VERSION 2.0

AGUSTUS , 2023

[PRAKTIKUM PEMROG. FUNGSIONAL]

MODUL 5 – Implementasi Pemrograman Fungsional dalam Studi Kasus Statistika untuk Visualisasi Data.

DISUSUN OLEH :
Fildzah Lathifah

Hania Pratiwi Ningrum

DIAUDIT OLEH
Fera Putri Ayu L., S.Kom., M.T.

PRESENTED BY: TIM LAB-IT UNIVERSITAS MUHAMMADIYAH MALANG

PERSIAPAN MATERI

Praktikan diharapkan telah menguasai materi dari modul-modul sebelumnya.

TUJUAN PRAKTIKUM

1. Praktikan diharap dapat memahami konsep dasar Probability Density Function dalam konteks statistik dan probabilitas.
2. Praktikan diharap dapat memahami dan menggunakan pustaka matematika atau statistik untuk menghitung dan memanipulasi PDF.

TARGET MODUL

Penguasaan materi:

1. Library pada python
2. Matplotlib
3. Numpy
4. Probability Density Function

PERSIAPAN SOFTWARE/APLIKASI

- Komputer/Laptop
- Sistem operasi Windows/Linux/Mac OS/Android
- Pycharm/Google Collab/ Jupyter Notebook

MATERI POKOK

Karena hampir semua materi pada praktikum pemrograman fungsional langsung diimplementasikan dalam *source code*, maka semua materi bisa anda akses pada *Google Collab* melalui tautan ini.

Library pada python

Di modul 5 ini kita akan mulai menggunakan berbagai pustaka (library) yang berkaitan dengan statistika dan visualisasi data, yang akan membantu dalam mengolah dan menampilkan informasi secara efektif.

Python adalah bahasa pemrograman yang mempunyai banyak library. Library digunakan untuk membantu dalam mengolah atau mengerjakan task.

Python mempunyai library yang sudah built-in artinya library – library tersebut sudah siap digunakan setelah python telah terinstall. Python mempunyai banyak built-in library yang bisa anda baca di dokumentasinya.

Library Python ini akan membawahi berbagai fungsi yang ada. Sehingga ketika kita ingin menggunakan sebuah fungsi, maka kita harus memanggil library yang membawahinya terlebih dahulu. Library ini dapat diibaratkan sebagai rumah, jika kita ingin bertemu dengan seseorang maka kita harus menuju rumahnya terlebih dahulu.

Cara memanggil library, salah satunya adalah dengan sintaks import. Cara lain bisa kalian pelajari disini.

```
[ ]  import math

     math.sqrt(36)

     6.0
```

Selain built-in library ada juga library lainnya yang banyak digunakan untuk mengolah data di python. Sebagian besar library (non build-in), dibangun dan dibuat oleh pribadi maupun institusi/komunitas sehingga perlu ditambahkan (install) terlebih dahulu. Jika dibutuhkan, kalian bisa mengunjungi link berikut untuk cara instalasi.

Dalam bidang data science, adanya library membuat pemrograman python menjadi lebih sederhana dan nyaman bagi programmer karena tidak perlu menulis kode yang sama berulang kali untuk program yang berbeda. Library python memainkan peran yang sangat penting dalam bidang pembelajaran mesin, data science, visualisasi data, aplikasi manipulasi gambar dan data, dan masih banyak lagi.

Dalam modul ini, kita akan mulai menggunakan beberapa library python yang secara khusus dikembangkan untuk analisis statistika dan visualisasi data. Ada beberapa library Python diantaranya adalah matplotlib, numpy, seaborn, pandas, dll. Untuk lebih lengkapnya bisa cek di link berikut.

1. MATPLOTLIB

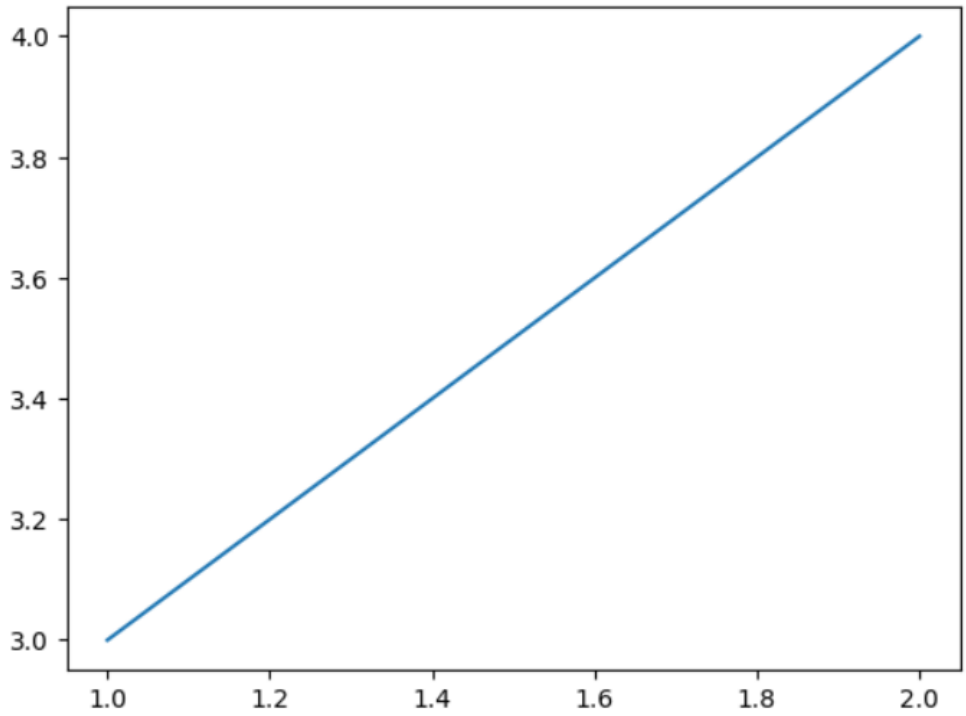
Matplotlib adalah pustaka visualisasi data populer dalam bahasa pemrograman Python. Ini

memungkinkan kita membuat berbagai jenis grafik dan plot untuk mewakili data. Matplotlib dapat digunakan untuk memvisualisasikan distribusi probabilitas secara grafis. Sebagian besar utilitas Matplotlib terletak di bawah pyplotsubmodule, dan biasanya diimpor dengan plt alias:

```
[ ]  import matplotlib.pyplot as plt
```

Sekarang paket Pyplot dapat disebut sebagai plt dan dapat digunakan untuk menggambar sebuah garis seperti berikut:

```
[ ]  plt.plot([1,2], [3,4])
     plt.show()
```



Fungsi plot() digunakan untuk membuat garis atau kurva berdasarkan koordinat yang diberikan. Pada sumbu X, kita memiliki [1, 2], yang berarti titik pertama memiliki koordinat X = 1 dan titik kedua memiliki koordinat X = 2. Pada sumbu Y, kita memiliki [3, 4], yang berarti titik pertama memiliki koordinat Y = 3 dan titik kedua memiliki koordinat Y = 4. Jadi, garis atau kurva akan menghubungkan titik (1, 3) dan (2, 4).

Selanjutnya fungsi show() akan mengaktifkan tampilan plot interaktif, sehingga kita dapat melihat plot yang telah dibuat.

2. NumPy

NumPy (Numerical Python) adalah sebuah library pada bahasa pemrograman Python yang digunakan untuk komputasi numerik. NumPy menyediakan objek array multidimensional (ndarray) yang efisien dan fleksibel, serta berbagai fungsi matematika yang dapat digunakan untuk melakukan operasi pada array tersebut. NumPy merupakan salah satu library dasar

dalam ekosistem komputasi ilmiah Python, dan sering digunakan dalam analisis data, visualisasi, pemrosesan citra, dan berbagai aplikasi ilmiah lainnya.

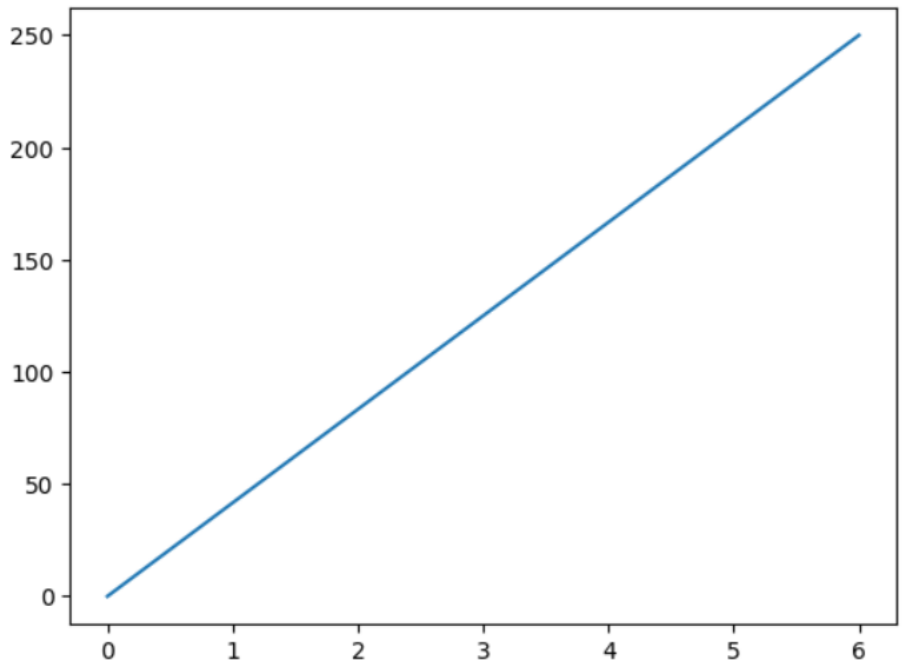
NumPy juga sering digunakan dalam visualisasi data, terutama ketika digunakan bersama dengan library lain seperti Matplotlib, Seaborn, atau Plotly. NumPy menyediakan array yang kompatibel dengan banyak library visualisasi, sehingga memudahkan untuk memasukkan data dalam bentuk yang sesuai. Untuk penjelasan lebih lanjut dan dokumentasi resmi tentang NumPy, kalian dapat mengunjungi situs web resmi NumPy di link berikut.

Berikut adalah contoh penggunaan NumPy dalam visualisasi data menggunakan Matplotlib:

Percobaan 1

Gambar garis dalam diagram dari posisi (0,0) ke posisi (6,25):

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

     xpoints = np.array([0, 6])
     ypoints = np.array([0, 250])

     plt.plot(xpoints, ypoints)
     plt.show()
```



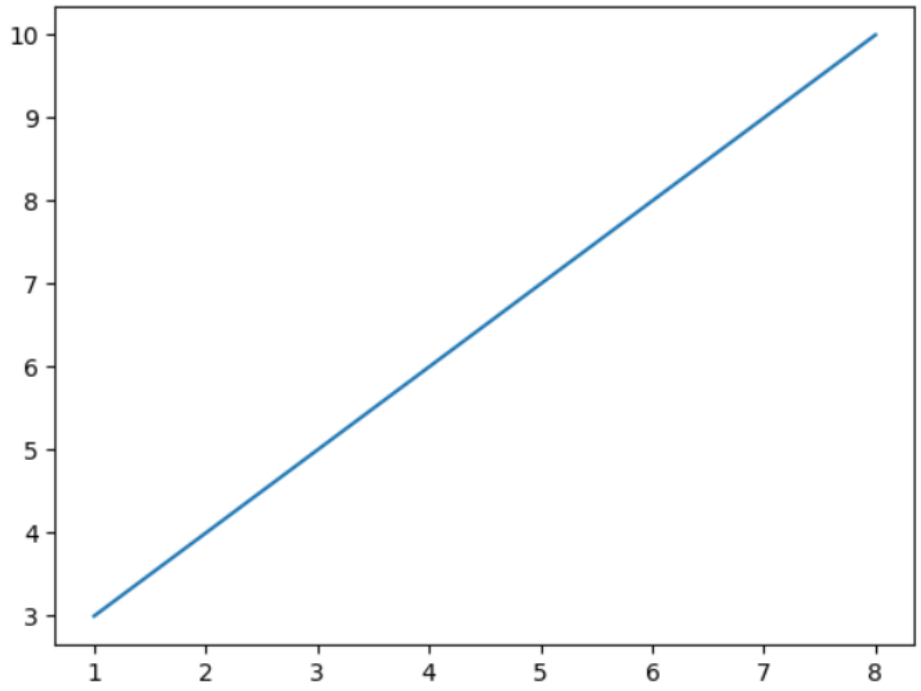
Plotting titik x dan y

Fungsi plot() ini digunakan untuk menggambar titik (marker) pada diagram. Secara default, fungsi plot() menarik garis dari titik ke titik. Fungsi mengambil parameter untuk menentukan titik dalam diagram. Parameter 1 adalah larik yang berisi titik-titik pada sumbu x. Parameter 2 adalah larik yang berisi titik-titik pada sumbu y. Jika kita perlu memplot garis dari (1,3) ke

(8,10) kita harus meneruskan dua larik [1, 8] dan [3, 10] ke fungsi plot.

Sekarang coba kalian modifikasi kode percobaan 1 diatas untuk menggambar garis dari (1, 3) ke (8, 10)!

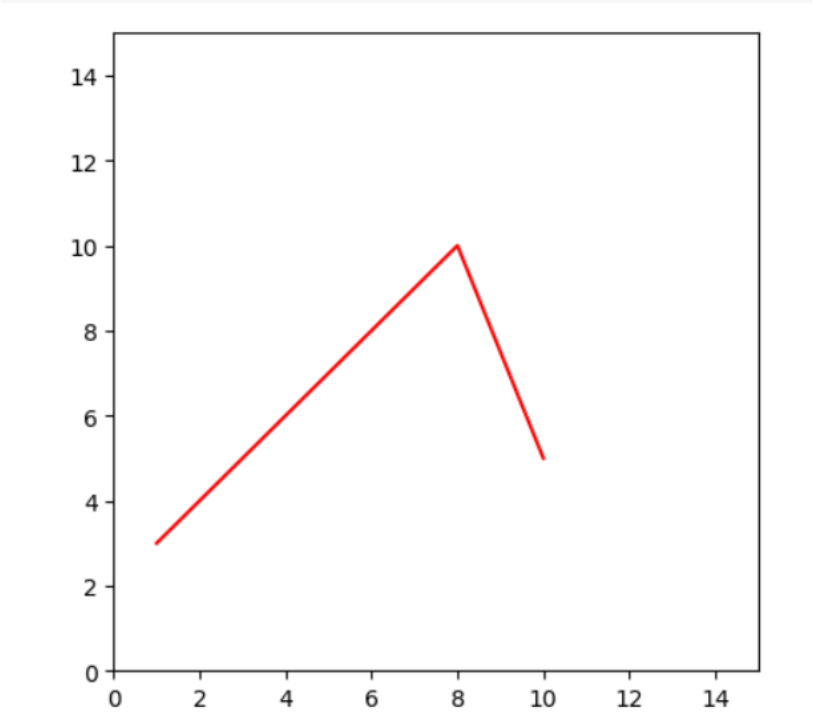
```
[ ]  # lakukan modifikasi pada kode percobaan 1
     # atau kalian bisa mengetikkannya kembali disini
     # tunjukkan pada asisten bagaimana kalian mengubah titik garis secara LANGSUNG
     # outputnya akan seperti ini:
```



Percobaan 2

Gambar garis dalam diagram dengan beberapa variasi tambahan:

```
[ ]  #pastikan kalian sudah melakukan import library yang dibutuhkan
     #sebelum menjalankan kode ini

     xpoints = np.array([1, 8, 10])
     ypoints = np.array([3, 10, 5])

     plt.figure(figsize=(5,5))
     plt.plot(xpoints, ypoints, color='red')
     plt.xlim([0,15])
     plt.ylim([0,15])
     plt.show()
```



Berbeda dengan percobaan 1, kali ini kita menambahkan perintah untuk set figur size, color, xlim, dan ylim. Cara tersebut dilakukan sebagai variasi tambahan dari sebuah grafik.

Beberapa variasi tambahan

Dalam grafik, variasi tambahan digunakan untuk memperbaiki dan memperkaya tampilan grafik, membuatnya lebih informatif dan mudah dipahami. Berikut adalah beberapa variasi tambahan umum yang dapat diterapkan pada grafik:

1. Label Axis (Label pada Sumbu): Menambahkan label pada sumbu X dan Y untuk memberikan informasi tentang data yang diplot. Label ini membantu pembaca untuk memahami apa yang direpresentasikan oleh setiap sumbu.
2. Judul Plot: Menambahkan judul pada grafik untuk memberikan konteks atau informasi tambahan tentang apa yang ditampilkan dalam grafik.
3. Legenda: Ketika ada beberapa garis atau elemen dalam satu plot, legenda memberikan penjelasan tentang apa yang setiap garis atau elemen wakili. Ini sangat berguna dalam kasus plot dengan beberapa seri data.
4. Warna dan Gaya Garis: Anda dapat mengubah warna dan gaya garis untuk membedakan berbagai garis dalam plot. Ini membantu dalam membedakan data ketika ada lebih dari satu seri.
5. Titik Data (Markers): Menambahkan titik, bentuk, atau simbol pada titik data dalam plot. Ini membantu menunjukkan posisi data dengan jelas pada plot.

Untuk lebih lengkapnya kalian bisa mempelajarinya di link berikut.

Setelah memahami materi diatas, silahkan kalian bisa berkreasi dengan memodifikasi grafik percobaan 2 sekreatif mungkin!

```
[ ]  # modifikasi percobaan 2 untuk menambahkan beberapa variasi
     # untuk melengkapi grafik agar lebih informatif dan mudah dipahami
     # tunjukkan pada asisten bagaimana kalian menambahkan variasi tertentu
     # dan dapat mengubah nya secara LANGSUNG
```

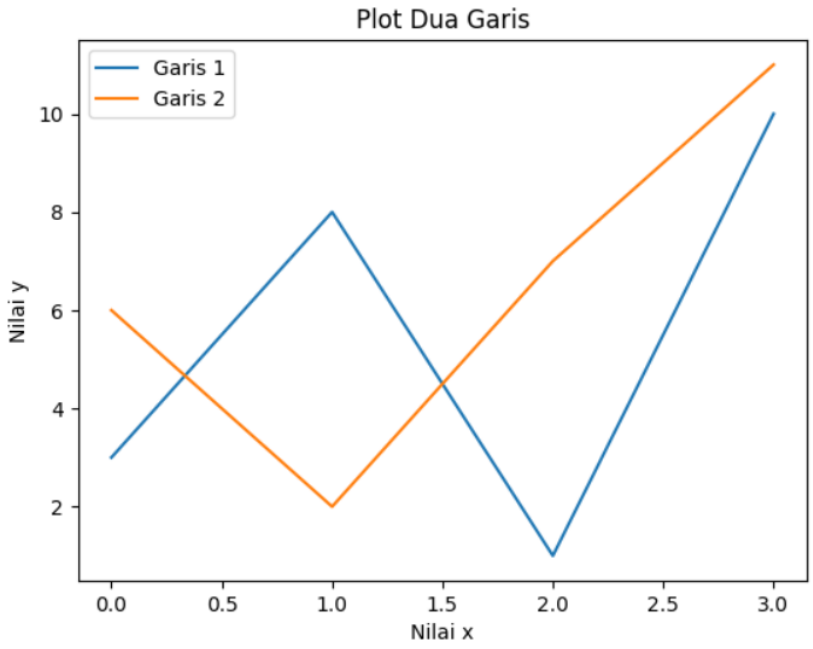
Percobaan 3

Multiple Lines

Kita dapat memplot garis sebanyak yang kita suka hanya dengan menambahkan lebih banyak fungsi plt.plot() sebagai berikut:

Gambar dua garis dengan menentukan plt.plot()fungsi untuk setiap garis!

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

     y1 = np.array([3, 8, 1, 10])
     y2 = np.array([6, 2, 7, 11])

     # Membuat plot untuk garis y1 dan y2
     plt.plot(y1, label='Garis 1')
     plt.plot(y2, label='Garis 2')

     # Menambahkan judul dan label sumbu
     plt.title('Plot Dua Garis')
     plt.xlabel('Nilai x')
     plt.ylabel('Nilai y')

     # Menambahkan keterangan (legend)
     plt.legend()
     plt.show()
```



Sekarang coba kalian modifikasi kode percobaan 3 diatas untuk menambahkan sebuah garis lagi dengan warna dan corak/bentuk yang berbeda!

```
[ ]  # modifikasi percobaan 3 untuk menggambar 3/lebih garis
     # dengan corak garis yang berbeda-beda.
     # tunjukkan pada asisten bagaimana kalian menambahkan garis
     # dan mengubah corak garis secara LANGSUNG
```

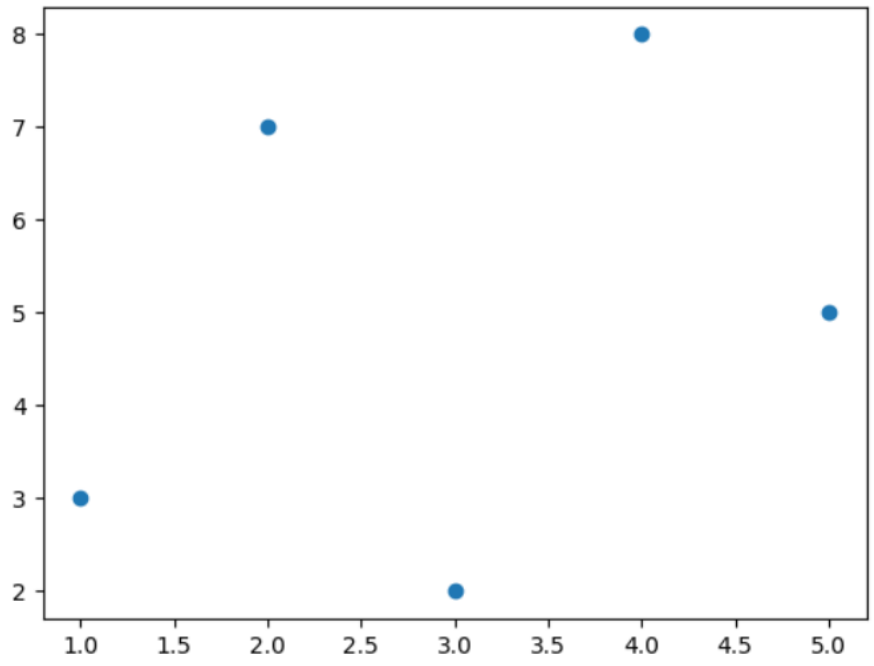
Percobaan 4

Scatter plot

Selain menggunakan pustaka pyplot dalam matplotlib untuk membuat plot garis dan grafik seperti pada percobaan-percobaan sebelumnya, kita juga dapat menggunakan visualisasi jenis lain seperti scatter plot.

Scatter plot adalah salah satu jenis visualisasi data yang digunakan untuk menggambarkan hubungan antara dua variabel. Scatter plot memungkinkan kita untuk memahami apakah terdapat korelasi, pola, atau hubungan lain di antara data yang diamati. Scatter plot terutama efektif dalam menggambarkan data numerik dan membantu mengidentifikasi tren atau anomali yang mungkin tidak terlihat dari data mentah.

```
[ ]  #pastikan kalian sudah melakukan import library dengan plt alias
     #sebelum menjalankan kode ini

     x = [1, 2, 3, 4, 5]
     y = [3, 7, 2, 8, 5]

     plt.scatter(x, y)
     plt.show()
```



Dalam kode ini, kita memiliki dua array, x dan y. Masing-masing array berisi pasangan nilai x dan y untuk setiap titik yang akan ditampilkan pada plot. Seperti percobaan sebelumnya, di percobaan kali ini scatter plot juga dapat melakukan variasi lebih lanjut, seperti mengubah warna, ukuran, atau bentuk titik-titik, serta menambahkan elemen lain seperti label sumbu, judul plot, dan sebagainya.

Sekarang coba kalian modifikasi kode percobaan 4 diatas untuk menambahkan beberapa variasi baru sesuai kreativitas kalian!

```
[ ]  # modifikasi percobaan 4 dengan menambah variasi baru
     # tunjukkan pada asisten variasi apa saja yang kalian buat
```

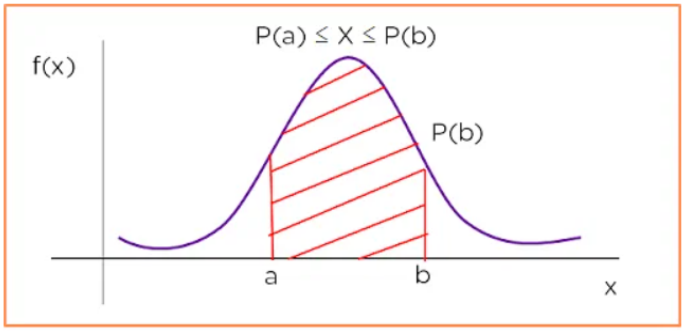
3. Probability Density Function

Fungsi yang menentukan hubungan antara variabel acak dan probabilitasnya, sehingga kita dapat menemukan probabilitas variabel menggunakan fungsi tersebut, disebut Fungsi Kepadatan Probabilitas (PDF) dalam statistik.

Berbagai jenis variabel. Mereka terutama terdiri dari dua jenis:

1. Variabel Diskrit: Variabel yang hanya dapat mengambil nilai terbatas tertentu dalam rentang tertentu disebut variabel diskrit. Biasanya memisahkan nilai dengan interval yang terbatas, misalnya, jumlah dari dua dadu. Saat melempar dua dadu dan menjumlahkan hasil yang dihasilkan, hasilnya hanya dapat dimiliki oleh sekumpulan angka yang tidak melebihi 12 (karena hasil maksimum lemparan dadu adalah 6). Nilainya juga pasti.
2. Variabel Kontinu: Variabel acak kontinu dapat mengambil nilai yang berbeda tak terbatas dalam rentang nilai, misalnya, jumlah curah hujan yang terjadi dalam satu bulan. Curah hujan yang teramati bisa mencapai 1,7 cm, namun nilai pastinya tidak diketahui. Sebenarnya, ini bisa menjadi 1,701, 1,7687, dll. Dengan demikian, kita hanya dapat menentukan rentang nilai yang termasuk di dalamnya. Dalam nilai ini, ia dapat mengambil nilai berbeda yang tak terbatas.

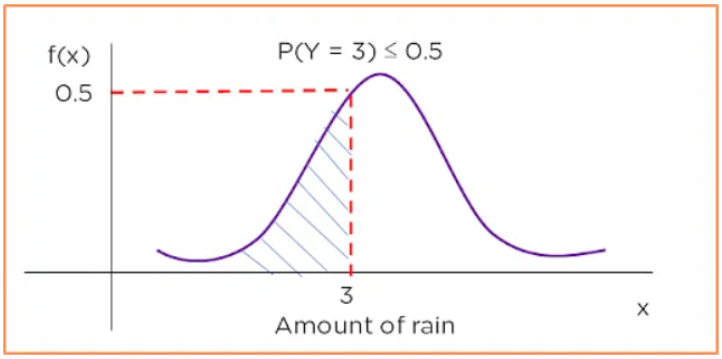
Sekarang, pertimbangkan variabel acak kontinu x, yang memiliki fungsi kerapatan probabilitas, yang menentukan rentang probabilitas yang diambil oleh fungsi ini sebagai f(x). Setelah memplot pdf, kita mendapatkan grafik seperti yang ditunjukkan di bawah ini:



Pada grafik di atas, kita mendapatkan kurva berbentuk lonceng setelah memplot fungsi terhadap variabel. Kurva biru menunjukkan ini. Sekarang pertimbangkan probabilitas titik b. Untuk menemukannya, kita perlu mencari luas di bawah kurva di sebelah kiri b. Ini diwakili oleh P(b). Untuk mencari peluang suatu variabel jatuh di antara titik a dan b, kita perlu mencari luas kurva antara a dan b. Karena probabilitas tidak boleh lebih dari P(b) dan kurang dari P(a), kita dapat menyatakannya sebagai:

P(a) <= X <= P(b)

Perhatikan grafik di bawah ini yang menunjukkan distribusi curah hujan dalam setahun di suatu kota. Sumbu x memiliki curah hujan dalam inci, dan sumbu y memiliki fungsi kerapatan probabilitas. Probabilitas sejumlah curah hujan diperoleh dengan mencari luas kurva di sebelah kirinya.



Untuk probabilitas curah hujan 3 inci, kita menggambar garis yang memotong sumbu y pada titik yang sama pada grafik seperti halnya garis yang memanjang dari 3 pada sumbu x. Ini memberi tahu kita bahwa probabilitas curah hujan 3 inci kurang dari atau sama dengan 0,5.

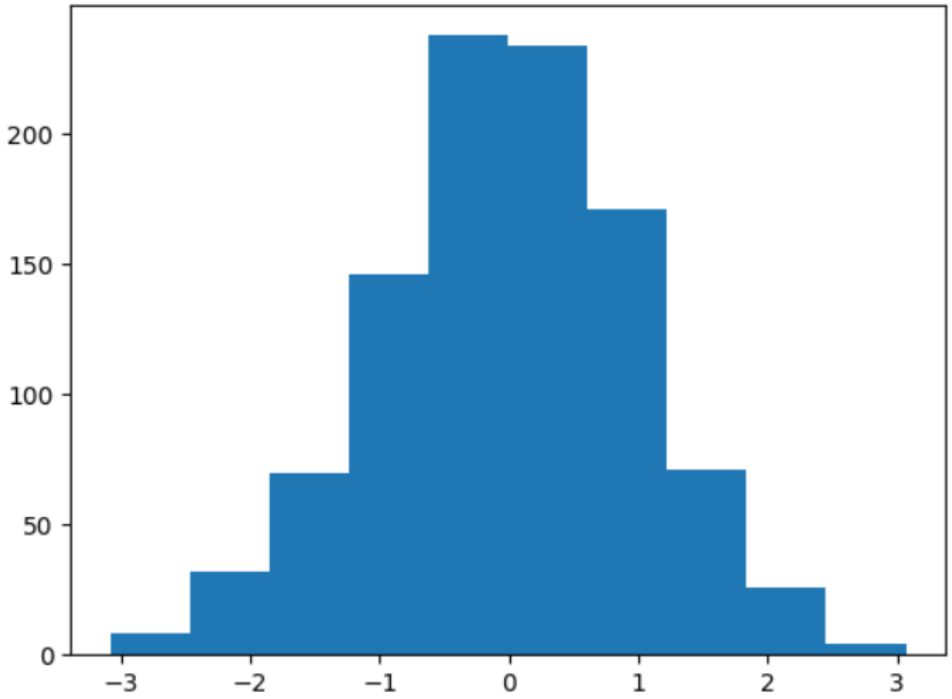
Menerapkan PDF dengan python

Kita akan melihat bagaimana menemukan fungsi kepadatan probabilitas dari sampel acak dengan bantuan Python . kita mulai dengan mengimpor modul yang diperlukan, yang akan membantu kita memplot histogram dan menemukan distribusinya.

```
[ ]  from matplotlib import pyplot as plt

     # Mengimpor fungsi normal dari pustaka numpy.random
     # Digunakan untuk menghasilkan sampel data dari distribusi normal
     from numpy.random import normal

     # Mengimpor fungsi mean dan std dari pustaka numpy
     # Digunakan untuk menghitung rata-rata dan standar deviasi data
     from numpy import mean, std

     # Mengimpor distribusi normal dari pustaka scipy.stats
     # Digunakan untuk analisis statistik terkait distribusi normal
     from scipy.stats import norm
```

Sekarang hasilkan sampel acak yang memiliki fungsi kerapatan probabilitas yang menyerupai kurva berbentuk lonceng. Jenis distribusi probabilitas ini disebut Distribusi Normal.

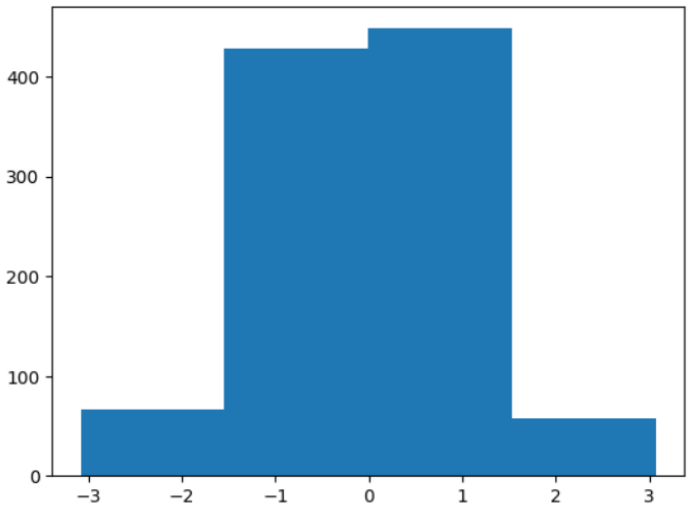
```
[ ]  sample = normal (size=1000)
     plt.hist(sample, bins=10)
     plt.show()
```



Saat memplot histogram, penting untuk memplotnya menggunakan jumlah nampan yang tepat. Pada diagram di atas, kita menggunakan 10 bins. Lihat apa yang terjadi jika kita menggunakan 4 bins.

```
plt.hist(sample, bins=4)
plt.show()
```

```
[ ]  # Ketik kembali kode diatas untuk coba menjalankan

     # Contoh Output
```



Seperti yang kita lihat, histogram ini tidak menyerupai bentuk lonceng sebanyak yang memiliki 10 bins. Hal ini dapat membuat sulit untuk mengenali jenis distribusi.

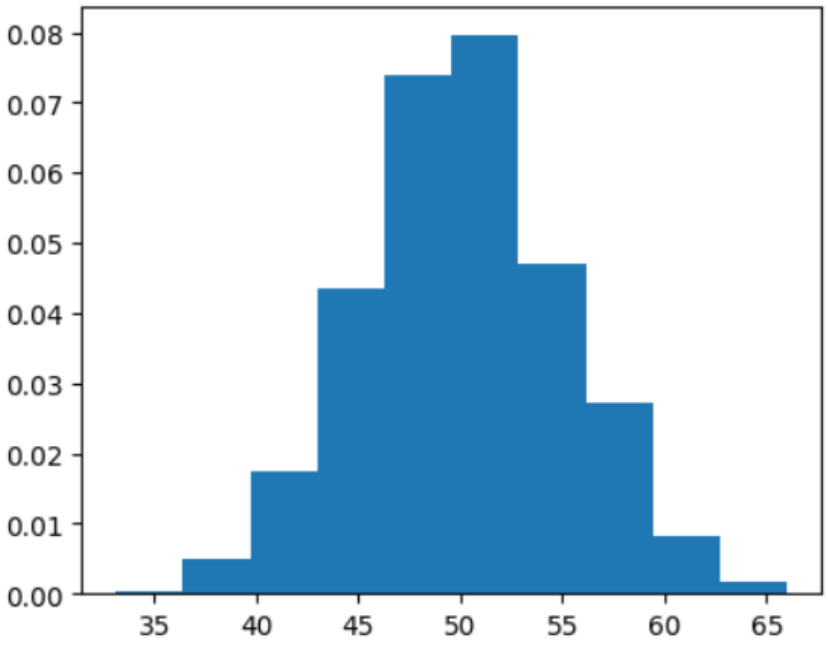
Percobaan 5

Sekarang, lihat bagaimana melakukan estimasi kepadatan parametrik. Pertama, buat 1000 sampel normal dengan rata-rata 50 dan standar deviasi* 5 seperti berikut:

*)Kalian dapat menemukan informasi lebih lanjut tentang standar deviasi di sini.

```
#cara membuat sample data
sample = normal(loc=50, scale=5, size=1000)
#panggil variabel sample untuk mengetahui hasilnya:
#sample
#atau tampilkan dalam bentuk histogram:
plt.figure(figsize=(5,4)) #perkecil media gambar
plt.hist(sample, bins=10, density=True)
plt.show()
```

```
[ ]  # Ketik kembali kode diatas untuk melakukan percobaan

     # Contoh Output
```



Untuk melakukan estimasi parametrik, asumsikan bahwa kita tidak mengetahui distribusi sampel ini. Hal pertama yang perlu kita lakukan dengan sampel adalah mengasumsikan distribusinya. Mari kita asumsikan distribusi normal. Parameter yang berhubungan dengan distribusi normal adalah mean dan standar deviasi (std) untuk menghitung rata-rata dan standar deviasi sebuah sampel.

```
sample_mean = mean(sample)
sample_std = std(sample)
print('Mean=%.3f \nStandard Deviation=%.3f' % (sample_mean, sample_std))
```

```
[ ]  # Ketik kembali kode diatas untuk melanjutkan percobaan

     # Contoh Output
```

```
Mean=50.173
Standard Deviation=4.974
```

Sekarang, tentukan distribusi normal dengan rata-rata dan standar deviasi di atas dengan cara:

```
[ ]  dist = norm(sample_mean, sample_std)
     dist
```

```
<scipy.stats._distn_infrastructure.rv_continuous_frozen at 0x7d6da4a74c40>
```

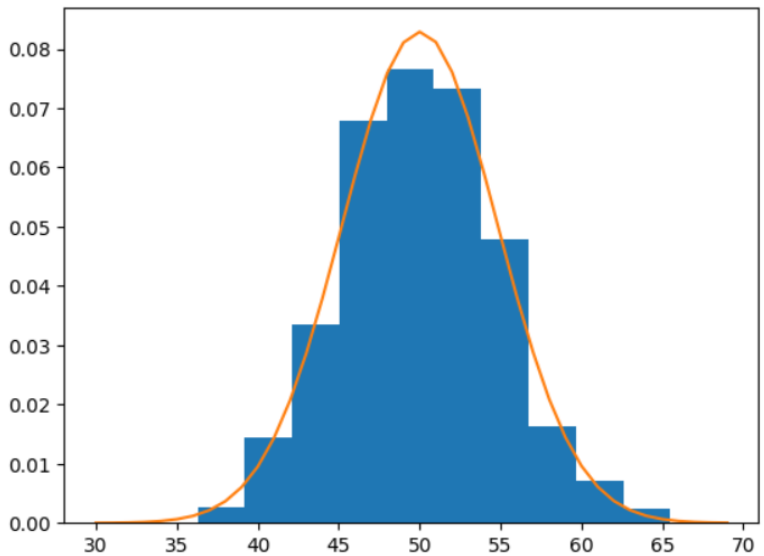
Sekarang, temukan distribusi probabilitas untuk distribusi yang didefinisikan di atas, dengan cara seperti ini:

```
# pertama, buat list nilai yang akan digunakan dalam perhitungan
values = [value for value in range(30, 70)] #menggunakan list comprehension
# manfaatkan list comprehension untuk menerapkan method pdf
# berdasarkan value (yang telah disiapkan sebelumnya) pada data  dist
probabilitas = [dist.pdf(value) for value in values]
probabilitas # panggil variabel probabilitas untuk mengetahui hasilnya
```

Sekarang, plot probabilitas distribusi yang telah kita tentukan di atas bersamaan dengan data sampel:

```
plt.hist(sample, bins=10, density=True)
plt.plot(values, probabilitas)
plt.show()
```

```
[ ]  # Ketik kembali kode-kode diatas untuk menyelesaikan percobaan

     # Contoh Output
```



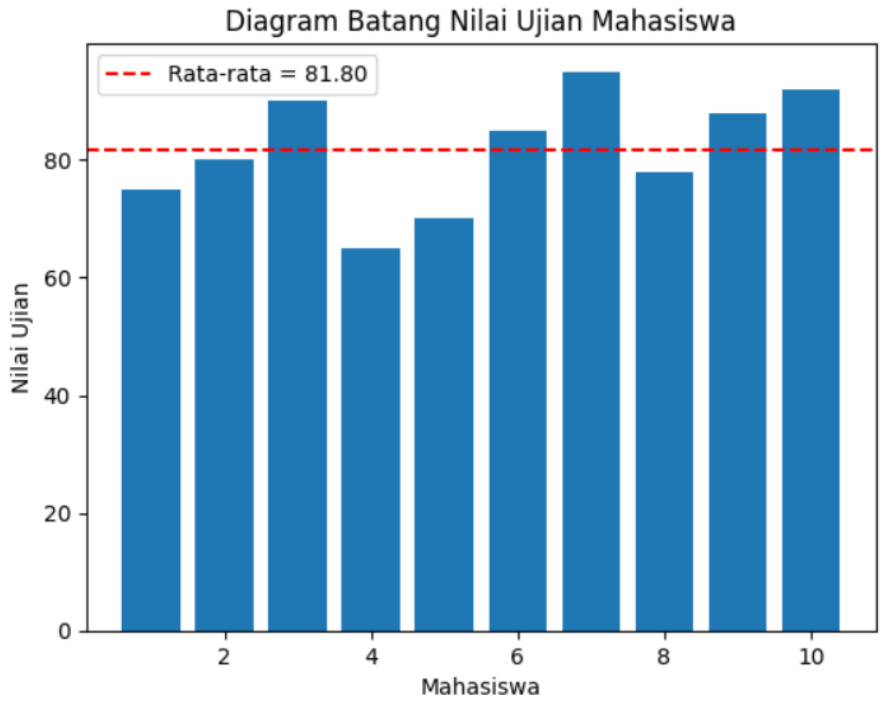
Seperti yang kita lihat, distribusi yang kita asumsikan hampir cocok untuk sampel. Artinya sampel berdistribusi normal. Jika ini tidak sama, kita harus menganggap sampel memiliki distribusi lain dan mengulangi prosesnya. Distribusi lain yang dimaksud disini adalah distribusi tidak normal. Untuk lebih lengkapnya kalian bisa mengunjungi link berikut.

TUGAS PRAKTIKUM

KEGIATAN 1 (BEGINNER)

Diberikan sebuah list data berisi nilai-nilai ujian mahasiswa: [75, 80, 90, 65, 70, 85, 95, 78, 88, 92]. Buatlah sebuah program untuk menghitung rata-rata dari nilai-nilai ini. Setelah itu, visualisasikan data tersebut dalam bentuk diagram batang menggunakan plt.bar(x,y).

```
import matplotlib.pyplot as plt
from functools import reduce

# Data nilai-nilai ujian mahasiswa
nilai_mahasiswa = [75, 80, 90, 65, 70, 85, 95, 78, 88, 92]

# TODO 1: Menghitung rata-rata menggunakan fungsi reduce
# TODO 2: Membuat label mahasiswa (sumbu x) dalam bentuk fungsional dinamis (list-map-lamda)
# TODO 3: Visualisasi data dalam bentuk diagram batang
```

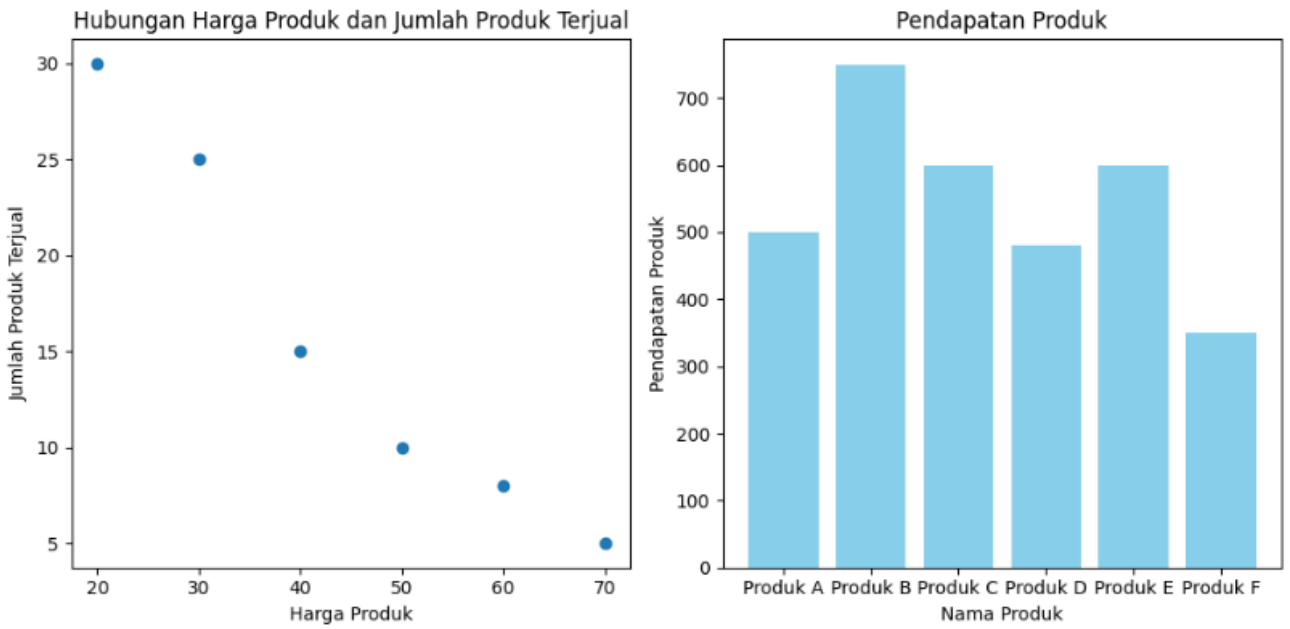


KEGIATAN 2 (INTERMEDIATE)

Diberikan daftar data transaksi penjualan produk dalam bentuk list tupel, di mana setiap tupel berisi informasi sebagai berikut: (nama_produk, harga_produk, jumlah_terjual). Anda diminta untuk membuat visualisasi scatter plot yang menunjukkan hubungan antara harga produk dan

jumlah produk yang terjual serta bar untuk menyajikan pendapatan produk.

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

     data_transaksi = [
         ("Produk A", 50, 10),
         ("Produk B", 30, 25),
         ("Produk C", 20, 30),
         ("Produk D", 60, 8),
         ("Produk E", 40, 15),
         ("Produk F", 70, 5),
     ]

     # TODO 1: Ekstrak harga produk dan jumlah produk terjual untuk visualisasi pertama
     # TODO 2: Buat scatter plot untuk hubungan antara harga produk dan jumlah produk terjual
     # TODO 3: Hitung total pendapatan untuk setiap produk
     # TODO 4: Tambahkan bar chart untuk menyajikan pendapatan produk

     # PS: Kalian bisa memanfaatkan list comprehension, map, dan lambda
     # contoh output: (bebas divariasikan sesuai selera)
```

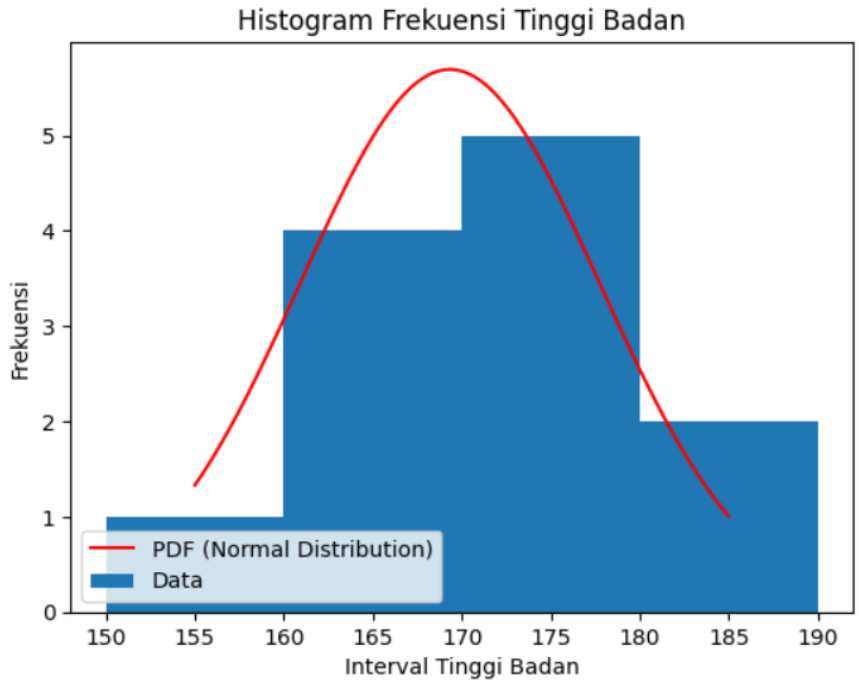


Bisakah kalian menyimpulkan sesuatu dari hasil plot data transaksi penjualan produk diatas? Yah, seperti itulah tujuan dari visualisasi data.

KEGIATAN 3 (ADVANCE)

Diberikan sebuah list data berisi tinggi badan beberapa individu dalam sentimeter: [165, 170, 155, 180, 160, 175, 165, 185, 175, 170, 160]. Buatlah sebuah program untuk menghitung frekuensi tinggi badan dalam interval tertentu (misalnya: 150-160, 161-170, dst.) dan visualisasikan data tersebut dalam bentuk histogram dan kurva PDF dari distribusi normal yang dihasilkan dari data tinggi badan yang ada.

```
[ ]  import matplotlib.pyplot as plt

     # Data tinggi badan individu dalam sentimeter
     tinggi_badan = [165, 170, 155, 172, 180, 160, 175, 165, 185, 175, 170, 160]
     interval_size = 10  # Misalnya interval ukuran per 10 sentimeter

     # TODO 1: buat Fungsi untuk mengelompokkan tinggi badan ke dalam interval tertentu
     # TODO 2: Menghitung frekuensi tinggi badan dalam interval
     # TODO 3: Visualisasi data dalam bentuk histogram
     # TODO 4: Menambahkan kurva pdf pada hasil visualisasi data
```



KRITERIA & DETAIL PENILAIAN TUGAS PRAKTIKUM

Kriteria	Bobot	Program identik*
1. Praktikan mengerjakan kegiatan 1	20	10*
2. Praktikan mengerjakan kegiatan 2	30	15*
3. Praktikan mengerjakan kegiatan 3	40	25*
4. Menjelaskan dan menjawab pertanyaan asisten saat demo	10	10*
Total Nilai (maksimal)	100	60*

*) Jika program identik dengan praktikan lain. Maka akan ada pengurangan nilai pada kedua praktikan.