jika kita hendak membuat grafik sederhana dari fungsi berikut ini:

$$f(x) = |\log x|, x \in (0, 10]$$

Langkah kerja menggunakan *Python* adalah sebagai berikut:

- **STEP 1** panggil *packages* `numpy` dan `matplotlib`.
- **STEP 2** kita akan generate $x \in (0, 10]$ menggunakan `numpy`. Untuk simplifikasi, saya akan buat batas x di $[0.002, 10]$.
- **STEP 3** hitung $f(x) = \log x$.
- **STEP 4** *plot* garis $x, f(x)$ menggunakan `matplotlib`.

Berikut adalah programnya:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y = abs(np.log(X))

# STEP 4
plt.plot(X, Y)
plt.show()
```

Jika kita *run* program tersebut, kita akan dapatkan grafik sebagai berikut:

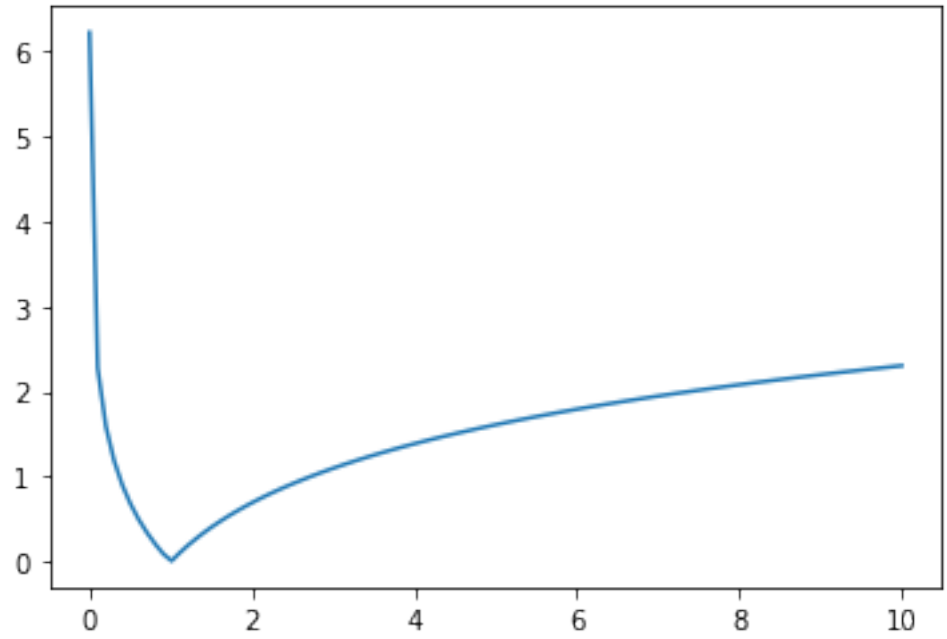

Figure 1: Grafik f(x)

plot di atas merupakan plot dengan *default setting* Python. Kelak akan ditunjukkan caranya agar kita bisa mengubah *settings* yang ada.

Contoh Sederhana Dua *Plots*

Kita juga bisa menggambar dua *plots* dalam satu *canvas* menggunakan `matplotlib`.

Misalkan kita hendak menggambar:

$$f(x) = |\log x| g(x) = -|\log x| x \in (0.002, 10]$$

Kita cukup mengulang langkah di atas **dengan syarat membuat dua plot sekaligus**.

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))
```

```
# STEP 4
plt.plot(X, Y1)
plt.plot(X, Y2)
plt.show()
```

Berikut adalah hasilnya:

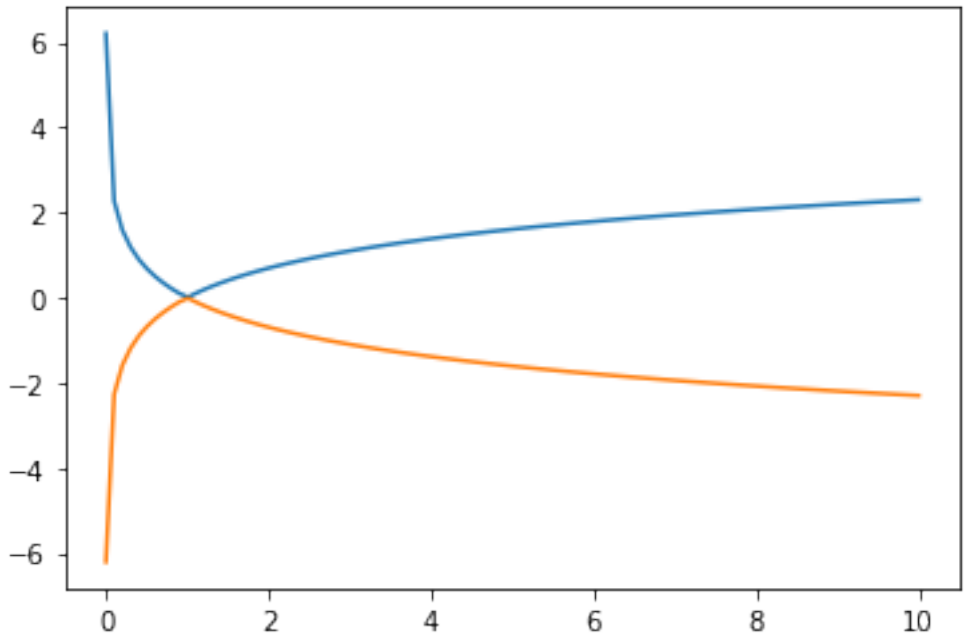

Figure 2: Grafik f(x) dan g(x)

1.2 *Save Plot*

Salah satu perintah yang sering digunakan adalah perintah *save* hasil grafik ke dalam format `.png` atau `.jpeg`. Meminjam gambar $f(x)$ dari bagian sebelumnya, kita tinggal menambahkan perintah *save* di akhir baris **STEP 4**.

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1)
plt.plot(X, Y2)
plt.show()
plt.savefig("dual_plot.png", dpi=101)
```

Kita bisa memasukkan nilai *dots per inch* ke dalam perintah tersebut. Semakin tinggi nilainya, semakin bagus gambar yang dihasilkan namun *filesize*-nya akan membesar.

Sebagai contoh, dengan konfigurasi seperti di atas, saya mendapatkan *file* `dual_plot.png` berukuran `2.4 kB (2,358 bytes)`.

1.3 Kustomisasi *Plot*

Karena bersifat *coding*, kita bisa melakukan banyak kustomisasi terhada grafik yang hendak ditampilkan. Misalkan kita hendak membuat grafik dari fungsi berikut:

$$f(x) = |x|, x \in [-3, 3]$$

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
x = np.linspace(-3, 3, 100)

# STEP 3
y = abs(x)

# STEP 4
plt.plot(x, y)
plt.show()
```

Berikut adalah hasilnya:

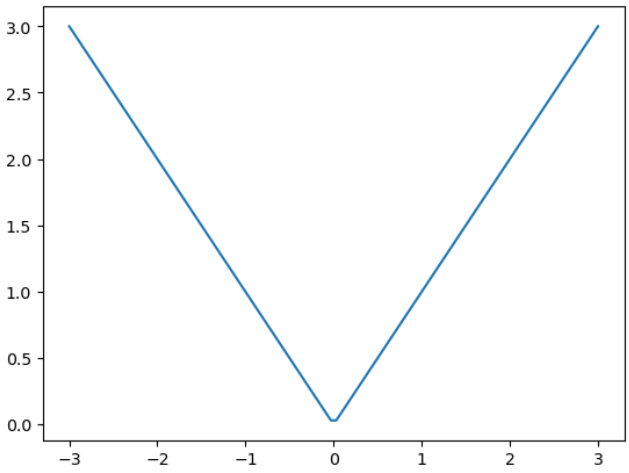

Figure 3: Grafik |x|

1.3.1 *Ratio*

Seandainya kita hendak mengubah *ratio* gambar dan tingkat kualitas gambar yang dihasilkan, kita bisa menambahkan perintah berikut di **STEP 4**:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
x = np.linspace(-3, 3, 100)

# STEP 3
y = abs(x)

# STEP 4
plt.figure(figsize=(16, 7), dpi=80)
plt.plot(x, y)
plt.show()
```

Kita akan mendapat gambar yang lebih **lebar** secara *ratio*.

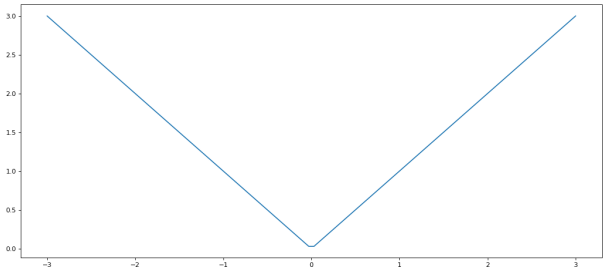

Figure 4: Grafik |x|

1.3.2 Memberikan Warna Secara Manual

Kita lihat kembali contoh berikut:

$$f(x) = |\log x| g(x) = -|\log x| x \in (0.002, 10]$$

Jika kita hendak memberikan warna `biru` untuk $f(x)$ dan `merah` untuk $g(x)$, berikut perintahnya:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--")
plt.show()
```

Kita juga bisa mengubah **ketebalan garis** dan *linestyle* pada perintah tersebut.

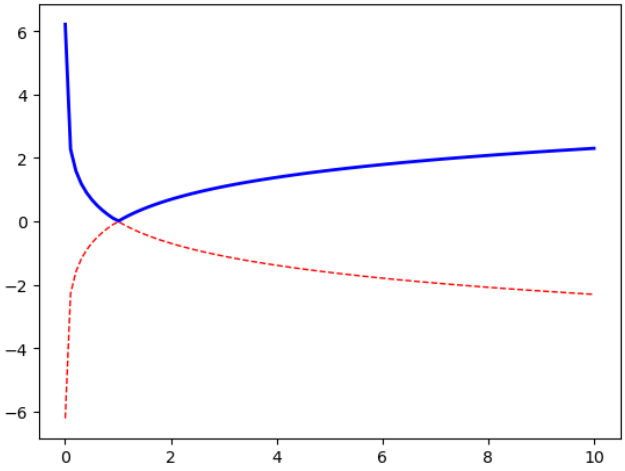

Figure 5: Grafik Ganti Warna Garis

1.3.3 Memberikan Batas Grafik

Dari grafik sebelumnya, jika kita hendak membuat batas sumbu y hanya boleh berada di selang $[-1, 1]$, maka kita tuliskan:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--")
plt.ylim(-1.0, 1.0)
plt.show()
```

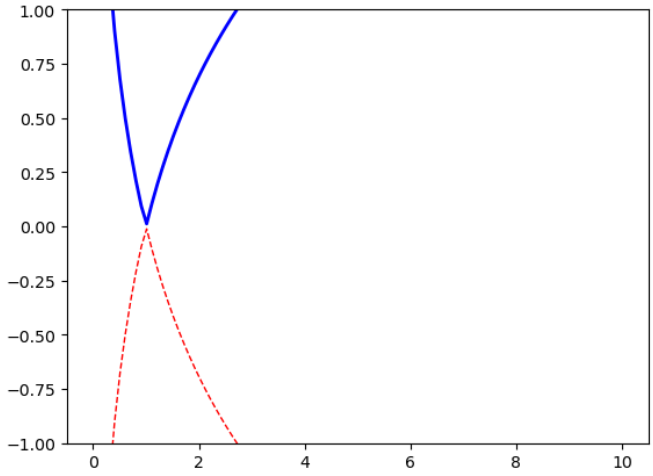

Figure 6: Grafik Ganti Batas sb y

Hal serupa jika kita hendak membatasi sumbu x di selang $[0.5, 1.5]$:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--")
plt.xlim(0.5,1.5)
plt.show()
```

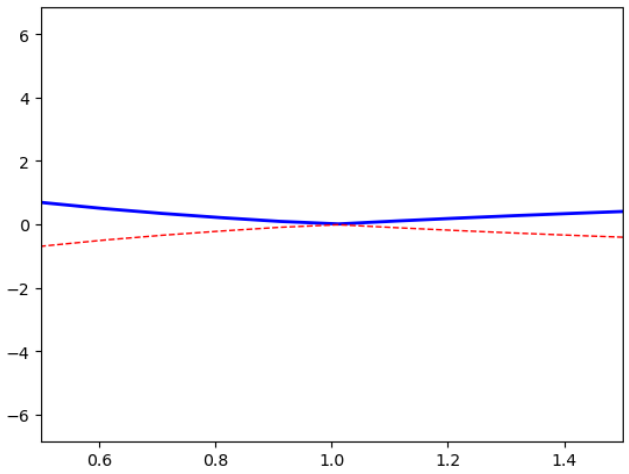

Figure 7: Grafik Ganti Batas sb x

1.3.4 Mengubah Label Sumbu x dan Sumbu y

Misalkan kita hendak melakukan *highlight* dengan cara menuliskan label sumbu x atau y secara detail, kita bisa lakukan

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--")
plt.xticks(np.linspace(5, 10, 5))
plt.show()
```

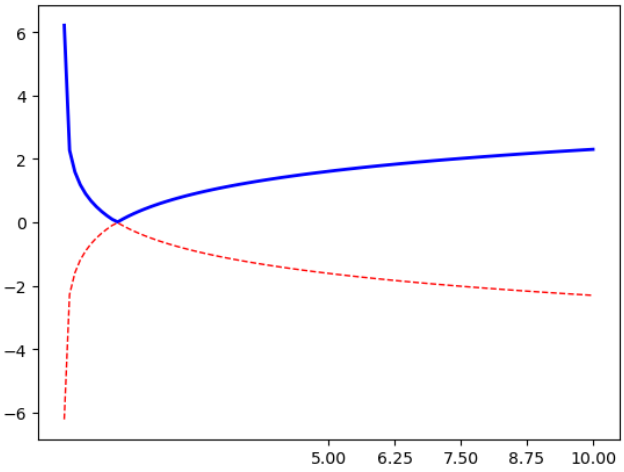

Figure 8: Grafik Labelling sb x

Selain itu, kita juga bisa mengubah bagaimana `matplotlib` menampilkan label sumbu. Misalkan sebagai berikut:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--")

# kustomisasi label di sumbu x
plt.xticks([1,4,8],
           [r'$2/2$', r'$8/2$', r'$16/2$'])

# kustomisasi label di sumbu y
plt.yticks([2,0,-2],
           [r'$4/2$', r'$y = 0$', r'$-4/2$'])

plt.show()
```

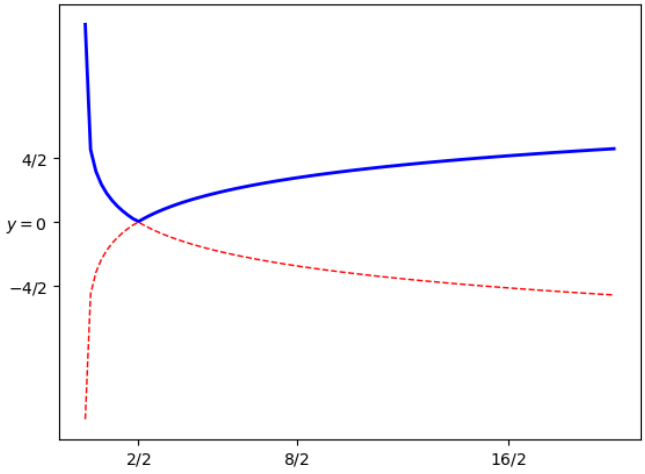

Figure 9: Grafik Re-labelling sb x dan sb y

1.3.5 Menambah *Legend*

Salah satu elemen penting dalam grafik adalah *legend*. Kita bisa menambahkan *legend* dengan `matplotlib` setelah kita memberikan label pada masing-masing grafik.

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
Y1 = abs(np.log(X))
Y2 = - abs(np.log(X))

# STEP 4
# menambahkan label di masing-masing plot
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-",
         label = "|log(x)|")
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--",
         label = "- |log(x)|")
# menambahkan legend
plt.legend(loc='upper right')

plt.show()
```

Kita bisa mengatur di mana letak *legend* berada. Sebagai contoh, saya meletakkan *legend* di kanan atas.

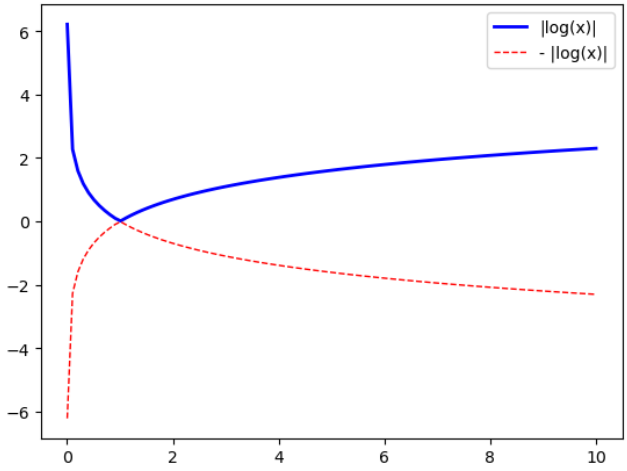

Figure 10: Menambah Legend Grafik

1.3.6 *Annotations*

Pada grafik di atas, kita bisa menambahkan beberapa titik dan label tertentu. Misalkan saya hendak menambahkan titik $x = 2$ pada $f(x)$ dan $g(x)$.

$f(2) = 0.6931472$ dan $g(2) = -0.6931472$

Maka langkahnya adalah sebagai berikut:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
X = np.linspace(0.002, 10, 100)

# STEP 3
# menghitung f(2) dan g(2)
x_i = 2
f_x = abs(np.log(x_i))
g_x = - abs(np.log(x_i))

# STEP 3
# menghitung f(x) untuk setiap x
```

```
Y1 = abs(np.log(X))

# menggambar f(x)
plt.plot(X, Y1, color="blue", linewidth=2.0, linestyle="-",
         label = "|log(x)|")

# menambahkan titik (x,f(x)) pada saat x = 2
plt.scatter([x_i, ], [f_x, ], 50, color='black')

# memberikan annotation label di titik (x,f(x)) pada saat x = 2
plt.annotate(r'$| \log{2} | \sim 0.69$', # label yang hendak kita tuliskan
                     # ditulis dalam bentuk LATEX
             xy=(x_i, f_x), # koordinat letak target
             xycoords='data',
             xytext=(20, 30), # offset penulisan label
                         # didefinisikan mau berapa jarak x dan y dari target
             textcoords='offset points',
             fontsize=10, # fontsize
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) # arrow

# STEP 4
# menghitung g(x) untuk setiap x
Y2 = - abs(np.log(X))

# menggambar g(x)
plt.plot(X, Y2, color="red", linewidth=1.0, linestyle="--",
         label = "- |log(x)|")

# menambahkan titik (x,g(x)) pada saat x = 2
plt.scatter([x_i, ], [g_x, ], 50, color='black')

# memberikan annotation label di titik (x,g(x)) pada saat x = 2
plt.annotate(r'$- | \log{2} | \sim -0.69$', # label yang hendak kita tuliskan
                     # ditulis dalam bentuk LATEX
             xy=(x_i, g_x), # koordinat letak target
             xycoords='data',
             xytext=(-20, -30), # offset penulisan label
                         # didefinisikan mau berapa jarak x dan y dari target
             textcoords='offset points',
             fontsize=10, # fontsize
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) # arrow


# STEP 5
# menambahkan legend
```

```
plt.legend(loc='upper right')

plt.show()
```

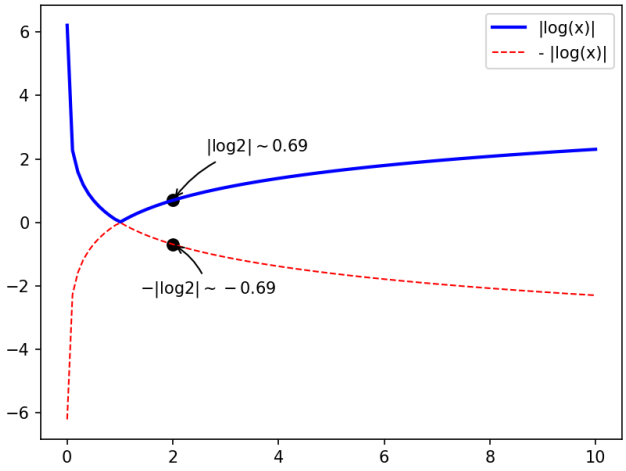

Figure 11: Menambah Annotation

1.3.7 *Fill* Warna di Bawah Kurva

Kita bisa menambahkan warna di bawah kurva sesuai dengan kebutuhan. Misalkan kita hendak menggambar:

$$f(x) = \sin x, x \in [-\pi, \pi]$$

Lalu memberi warna "merah" untuk area di bawah kurva $f(x)$. Maka:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
n = 100
X = np.linspace(-np.pi, np.pi, n)

# STEP 3
Y = np.sin(2 * X)

# STEP 4
# menggambar plot
plt.plot(X, Y, color='blue', alpha=1.00)
# memberikan warna
plt.fill_between(X,Y,
                 color = "red", # warna merah
                 alpha = .4) # alpha adalah tingkat transparansi warna
plt.show()
```

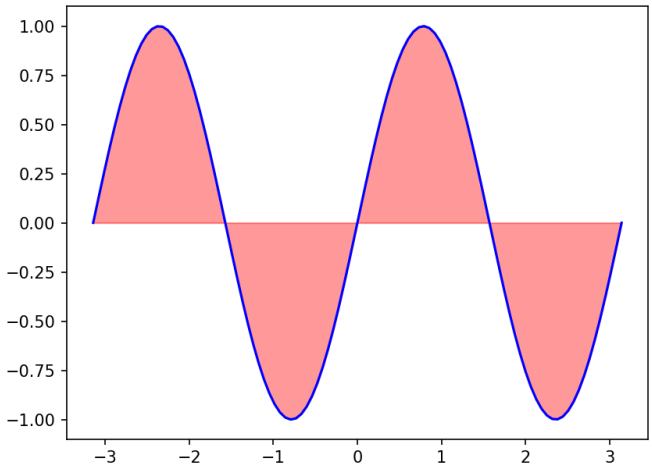

Figure 12: Memberi Warna di Bawah Kurva

BENTUK *PLOTS* LAIN

2 CHAPTER II

2.1 *Scatter Plot*

Scatter plot adalah grafik yang memasangkan titik (x, y) dalam bidang kartesian.

Misalkan saya memiliki data berat badan dan tinggi badan sebagai berikut:

Table 1: Tabel data berat dan tinggi badan

berat	tinggi	bmi
62	178	19.6
54	172	18.3
76	150	33.8
67	160	26.2
63	143	30.8
77	155	32.0
79	147	36.6
57	140	29.1
65	164	24.2
52	171	17.8
64	154	27.0
61	142	30.3
60	174	19.8
55	152	23.8
50	180	15.4
51	169	17.9
75	163	28.2
66	148	30.1
58	162	22.1
68	145	32.3

Kita hendak membuat *scatter plot*-nya sebagai berikut:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
# memasukkan data berat dan tinggi
berat = np.array([62,54,76,67,63,77,79,57,65,52,
```

```
          64,61,60,55,50,51,75,66,58,68])
tinggi = np.array([178,172,150,160,143,155,147,
           140,164,171,154,142,174,152,
           180,169,163,148,162,145])
# menghitung bmi
bmi = berat/(tinggi / 100)**2
# mengelompokkan bmi
kategori = bmi<=20

# STEP 3
# membuat scatter plot
# besar kecilnya titik ditentukan oleh bmi
# warna titik ditentukan oleh kategori bmi
plt.scatter(berat,tinggi,
        s = bmi*2,
        c = kategori)

plt.show()
```

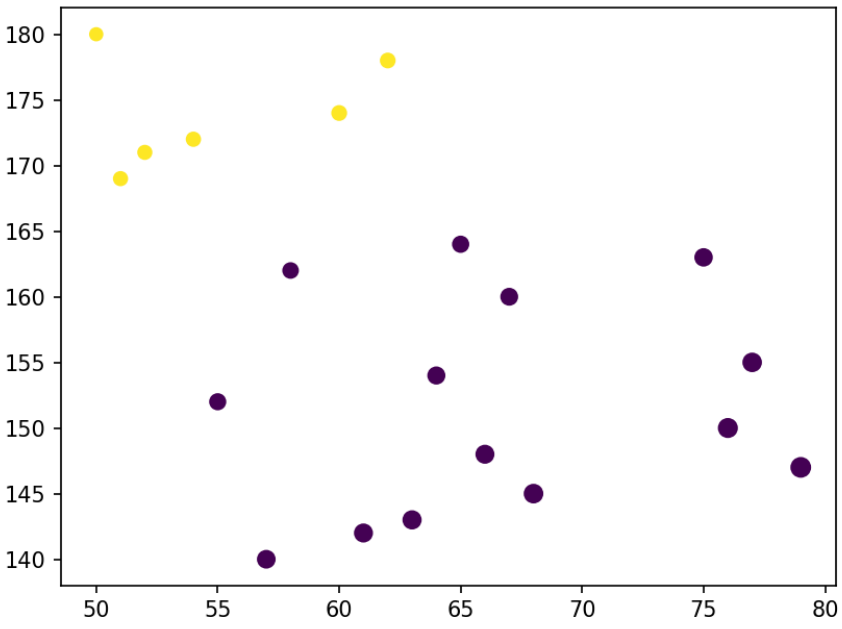

Figure 13: Scatter plot

2.2 *Bar Plot*

Bentuk *plot* lain yang sering digunakan adalah diagram batang. Misalkan saya memiliki data sebagai berikut:

Table 2: Tabel data berat dan tinggi badan

berat	tinggi	bmi	kategori
62	178	19.56824	2
54	172	18.25311	1
76	150	33.77778	4
67	160	26.17188	3
63	143	30.80835	4
77	155	32.04995	4
79	147	36.55884	4
57	140	29.08163	4
65	164	24.16716	2
52	171	17.78325	1
64	154	26.98600	3
61	142	30.25193	4
60	174	19.81768	2
55	152	23.80540	2
50	180	15.43210	1
51	169	17.85652	1
75	163	28.22839	3
66	148	30.13148	4
58	162	22.10029	2
68	145	32.34245	4

Kemudian saya kelompokkan menjadi:

Table 3: Pengelompokkan BMI

kategori	n
1	4
2	5
3	3
4	8

Bentuk diagram batangnya adalah sebagai berikut:

```
# STEP 1
import numpy as np
```

```
import matplotlib.pyplot as plt

# STEP 2
# memasukkan data kategori berat dan tinggi
kat = np.array([1,2,3,4])
# memasukkan banyak orang
banyak_org = np.array([4,5,3,8])

# STEP 3
# menggambar barplot
plt.bar(kat, banyak_org,
    facecolor='#226fb3', # kode hex warna bisa dicari dengan colorpicker
    edgecolor='white')
# mengganti label dengan arti dari kategori bmi
plt.xticks([1,2,3,4],
       ['under','normal','over','obese'])

plt.show()
```

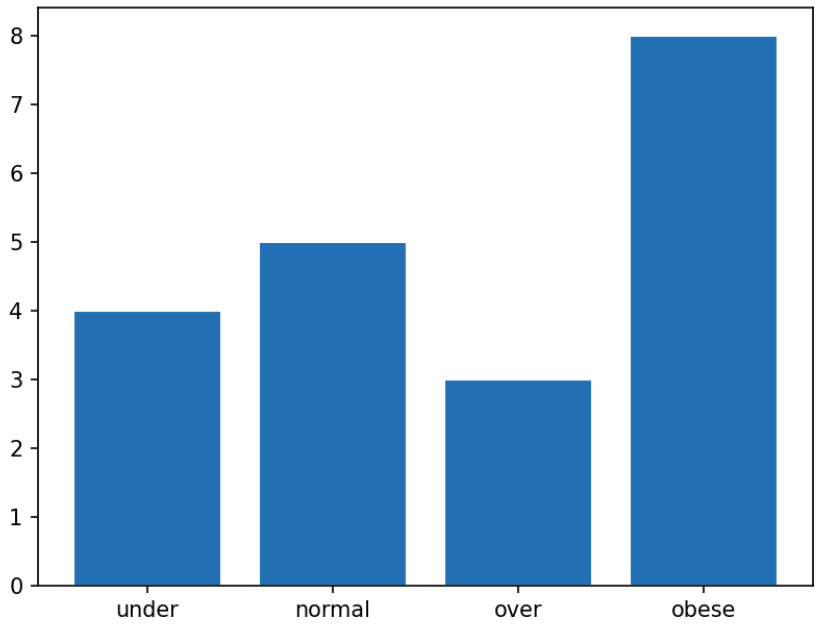

Figure 14: Bar plot

2.3 *Pie Charts*

Bentuk diagram batang di atas, bisa kita modifikasi menjadi bentuk *pie chart* sebagai berikut:

```
# STEP 1
import matplotlib.pyplot as plt

# STEP 2
# memasukkan data kategori berat dan tinggi
kat = ['under','normal','over','obese']
# memasukkan banyak orang
banyak_org = [4,5,3,8]

# STEP 3
# menggambar pie chart
plt.pie(banyak_org,
    labels = kat)

plt.show()
```

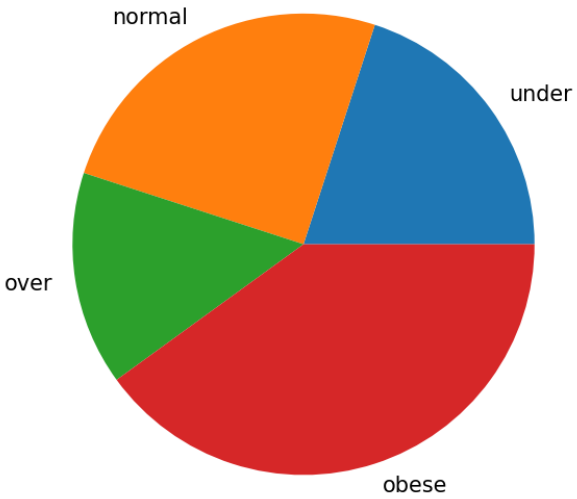

Figure 15: Pie Chart

2.4 *Boxplot*

Boxplot biasa digunakan untuk melihat persebaran data. Mari kita lihat kembali persebaran tinggi badan:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
# memasukkan data tinggi badan
tinggi_bdn = np.array([178,172,150,160,143,155,147,140,
                       164,171,154,142,174,152,180,169,163,
                       148,162,145])

# STEP 3
# menggambar pie chart
plt.boxplot(tinggi_bdn)
# mengganti label sb x
plt.xticks([1],
       ['Tinggi Badan'])
plt.show()
```

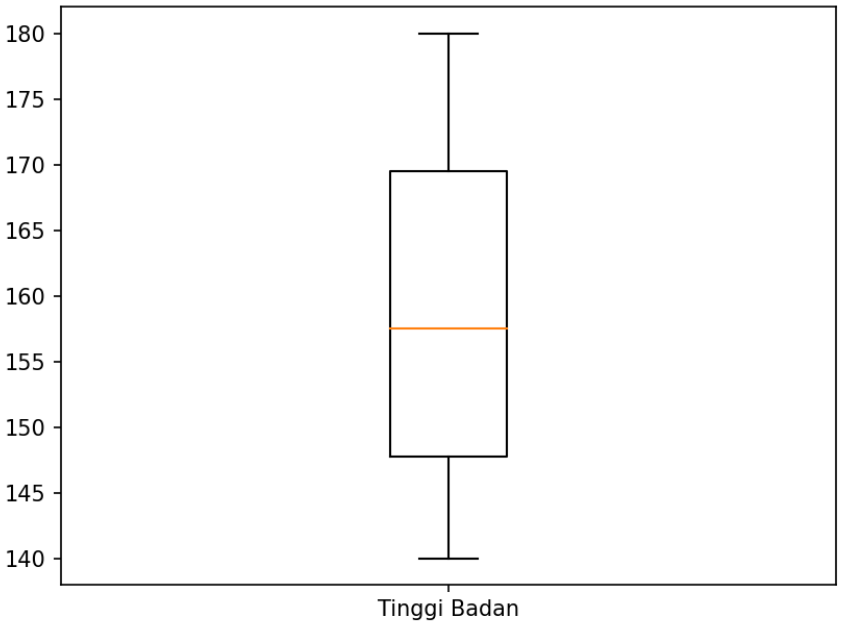

Figure 16: Boxplot

2.5 *Contour Plot*

Contour plot adalah membuat grafik 3D ke dalam bidang 2D. Sebagai contoh, saya akan membuat grafik dari fungsi *body mass index* dari contoh sebelumnya.

$$bmi = \frac{bb}{tb^2}$$

Nanti saya akan *plot* dengan `berat badan` sebagai sumbu x dan `tinggi badan` seagai sumbu y.

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt

# STEP 2
# membuat fungsi bmi
def bmi(berat,tinggi):
  return berat / (tinggi/100)**2

# STEP 3
# memasukkan semua berat dan tinggi yang mungkin
n = 1000
berat = np.linspace(0, 200, n)
tinggi = np.linspace(80, 250, n)

# STEP 4
# berat menjadi sumbu x
# tingg menjadi sumbu y
X, Y = np.meshgrid(berat, tinggi)

# membuat grafik 3d dalam 2d
plt.contourf(X, Y, bmi(X, Y), 8, alpha=.75, cmap=plt.cm.hot)
C = plt.contour(X, Y, bmi(X, Y), 8, colors='black')
plt.clabel(C, inline=1, fontsize=6)

plt.show()
```

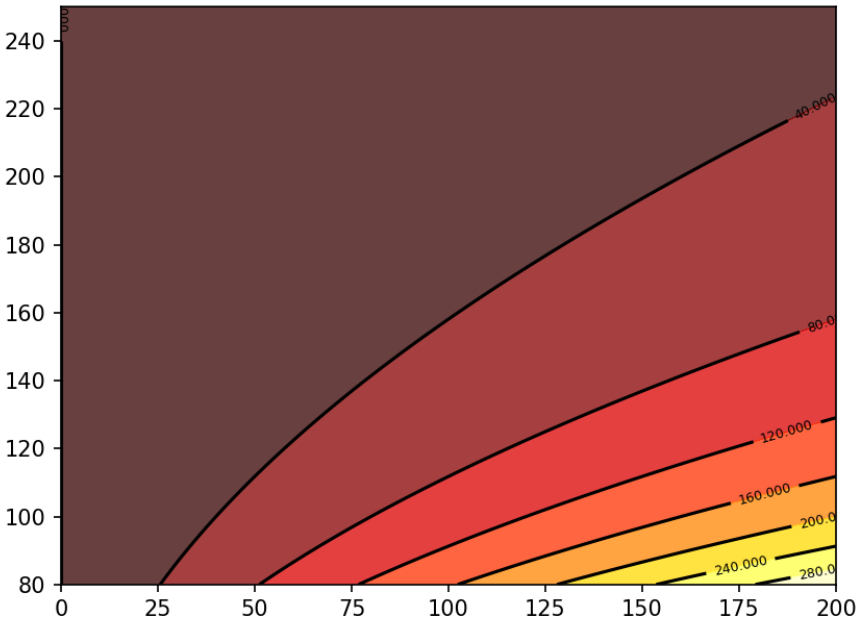

Figure 17: Contour Plot

2.6 3D Plot

Bentuk dari *contour plot* `bmi` di atas, bisa kita gambarkan dalam bentuk aslinya di ruang 3D sebagai berikut:

```
# STEP 1
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# STEP 2
# membuat fungsi bmi
def bmi(berat,tinggi):
  return berat / (tinggi/100)**2

# STEP 3
# membuat axis ruang 3D
fig = plt.figure()
ax = Axes3D(fig)

# STEP 4
# memasukkan semua berat dan tinggi yang mungkin
n = 1000
berat = np.linspace(0, 200, n)
tinggi = np.linspace(80, 250, n)

# STEP 5
# berat menjadi sumbu x
# tinggi menjadi sumbu y
X, Y = np.meshgrid(berat, tinggi)
# bmi sebagai sumbu z
Z = bmi(X,Y)

# membuat grafik 3d
ax.plot_surface(X, Y, Z, cmap='hot')

plt.show()
```

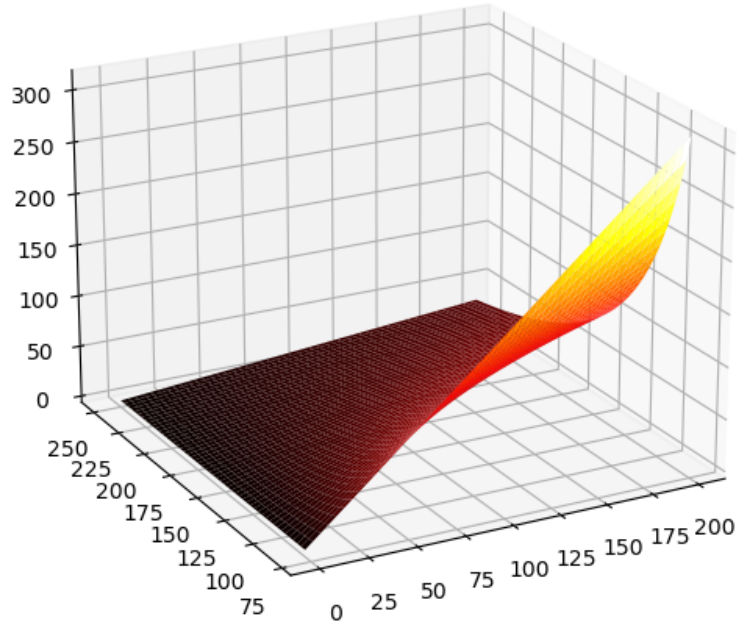

Figure 18: 3D Plot

== End ==