Modul Pengenalan Komputasi Python

Tim Materi Pengenalan Komputasi 2020/2021

2020-10-16

Catatan

1. Modul ini dirancang untuk dapat menjadi pegangan pemrograman Pengenalan Komputasi, sehingga banyak hal yang dipotong karena keluar dari konteks Pengenalan Komputasi.
2. Anda dapat membuka modul ini saat latihan praktikum.
3. Anda sangat disarankan untuk mencoba menjalankan semua program modul ini di komputer Anda, supaya Anda dapat mengetahui keluaran dari program yang ada.
4. Anda sangat disarankan untuk bereksperimen dari program-program yang ada di modul ini supaya Anda mendapat gambaran lebih jelas mengenai apa yang program Anda lakukan.
5. Anda sangat disarankan membaca tutorial dari tempat lain dan mengeksplor sendiri bahasa yang Anda gunakan.

Contents

1 Modul 1

1.1 Pendahuluan

Pada modul ini, kita menggunakan Python versi 3.9. Anda dapat mendownload Python di https://www.python.org/downloads/release/python-390/, pilihan "Windows x86-64 executable installer".

Beberapa karakteristik dari bahasa ini:

- Python bersifat *case-sensitive*, artinya perbedaan huruf besar dan huruf kecil menyebabkan perbedaan makna.
- Python sangat memerhatikan indentasi dan pergantian baris. Kesalahan indentasi dapat menyebabkan gagal *compile* hingga kesalahan program.
- Variabel di python bersifat implisit dan dinamis. Artinya, sebuah variabel tidak perlu dideskripsikan tipe datanya. Namun di modul ini kita tetap mempelajari tipe data yang ada.

1.2 Input dan Output

Dalam python, program untuk menulis `"Hello, World!"` ke layar adalah seperti berikut:

```
print("Hello, World!")            # (1)

print("Ini program", end="")      # (2)
print(" pertama saya")            # (3)

# Ini adalah sebuah komentar.

# Semua yg ada setelah tanda pagar (#)
# akan diabaikan oleh interpreter.

"""
Selain itu, kita juga bisa membuat komentar
multi baris dengan tiga petik (""")
"""
```

Bagian yang ditandai nomor 1 bertugas menuliskan `"Hello, World!"` ke layar. Sintaks `print` akan otomatis memindahkan baris setelah selesai menulis ke layar, kecuali disertai parameter `end=""`. Berarti, output program di atas adalah `"Hello, World!"` diikuti dengan `"Ini program pertama saya "` di baris selanjutnya.

Untuk melakukan input, kita membutuhkan penampung untuk menyimpan data yang diinputkan. Sebagai contoh, di bawah ini adalah program yang menerima input dan menuliskan ulang yang dimasukkan.

```
S = input("Masukkan kalimat: ")                          # 1
print("Anda memasukkan kalimat " + S)                    # 2

N = int(input("Masukkan sebuah angka: "))                # 3
print("Jika angka Anda ditambah 5, hasilnya " + str(N + 5)) # 4
```

Pada bagian nomor (1) dan (3), program membaca input dari user dan dimasukkan ke variabel `S` dan `N`. Lalu, bila ingin program menerima sebuah bilangan (sehingga bisa dijumlah / dikali / lainnya), kita bungkus `input()` dengan `int()`. Selain `int`, `string` (kumpulan karakter), Python dapat menerima tipe data `float` (bilangan real), dan masih banyak lagi.

1.3 Tipe Data

Ada beberapa macam tipe data, namun dalam Pengenalan Komputasi kita hanya akan banyak menggunakan tipe data berikut:

`bool`	Boolean	`True` atau `False`
`int`	Bilangan bulat	seluruh bilangan bulat
`float`	Bilangan real	seluruh bilangan real
`string`	Teks	kumpulan karakter

Contoh penggunaan:

```
B = True          # Boolean
I = 123456789012  # Bilangan bulat
R = 2.331973      # Bilangan real
S = "abc"         # Teks
S = 'def'
```

Perhatkan pada variabel `S`, Anda bebas menggunakan petik satu (') atau petik dua (")

1.4 Operator

1.4.1 Operator Aritmatika

Operator	Deskripsi	Contoh
`+`	Penjumlahan	`2 + 3` bernilai `5`
`-`	Pengurangan	`1 - 8` bernilai `-7`
`*`	Perkalian	`5 * 6` bernilai `30`
`/`	Pembagian	`13 / 5` bernilai `2.6`
`//`	Pembagian (dibulatkan ke bawah)	`13 // 5` bernilai `2`
`%`	Sisa Bagi / Modulo	`13 % 5` bernilai `3`

1.4.2 Operator Assignment

Operator	Deskripsi	Contoh
`=`	Assignment	`N = 5`
`+=`	Penjumlahan	`N += 5`, N akan ditambah 5.
`-=`	Pengurangan	`N -= 5`, N akan dikurang 5.
`*=`	Perkalian	`N *= 5`, N akan dikali 5.
`/=`	Pembagian	`N /= 5`, N akan dibagi 5.
`//=`	Pembagian (dibulatkan ke bawah)	`N //= 5`, N akan dibagi 5 (dibulatkan ke bawah).
`%=`	Sisa Bagi / Modulo	`N %= 5`, N akan dimodulo 5.

1.4.3 Operator Relasional

Operator	Deskripsi	Contoh True	Contoh False
==	Sama dengan	`2 == 2`	`2 == 3`
!=	Tidak sama dengan	`3 != 2`	`3 != 3`
<	Kurang dari	`2 < 3`	`2 < 2`
>	Lebih dari	`3 > 2`	`2 > 3`
<=	Kurang dari sama dengan	`2 <= 2`	`3 <= 1`
>=	Lebih dari sama dengan	`6 >= 5`	`2 >= 4`

1.4.4 Operator Logika

Operator	Deskripsi	Contoh True	Contoh False
`and`	Dan: True jika kedua operand True	`(1 < 2)and (3 == 3)`	`(1 == 2)and (3 == 3)`
`or`	Atau: True jika salah satu operand True	`(1 < 2)or (4 == 3)`	`(3 < 2)or (2 == 3)`
`not`	Negasi: True jika operand False	`not (3 < 2)`	`not (1 < 2)`

1.5 Percabangan

Dalam pemrograman, terdapat percabangan. Dengan demikian, program kita dapat berperilaku tergantung input user. Misal kita buat program yang memeriksa apakah sebuah bilangan positif:

```
print("Masukkan nilai N: ", end="")
N = int(input())

if (N > 0):
    print(str(N) + " bilangan positif")
```

Lalu, jika kita ingin menuliskan kebalikannya:

```
...
    if (N > 0):
        print(str(N) + " bilangan positif")
    else: # N <= 0
        print(str(N) + " bilangan bukan positif")
...
```

Namun, kita tahu kadang bilangan bisa nol atau negatif, jadi perlu kita tambahkan:

```
...
    if (N > 0):
        print(str(N) + " bilangan positif")
    elif (N < 0):
        print(str(N) + " bilangan negatif")
    else: # N == 0
        print(str(N) + " bilangan nol")
...
```

Perhatikan juga kalau kita bisa membuat else ini berulang sampai yang kita mau. Selain itu, kita juga bisa meletakkan `if` di dalam `if`.

```
...
    if (N >= 0):
        if (N > 0):
            print(str(N) + " bilangan positif")
        else: # N == 0
            print(str(N) + " bilangan nol")
    else: # N < 0
        print(str(N) + " bilangan negatif")
...
```

2 Modul 2

2.1 Pengulangan

Pada pemrograman, sering kali dibutuhkan pemrosesan berulang-ulang untuk mencapai suatu hasil tertentu. Apabila pengulangan ini dilakukan secara manual ukuran file program akan menjadi terlalu besar. Contoh sederhana adalah jika kita ingin menuliskan "Hello World" di layar sebanyak 1000 kali, maka akan dibutuhkan paling tidak 1000 baris perintah. Menggunakan sintaks pengulangan, persoalan tersebut dapat diselesaikan hanya menggunakan beberapa baris program.

```
for i in range(1000):
    print("Hello world")
```

2.1.1 While Loop

Salah satu sintaks yang looping / pengulangan yang sering digunakan adalah sintaks While-Do. Program akan mengecek sebuah kondisi yang diberikan terlebih dahulu sebelum menjalankan statement yang ada di dalamnya.

Berikut adalah program yang menerima a dan b dan menuliskan $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
i = a
while (i <= b):
    print(i)
    i += 1
```

2.1.2 For Loop

Bentuk looping yang kedua adalah bentuk For. Bentuk ini umumnya digunakan untuk pengulangan yang sudah diketahui jumlahnya. Namun, for loop juga dapat dibuat menggunakan while-do loop.

Berikut adalah program yang menerima a dan b dan menuliskan $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
for i in range(a, b+1):
    print(i)
```

`range` dapat digunakan dengan satu, dua, atau tiga parameter. Untuk lebih jelasnya, perhatikan potongan kode berikut:

```
range(10)          # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
range(2, 5)        # 2, 3, 4
range(7, 2, -1)    # 7, 6, 5, 4, 3
range(2, 8, 2)     # 2, 4, 6
```

2.2 Perulangan bersarang

Perhatikan pula, perulangan dapat dilakukan di dalam perulangan. Sebagai contoh, berikut adalah program untuk membuat pola persegi.

```
n = int(input())
for i in range(n):
    for j in range(n):
        print("*", end="")
    print()
```

3 Modul 3

3.1 Array

Array adalah variabel dengan satu buah nama, tetapi mengandung banyak nilai. Akses nilai-nilainya dilakukan dengan indeks.

Perhatikanlah contoh berikut!

Indeks	0	1	2	3	4	5	6	7	8	9
A	3	10	5	7	11	19	23	35	37	12

- `A[0] = 3`
- `A[1] = 10`
- `A[7] = 35`

Pada contoh diatas, kita memiliki sebuah variabel yang bernama A. Variabel A tersebut memiliki 10 buah nilai, dimana nilai-nilai tersebut dapat diakses dengan indeks. Untuk mengakses indeks ke – x, gunakanlah `A[x]`. Dan nilai `A[x]` itu bisa kita anggap sebagai variabel yang berdiri sendiri. Konsep inilah yang kita sebut dengan array. Perhatikan pula bahwa indeks dimulai dari 0.

3.1.1 Deklarasi Array

Karena array juga merupakan sebuah variabel, maka array juga memerlukan deklarasi seperti variabel lainnya.

Contoh deklarasi array:

- `x = [0 for i in range(n)]` membuat array berukuran `n` dengan isi 0.
- `x = ["*"for i in range(100)]` membuat array berukuran `100` dengan isi karakter `"*"`.

Untuk contoh tersebut array `A` yang terdefinisi adalah `A[0]`, `A[1]`, `A[2]`, ..., `A[9]` Mengakses nilai indeks di luar batasan tersebut akan menyebabkan runtime error. Oleh karena itu, tentukanlah rentang indeks yang akan digunakan saat deklarasi dengan tepat (sesuai dengan kebutuhan).

3.1.2 Array dan Variabel

Suatu array dapat kita anggap sebagai variabel, sehingga segala jenis operasi pada variabel juga berlaku pada array. Sebagai contoh, kita memiliki suatu array

```
tabel = [0 for i in range(10)]
```

Maka array tabel tersebut akan terdefinisi untuk indeks 0 sampai dengan indeks 9. Maka kita bisa melakukan instruksi

```
tabel[2] = int(input())
```

Jika diberikan 5 buah bilangan , dan kita perlu menyimpan bilangan tersebut pada tabel, kita bisa melakukan

```
tabel[0] = int(input("Masukkan nilai ke-0: "))
tabel[1] = int(input("Masukkan nilai ke-1: "))
tabel[2] = int(input("Masukkan nilai ke-2: "))
tabel[3] = int(input("Masukkan nilai ke-3: "))
tabel[4] = int(input("Masukkan nilai ke-4: "))
```

Namun, cara menginput tersebut kurang efisien. Akan lebih efisien jika kita menginput menggunakan perulangan (looping).

```
for i in range(5):
    tabel[i] = int(input('Masukkan nilai ke-' + str(i) + ': '))
```

4 Modul 4

4.1 Fungsi

Apa itu fungsi? Fungsi adalah suatu bagian dari program yang mampu mengerjakan tugas atau operasi tertentu di luar program utama. Fungsi akan mengembalikan nilai sesuai algoritma yang diberikan.

Sebagai contoh, fungsi untuk menghitung kuadrat adalah sebagai berikut:

```
def kuadrat(x):
    x2 = x * x
    return x
```

Fungsi di atas bernama `kuadrat` dan mengembalikan nilai kuadrat. Fungsi itu juga menerima satu parameter bernama `x`.

Sebagai contoh, berikut program lengkap yang menerima input dan mengeluarkan kuadrat dari bilangan input.

```
def kuadrat(x):
    x2 = x * x
    return x2

n = int(input())
n2 = kuadrat(n)
print(n2)
```

Perhatikan pada contoh di atas, variabel di program utama bernama `n`. Namun di fungsi, variabel berubah nama menjadi `x`. Meskipun x berubah, nilai n di program utama tidak akan berubah.

Sebuah fungsi juga dapat menerima lebih dari satu parameter. Selain itu, fungsi juga dapat melakukan hal-hal layaknya program biasa, namun tidak dapat mengubah variabel di program utama. Sebagai contoh, berikut adalah fungsi yang menghitung nilai a^b.

```
def pangkat(a, b):
    # asumsi b >= 0
    c = 1
    for i in range(b):
        c *= a
    return c
```

Catatan: Anda tidak disarankan menaruh array sebagai parameter fungsi, karena ada hal khusus yang belum diajarkan di Pengenalan Komputasi.

4.2 Prosedur

Prosedur sebenarnya sama seperti fungsi, namun tidak ada kembalian. Sebagai contoh, berikut adalah program untuk menuliskan menu:

```
print("Menu:")
```

```
print("1. Burger")
print("2. Ayam Geprek")
print("3. Mie Instan")
print("Masukkan pilihan: ")
pilihan_makanan = int(input())

print("Menu:")
print("1. Jus Alpukat")
print("2. Thai Tea")
print("3. Teh Tarik")
print("Masukkan pilihan: ")
pilihan_minuman = int(input())

print("Menu:")
print("1. Kentang")
print("2. Krupuk")
print("3. Abon")
print("Masukkan pilihan: ")
pilihan_tambahan = int(input())
```

Seperti yang tertulis di atas, menuliskan menu perlu berkali-kali. Kita dapat meringkasnya menjadi:

```
def tulis_menu(pil1, pil2, pil3):
    print("Menu:")
    print("1. " + str(pil1))
    print("2. " + str(pil2))
    print("3. " + str(pil3))
    print("Masukkan pilihan: ")

tulis_menu("Burger", "Ayam Geprek", "Mie Instan")
pilihan_makanan = int(input())

tulis_menu("Jus Alpukat", "Thai Tea", "Teh Tarik")
pilihan_minuman = int(input())

tulis_menu("Kentang", "Krupuk", "Abon")
pilihan_tambahan = int(input())
```

4.3 Matriks

Matriks pada dasarnya adalah array 2 dimensi. Matriks dapat dideklarasikan dengan:

```
A = [[0 for j in range(20)] for i in range(10)]
```

Pada kode di atas, artinya kita membuat matriks dengan nama variabel A, dengan tipe elemen integer dan nilai awal 0, dan berukuran 10 baris $\times$ 20 kolom. Sama seperti array, untuk mengakses elemen baris ke-i dan kolom ke-j, kita perlu mengakses `A[i-1][j-1]` karena indeks matriks di mulai dari 0.

Sebagai contoh, berikut kode untuk membaca matriks dan menuliskannya kembali.

```
n = int(input())
m = int(input())
```

```
A = [[0 for j in range(m)] for i in range(n)]

for i in range(n):
    for j in range(m):
        A[i][j] = int(input("masukkan elemen baris " + str(i + 1) + " kolom " +
            str(j + 1) + ": "))

for i in range(n):
    for j in range(m):
        print(A[i][j], end=" ")
    print("")
```

Berikut contoh program untuk membaca matriks A berukuran $n \times m$, membaca matriks B berukuran $m \times l$, dan menuliskan hasil perkalian matriks A kali B berukuran $n \times l$.

```
n = int(input())
m = int(input())
l = int(input())

A = [[0 for j in range(m)] for i in range(n)]
B = [[0 for j in range(l)] for i in range(m)]
C = [[0 for j in range(l)] for i in range(n)]

for i in range(n):
    for j in range(m):
        A[i][j] = int(input("masukkan elemen A baris " + str(i + 1) + " kolom "
            + str(j + 1) + ": "))

for i in range(m):
    for j in range(l):
        B[i][j] = int(input("masukkan elemen B baris " + str(i + 1) + " kolom "
            + str(j + 1) + ": "))

for i in range(n):
    for j in range(l):
        C[i][j] = 0
        for k in range(m):
            C[i][j] += A[i][k] * B[k][j]
        print(C[i][j], end=" ")
    print("")
```

5 Modul 5

5.1 Menggunakan Jupyter Notebook

Untuk modul 4 dan 5, kita akan menggunakan Jupter Notebook. Untuk menginstall Jupyter Notebook, silakan buka halaman https://docs.anaconda.com/anaconda/install/.

Untuk menggunakan pandas, kita akan menambahkan line berikut di kernel kita, tepat sesudah header:

```
import pandas as pd
```

Pada modul ini, kita akan menggunakan data yang bisa didownload di https://drive.google.com/drive/folders/1o2Zg_Lc911dsW0Iw37uWgYqM0-dR8Jro?usp=sharing. Contoh notebook juga bisa didownload dari link yang sama.

5.2 Analisa Data

5.2.1 Membuat Dataframe

Untuk membuat data frame, kita perlu membuat dictionary dengan key berupa nama kolom dan berisi array dari data yang ada. Sebagai contoh, perhatikan potongan kode berikut:

```
import pandas as pd

input_data = {}
input_data["A"] = [0 for i in range(5)]
input_data["B"] = [0 for i in range(5)]

for i in range(5):
    input_data["A"][i] = input("Nilai A untuk data ke-" + str(i + 1) + ": ")
for i in range(5):
    input_data["B"][i] = input("Nilai B untuk data ke-" + str(i + 1) + ": ")

df = pd.DataFrame(data=input_data)
print(df)
```

5.2.2 Membaca dan Menulis Data

Untuk membaca data csv, kita dapat menggunakan method `read_csv`. Untuk membaca data excel, kita dapat menggunakan method `read_excel`. Untuk menulis data, baik ke csv maupun excel, perhatikan contoh berikut:

```
import pandas as pd

# pandas akan membaca file
# tingkatinflasi20082013.csv
# yang ada di folder yang sama dengan
# tempat kode python ini disimpan
df1 = pd.read_csv("tingkatinflasi20082013.csv")
```

```
print(df1)

# pandas akan membaca file data.xlsx
# yang ada di D:/, lalu meload data
# yang ada di sheet bernama "Sheet 1"
df2 = pd.read_excel("D:/data.xlsx", sheet_name="Sheet 1")
print(df2)

# pandas akan menulis data ke file
# data_out.csv yang ada di folder sama
# dengan file python ini
df1.to_csv("data_out.csv")

# pandas akan menulis data ke
# D:/data_out.xlsx di sheet bernama
# "Sheet 1" dan "Sheet 2"
writer = pd.ExcelWriter("data_out.xlsx")
df2.to_excel(writer, "Sheet1")
df2.to_excel(writer, "Sheet2")
writer.save()
```

5.2.3 Mengakses Data

Perhatikan contoh berikut:

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# mengambil data ke-5
print(df.loc[4])
# Tahun                          2009
# Cakupan            Prov. Jawa Barat
# Tingkat_Inflasi                2.02
# Name: 4, dtype: object

# mengambil data ke-5 hingga 7
print(df[4:7])
#    Tahun           Cakupan  Tingkat_Inflasi
# 4   2009  Prov. Jawa Barat             2.02
# 5   2009          Nasional             2.78
# 6   2010      Kota Bandung             4.53

# mengambil data ke-17 hingga akhir
print(df[16:])
#     Tahun           Cakupan  Tingkat_Inflasi
# 16   2013  Prov. Jawa Barat             9.15
# 17   2013          Nasional             8.38

# mengambil 5 data pertama
print(df[:5])
#    Tahun           Cakupan  Tingkat_Inflasi
# 0   2008      Kota Bandung            10.23
# 1   2008  Prov. Jawa Barat            11.11
# 2   2008          Nasional            11.06
# 3   2009      Kota Bandung             2.11
# 4   2009  Prov. Jawa Barat             2.02
```

```
# melihat panjang data
print(len(df))
# 18

# mengambil kolom "Cakupan" dari data ke-2
print(df.loc[1, "Cakupan"])
# 'Prov. Jawa Barat'
```

Selain itu, kita bisa mengakses data berdasar kriteria. Perhatikan contoh berikut:

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# mengambil data tahun 2012
print(df.loc[df["Tahun"] == 2012])
#     Tahun           Cakupan  Tingkat_Inflasi
# 12   2012      Kota Bandung             4.02
# 13   2012  Prov. Jawa Barat             3.86
# 14   2012          Nasional             4.30

# mengambil data Kota Bandung sebelum tahun 2012
print(df.loc[(df["Cakupan"] == "Kota Bandung") & (df["Tahun"] < 2012)])
#     Tahun       Cakupan  Tingkat_Inflasi
# 0    2008  Kota Bandung            10.23
# 3    2009  Kota Bandung             2.11
# 6    2010  Kota Bandung             4.53
# 9    2011  Kota Bandung             2.75

# mengambil data dengan tingkat inflasi di atas 10
# atau di bawah 3
print(df.loc[(df["Tingkat_Inflasi"] > 10) | (df["Tingkat_Inflasi"] < 3)])
#    Tahun           Cakupan  Tingkat_Inflasi
# 0   2008      Kota Bandung            10.23
# 1   2008  Prov. Jawa Barat            11.11
# 2   2008          Nasional            11.06
# 3   2009      Kota Bandung             2.11
# 4   2009  Prov. Jawa Barat             2.02
# 5   2009          Nasional             2.78
# 9   2011      Kota Bandung             2.75
```

5.2.4 Mengambil Ekstremum

Ekstremum adalah data yang ekstrem: paling tinggi atau paling rendah

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# Mengambil data dengan inflasi maksimum
imax = df["Tingkat_Inflasi"].idxmax()
print(df[imax:imax + 1])
#    Tahun           Cakupan  Tingkat_Inflasi
# 1   2008  Prov. Jawa Barat            11.11
```

```
# Mengambil data dengan inflasi minimum
imin = df["Tingkat_Inflasi"].idxmin()
print(df[imin:imin + 1])
#    Tahun           Cakupan  Tingkat_Inflasi
# 4   2009  Prov. Jawa Barat             2.02
```

5.2.5 Mengurutkan Data

Data dapat diurutkan secara tidak menurun (ascending) tidak menaik (descending).

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# Mengurutkan data berdasar tingkat inflasi, ascending
print(df.sort_values(["Tingkat_Inflasi"], ascending=[1]))

# Mengurutkan data berdasar tahun ascending,
# lalu tingkat inflasi descending
print(df.sort_values(["Tahun", "Tingkat_Inflasi"], ascending=[1, 0]))
```

5.2.6 Tabel Frekuensi

Kita dapat membuat tabel frekuensi. Tabel frekuensi berdasar kolom X artinya kita mendaftar semua kemungkinan nilai di kolom X secara unik, lalu menghitung berapa kali nilai itu muncul.

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# Mendaftar kemunculan tiap tahun pada data
print(df["Tahun"].value_counts())
# 2013    3
# 2012    3
# 2011    3
# 2010    3
# 2009    3
# 2008    3
```

5.2.7 Menentukan Range

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

# Mengambil nilai minimum dan maximum tiap kolom
minimum = df.min()
maximum = df.max()

# Menuliskan range tingkat inflasi
```

```
print(maximum["Tingkat_Inflasi"]) # 11.11
print(minimum["Tingkat_Inflasi"]) # 2.02
```

5.2.8 Statistik Sederhana

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

df.describe()
#              Tahun  Tingkat_Inflasi
# count    18.000000        18.000000
# mean   2010.500000         5.818889
# std       1.757338         3.148673
# min    2008.000000         2.020000
# 25%    2009.000000         3.272500
# 50%    2010.500000         4.415000
# 75%    2012.000000         8.277500
# max    2013.000000        11.110000
```

Dari data di atas, kita bisa melihat ada 18 data. Rata-rata tingkat inflasi adalah 5.8189. Standar deviasi dari tingkat inflasi adalah 3.1487. Tingkat inflasi minimum adalah 2.02 dan maksimumnya 11.11. Tingkat inflasi juga memiliki kuartil bawah 3.273, kuartil tengah 4.4150, dan kuartil atas 8.2775.

Kita dapat juga mengambil statistik satu per satu:

```
import pandas as pd

df = pd.read_csv("tingkatinflasi20082013.csv")

df.mean()
# Tahun              2010.500000
# Tingkat_Inflasi       5.818889

df["Tingkat_Inflasi"].mean() # 5.818888888888889
df["Tingkat_Inflasi"].std() # 3.1486727154915681
```

5.2.9 Koefisien Korelasi

Dua kolom pada data yang sama bisa memiliki korelasi. Tingkat korelasi ini kita sebut sebagai koefisien korelasi. Cara memaknai koefisien korelasi adalah sebagai berikut:

- Semakin mendekati 0, semakin dua kolom tidak berkorelasi.
- Semakin mendekati 1, semakin dua kolom berbanding lurus.
- Semakin mendekati -1, semakin dua kolom berbanding terbalik.

Berikt ini adalah contoh bila kita ingin melihat korelasi tahun dengan tingkat inflasi:

```
df["Tingkat_Inflasi"].corr(df["Tahun"])
```

5.3 Visualisasi Data

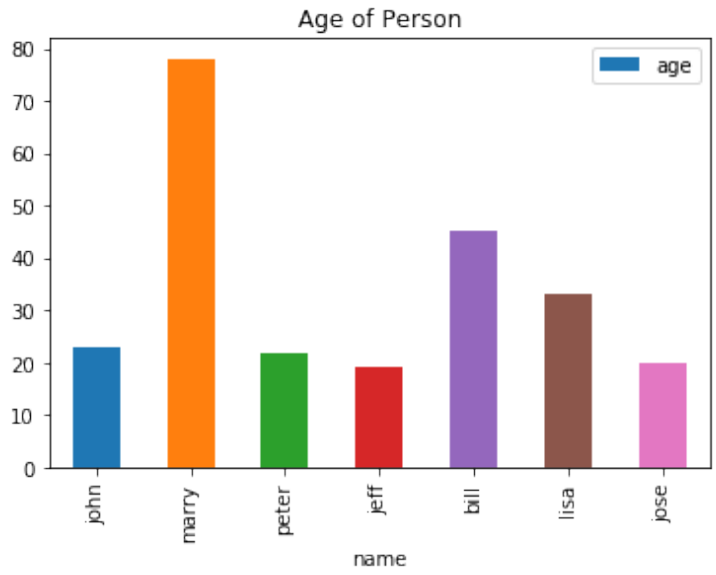
Untuk visualisasi data, kita akan menggunakan matplotlib. Matplotlib harusnya sudah diinstall saat Anda menginstall Anaconda.

Pada subbab ini, kita akan menggunakan data yang bisa didownload di https://drive.google.com/drive/folders/1o2Zg_Lc911dsW0Iw37uWgYqM0-dR8Jro?usp=sharing. Contoh notebook juga bisa didownload dari link yang sama. Ada 3 data yang bisa akan digunakan:

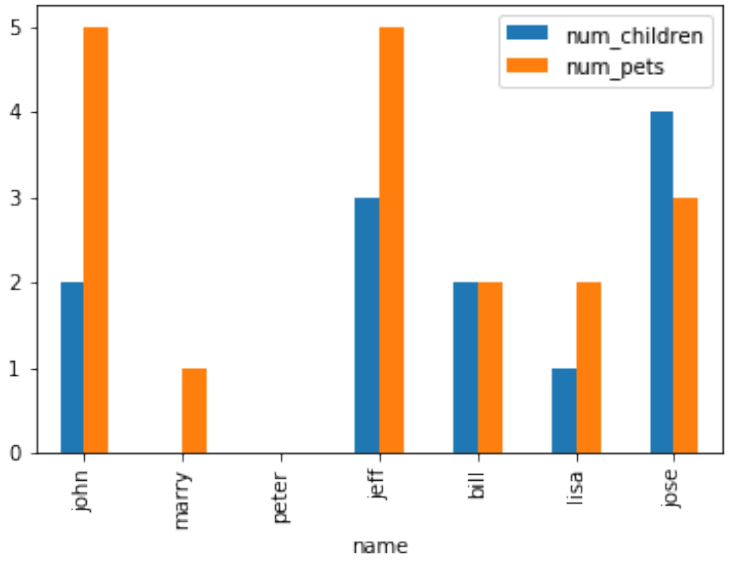
```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("data.csv")
medali = pd.read_csv("medali.csv")
animal = pd.read_csv("animal.csv")
```

5.3.1 Bar Chart

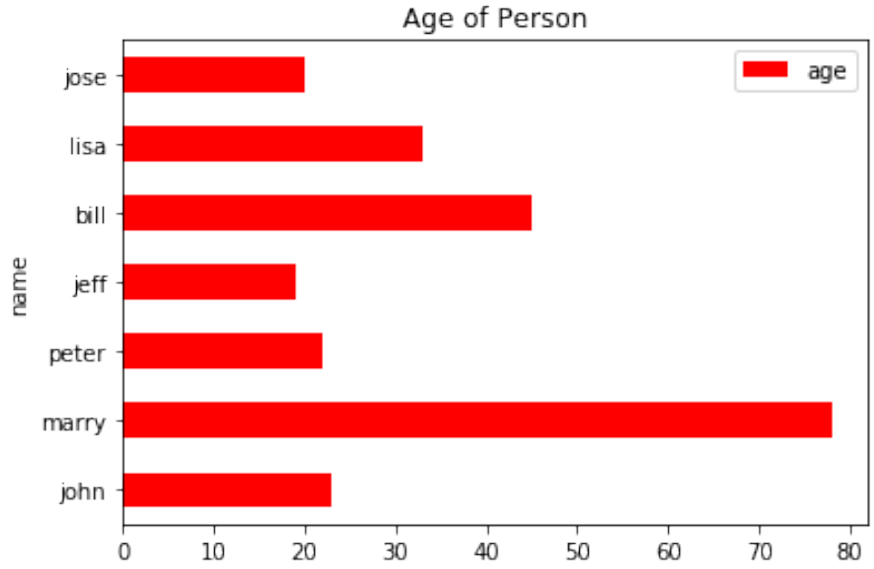
```
# Vertical bar chart untuk menampilkan umur dari setiap orang
data.plot(kind="bar",x="name",y="age",title="Age of Person")
```



```
# Banyaknya anak (num_children) dan banyaknya piaraan (num_pets) dalam
# 1 grafik vertical bar chart
data.plot(kind="bar",x="name",y=["num_children","num_pets"])
```

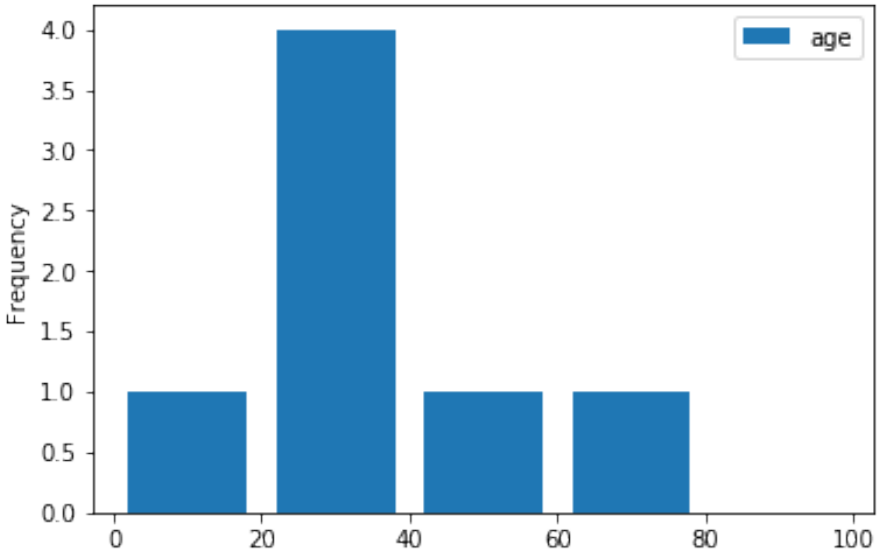



```
data.plot(kind="barh",x="name",y="age",title="Age of Person", color="red")
```



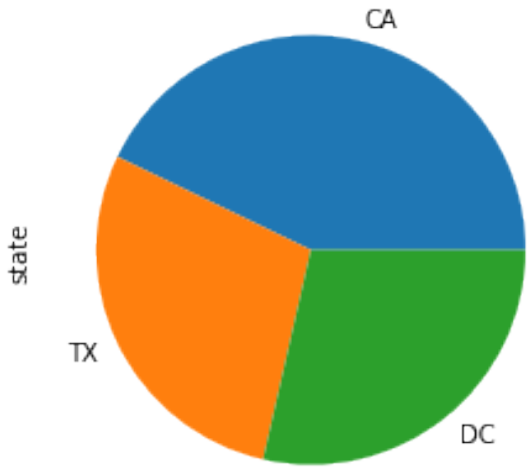
5.3.2 Histogram

```
# Histogram orang berdasarkan kelompok umur: 0-20; 21-40; 41-60; 61-80; 81-100
data[["age"]].plot(kind="hist",bins=[0,20,40,60,80,100],rwidth=0.8)
```



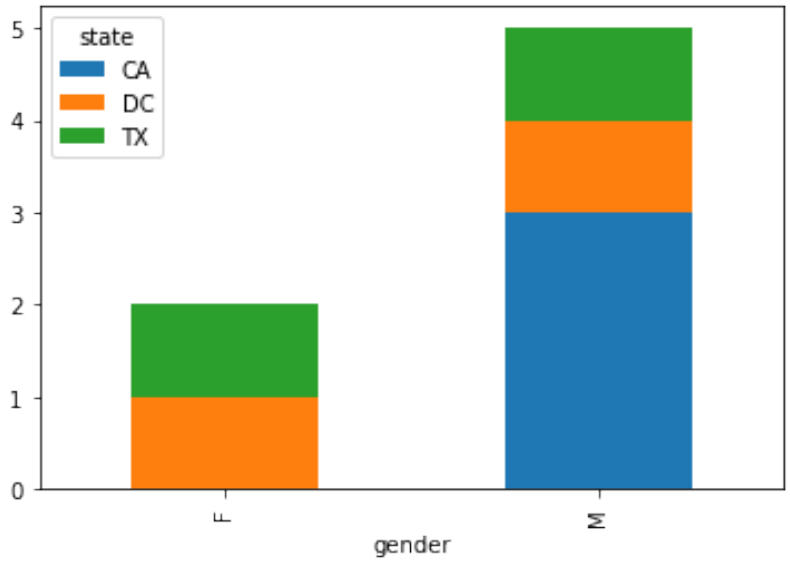
5.3.3 Pie Chart

```
# Komposisi banyaknya orang berdasarkan negara
data["state"].value_counts().plot(kind = "pie")
```

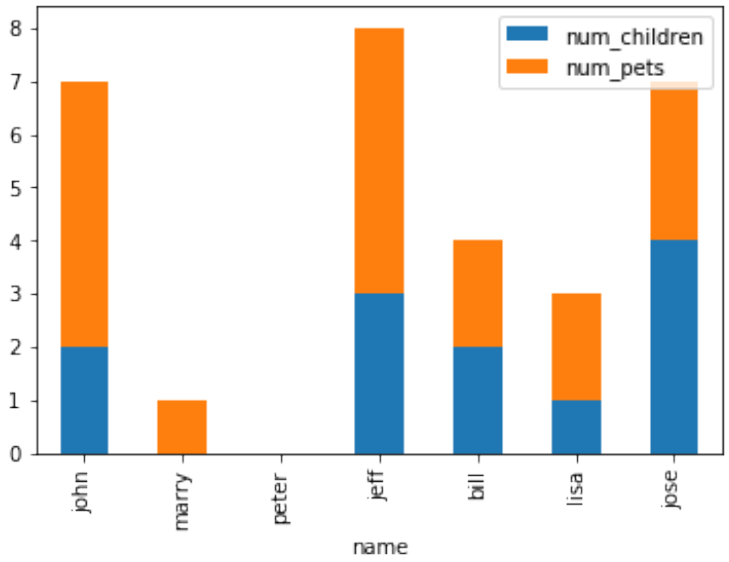


5.3.4 Stacked Bar Chart

```
# Banyaknya data per jenis kelamin (gender) per negara bagian (state)
data.groupby(["gender","state"])["name"].size().unstack().plot(kind="bar",
    stacked=True)
```

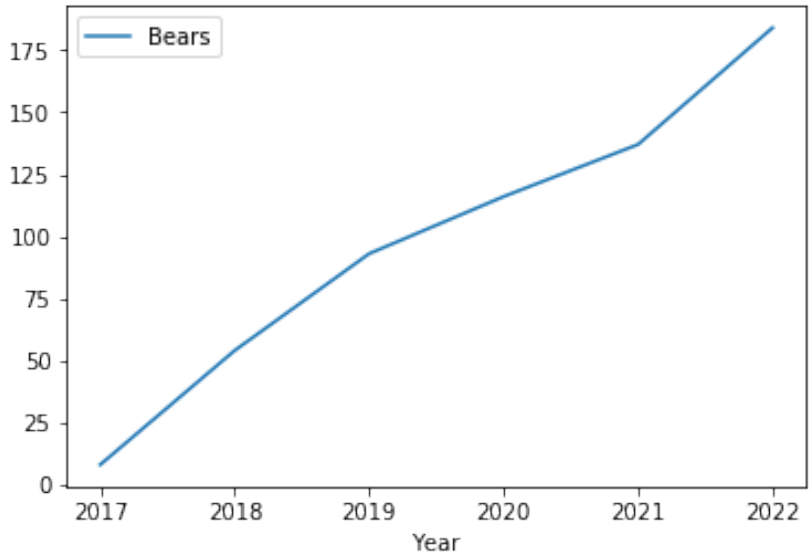



```
# Banyaknya anak (num_children) dan banyaknya piaraan (num_pets)
# dalam 1 grafik stacked bar chart
data.plot(kind="bar", x="name", y=["num_children","num_pets"], stacked=True)
```

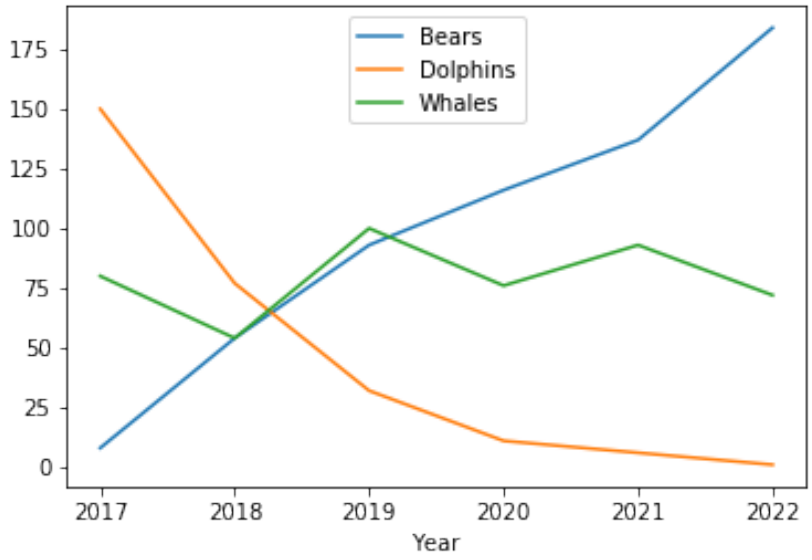


5.3.5 Line Chart

```
# Pertumbuhan populasi beruang (Bears) dari tahun ke tahun dalam line chart
animal.plot(kind="line",x="Year",y="Bears")
```

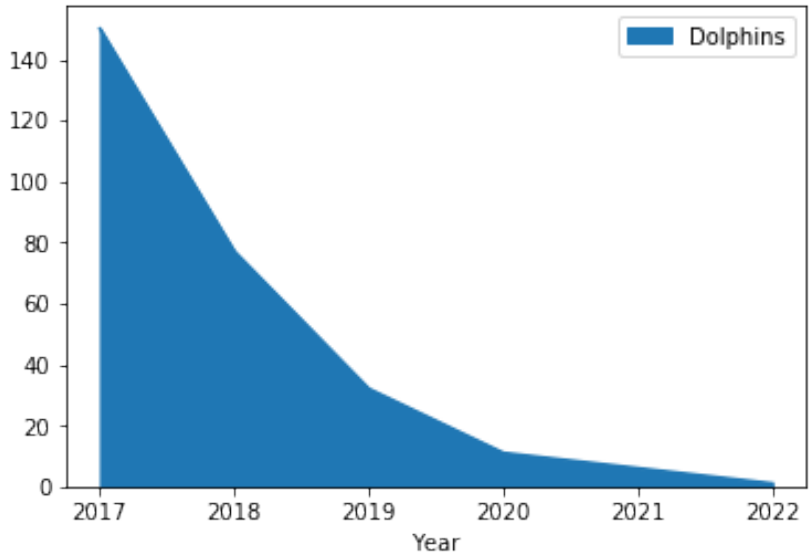



```
# Pertumbuhan populasi beruang (Bears), lumba-lumba (Dolphins), dan ikan paus (
    Whales)
# dari tahun ke tahun dalam 1 line chart
animal.plot(kind="line",x="Year", y=["Bears","Dolphins","Whales"])
```

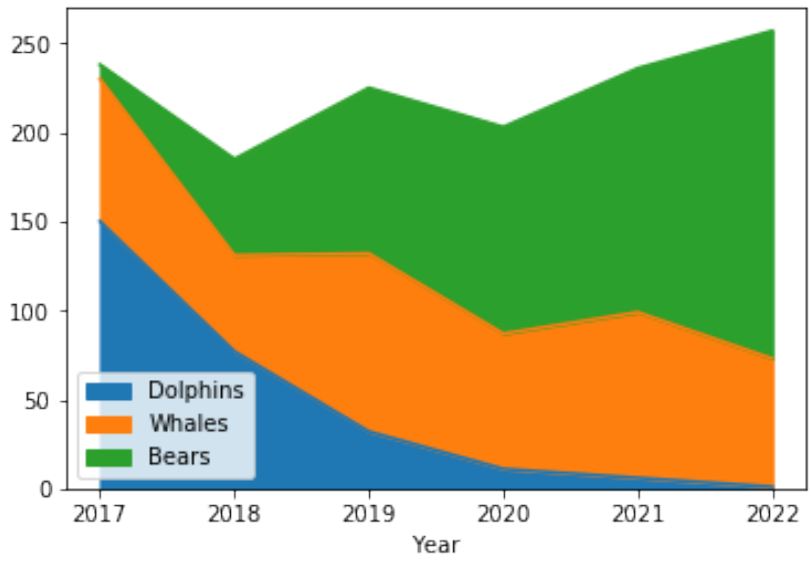


5.3.6 Area Chart

```
# Pertumbuhan populasi lumba-lumba (Dolphins) dari tahun ke tahun
# dalam area chart
animal.plot(kind="area",x="Year", y="Dolphins")
```

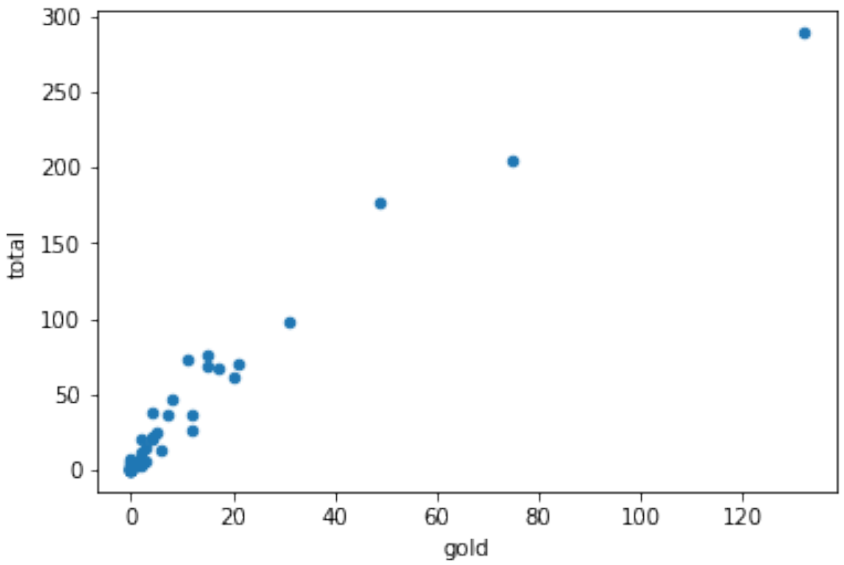



```
# Pertumbuhan populasi lumba-lumba (Dolphins), ikan paus (Whales), dan beruang
    (Bears),
# dari tahun ke tahun dalam stacked area chart
animal.plot(kind="area",x="Year", y=["Dolphins","Whales","Bears"])
```



5.3.7 Scatter dan Bubble Plot

```
# Relationship antara variable gold dan total dalam grafik scatter plot
# dan tunjukkan adanya korelasi positif
medali.plot(kind="scatter", x="gold", y="total")
```




```
# Banyaknya total medali dikaitkan dengan perolehan nilai medali emas (gold)
# pada sumbu x dan perolehan medali perak (silver) pada sumbu y
# dalam grafik bubble plot
medali.plot(kind="scatter",x="gold", y="silver",sizes=medali["total"], color="
    orange")
```