# pertemuan-1

February 26, 2024

## 1 Pertemuan 1 Pengantar Data Mining

### 1.1 Pengantar Dasar Python

## 2 Deskripsi Singkat

Python merupakan Bahasa pemrograman populer yang dibuat oleh Guido van Rossum. Python dirilis pada tahun 1991. Python adalah bahasa pemrograman yang sangat sederhana sehingga mudah untuk digunakan. Python dapat digunakan untuk berbagai kepentingan antara lain dapat digunakan di server untuk membuat aplikasi web, digunakan bersama perangkat lunak untuk membuat alur kerja, dapat terhubung ke sistem basis data dan dapat digunakan untuk membaca dan memodifikasi file. Python mampu menangani data besar dan melakukan matematika yang rumit. Python dapat juga digunakan untuk pembuatan prototipe cepat, atau untuk pengembangan perangkat lunak yang siap produksi. Python memiliki banyak keunggulan antara lain. * Python mampu bekerja pada platform yang berbeda (Cross platform. Python tersedia dan dapat berjalan di berbagai sistem operasi seperti Mac, Windows, Linux, Unix dll. Hal ini membuatnya menjadi cross platform dan bahasa portabel. * Python berjalan pada sistem interpreter, artinya kode dapat dieksekusi segera setelah ditulis sehingga prototyping bisa sangat cepat. * Python dapat diperlakukan dengan cara prosedural, cara berorientasi objek atau cara fungsional. * Python mudah untuk dipelajari karena menggunakan bahasa pemrograman tingkat tinggi yang ekspresif, yang artinya mudah untuk memahami bahasa dan karenanya mudah dipelajari. Python memiliki sintaks sederhana yang mirip dengan bahasa Inggris. Python juga memiliki sintaks yang memungkinkan pengembang untuk menulis program dengan garis lebih sedikit daripada beberapa bahasa pemrograman lainnya. * Python merupakan perangkat lunak atau bahasa pemrograman open source. Ini berarti Python dapat diunduh secara gratis dan digunakannya di aplikasi. Python merupakan FLOSS (Free / Libre Open Source Software), yang berarti siapa saja dapat dengan bebas mendistribusikan salinan perangkat lunak ini, membaca kode sumbernya dan memodifikasinya. * Python dilengkapi dengan pustaka standar besar yang memiliki beberapa kode praktis dan fungsi yang dapat kita gunakan saat menulis kode dalam Python. * Mendukung penanganan pengecualian. Pengecualian adalah peristiwa yang dapat terjadi selama pengecualian program dan dapat mengganggu aliran normal program. Python mendukung penanganan pengecualian yang berarti kita dapat menulis lebih sedikit kode rawan kesalahan dan dapat menguji berbagai skenario yang dapat menyebabkan pengecualian di kemudian hari. * Python mendukung manajemen memori otomatis yang berarti memori dihapus dan dibebaskan secara otomatis. Anda tidak perlu repot membersihkan memori.

## 3 Jupyter Notebook

Jupyter Notebook merupakan tool yang popular untuk mengolah data dalam Bahasa pemrograman Python. Jupyter Notebook memungkinkan untuk mengintegrasikan antara kode dengan output di dalam satu dokumen secara interaktif. Jupyter Notebook sudah otomatis terinstall ketika kita telah menginstall python dengan anaconda.

Halaman Jupyter Notebook akan muncul di lokasi folder tempat menjalankan sintaks Jupyter Notebook. Untuk membuat file pemrograman baru, dapat dilakukan dengan mengklik tab new sebelah kanan atas. Kemudian pilih Python 3.

## 4 Tipe Data

- Tipe Text : str (string)
- Tipe Numerik : int (integer), float, complex
- Tipe Barisan : list, tuple, range
- Tipe Mapping : dict (dictionary)
- Tipe Himpunan : set, frozenset
- Tipe Boolean : bool
- Tipe Binary : bytes, bytearray, memoryview

Selain tipe data diatas, bahasa pemrograman Python juga menyediakan operator aritmetika seperti :

a. •
digunakan untuk penjumlahan

b. •
digunakan untuk pengurangan

c. •
digunakan untuk perkalian

d. / digunakan untuk pembagian

e. % digunakan untuk Modulus

f. ** digunakan untuk pangkat

g. // digunakan untuk Floor division

[ ]:

[1]:
```
# Contoh operasi Matematika sederhana tipe integer
a = 5
b = 2
c = a + b
# c
print(c)
print ('Hasilnya adalah ', c) #Fungsi untuk menampilkan hasil
```

```
7
Hasilnya adalah  7
```

[2]:
```
a = 5
b = 2
c = a + b
'Hasilnya adalah', c
```

[2]:
```
('Hasilnya adalah', 7)
```

[3]:
```
# Contoh operasi Matematika sederhana tipe float
a = 3.14
b = 7
c = a * b
print(c)
```

```
21.98
```

[6]:
```
# Contoh penggunaan tipe data text
string1 = "Pengenalan Program Python"
string2 = "Mata Kuliah Data Mining"
string3 = "Program Studi Statistika dan Aktuaria UGM"
print (string1 + ' ' +string2 + ' ' + string3)
```

```
Pengenalan Program Python Mata Kuliah Data Mining Program Studi Statistika dan
Aktuaria UGM
```

[7]:
```
# Floor division

m = 10
n = 4
p = m// n
print(p)
```

```
2
```

# 5 BOOLEAN

- Boolean adalah salah satu tipe data yang ada di dalam bahasa pemrograman Python.
- Tipe data Boolean merepresentasikan salah satu dari dua nilai logika yaitu True atau False.
- Fungsi dari Boolean itu sendiri secara umum adalah untuk mengetahui kebenaran dalam suatu ekspresi apakah bernilai True atau False.

### 5.0.1 Boolean Operations

Operator Equality Operator Equality adalah operator yang berfungsi untuk memeriksa **kesamaan** atau **ketidaksamaan** nilai satu dengan yang lain.

| Operator | Arti | Penggunaa dalam Python |
|---|---|---|
| == | **sama** dengan | var_1 == var_2 |

| Operator | Arti | Penggunaa dalam Python |
|---|---|---|
| != | **tidak sama** dengan | var_1 != var_2 |

Operator Comparsion adalah operator yang berfungsi untuk **komparasi** nilai satu dengan yang lain.

| Operator | Arti | Penggunaan dalam Python |
|---|---|---|
| < | **lebih kecil** dari | var_1 < var_2 |
| > | **lebih besar** dari | var_1 > var_2 |
| <= | **lebih kecil** atau **sama dengan** | var_1 <= var_2 |
| >= | **lebih besar** atau **sama dengan** | var_1 >= var_2 |

### 5.0.2 Boolean Logic

terdapat tiga operasi logic dasar boolean yaitu: * AND * OR * NOT

---

**AND** Secara sederhana, `and` memiliki karakteristik sebagai berikut: * Nilai `TRUE` hanya bisa diperoleh jika dua nilai bernilai `TRUE` * Secara fisik adalah irisan antara dua kondisi.

Truth table:

| A | B | A and B | Arti |
|---|---|---|---|
| True | True | True | Jika A bernilai True **Dan** B bernilai True maka outputnya **True** |
| True | False | False | Jika A bernilai True **Dan** B bernilai False maka outputnya **False** |
| False | True | False | Jika A bernilai False **Dan** B bernilai True maka outputnya **False** |

| A | B | A and B | Arti |
|---|---|---|---|
| False | False | False | Jika A bernilai False **Dan** B bernilai False maka outputnya False |

---

**OR** Secara sederhana, `OR` memiliki karakteristik sebagai berikut: * Nilai `TRUE` dapat diperoleh jika salah satu kondisi bernilai `TRUE` * Secara fisik, `OR` adalah gabungan antara dua kondisi. * Gabungan merupakan **mencakupi semua nilai maupun itu irisan atau diluar dari irisan**

Truth table:

| A | B | A or B | Arti |
|---|---|---|---|
| True | True | True | Jika nilai A True **Atau** nilai B True maka outputnya True |
| True | False | True | Jika nilai A True **Atau** nilai B False maka outputnya True |
| False | True | True | Jika nilai A False **Atau** nilai B True maka outputnya True |
| False | False | False | Jika nilai A False **Atau** nilai B False maka outputnya False |

---

**NOT** Secara sederhana, `not` memiliki karakteristik sebagai berikut: * Merupakan negasi dari nilai sebelumnya. * Negasi adalah penyangkalan atau peniadaan dari suatu pernyataan. * Jika nilai sebelumnya adalah `TRUE`, maka akan menghasilkan `False` dan sebaliknya.

Truth table

| Condition | NOT |
|---|---|
| TRUE | FALSE |
| FALSE | TRUE |

- Contoh:
  - Ibu tidak membeli Lemon.
  - Maka, jika Ibu membeli Lemon, kalimat tersebut akan salah.

## **Case Study 1 : Membuat Sistem Rekomendasi Beasiswa Mahasiswa**

Disebuah kampus menerapkan sistem rekomendasi untuk mahasiwa yang mendapatkan beasiswa dengan kriteria yang ditentukan untuk mahasiwa semester 3 sebagai berikut : 1. Mahasiswa dengan Indeks prestasi atau IP minimal lebih dari atau sama dengan 3.5 ditiap semesternya 2. Mahasiswa dengan jika mendapatkan prestasi sesuai dengan bidang studi, IP tidak memengaruhi 3. Mahasiswa dengan mendapatkan prestasi dalam olahraga, harus memiliki IP minimal lebih dari atau sama dengan 3.5 minimal 1 semester.

Berikut ketentuan variabel yang harus di gunakan:

| Nama | Nama Variabel | Tipe Data |
|---|---|---|
| IP Semester 1 | ip_smt_1 | float |
| IP Semester 2 | ip_smt_2 | float |
| Prestasi Bidang Studi | get_prestasi_bs | boolean |
| Prestasi Olahraga | get_prestasi_ol | boolean |

Jika Output True maka ia Mendapatkan beasiswa, jika False tidak mendapatkan beasiwa.

Buatlah operasi boolean logic berikut:

### 5.0.3 Problem 1A

Kandidat pertama mendapatkan beasiswa dengan memiliki IP 3.5 ditiap semesternya (semester 1 & semester 2) maka buatlah boolean logic yang menggambarkan apakah ia berhak mendapatkan beasiswa

[8]:
```
ip_smt_1 = 3.5
ip_smt_2 = 3.5

ip_smt_1 >= 3.5 and ip_smt_2 >= 3.5
```

[8]:
```
True
```

### 5.0.4 Problem 1B

Kandidat kedua mendapatkan beasiswa dengan nilai IP semester 1 yaitu 3.2, IP semester 2 yaitu 2.8,dan mendapatkan prestasi bidang studi.

[9]:
```
ip_smt_1 = 3.2
ip_smt_2 = 2.8
get_prestasi_bs = True
ip_smt_1 >= 3.5 or ip_smt_2 >=3.5 or get_prestasi_bs == True
```

[9]:
```
True
```

### 5.0.5 Problem 1C

Kandidat ketiga mendapatkan beasiswa dengan prestasi di bidang olahraga, dengan nilai IP semester 1 yaitu 3.0 dan nilai IP semester 2 yaitu 3.6

[10]:
```
# Tulis jawaban dibawah ini
...
```

Klik untuk melihat kunci jawaban

```
ip_smt_1 = 3.0
ip_smt_2 = 3.6
get_prestasi_ol = True

ip_smt_1 >= 3.5 or ip_smt_2 >=3.5 and get_prestasi_bs == True
```

---

### 5.0.6 Problem 1D

Kandidat keempat tidak mendapatkan beasiswa, dengan nilai IP semester 1 yaitu 3.0 dan nilai IP semester 2 yaitu 3.49 meskipun dengan prestasi di bidang olahraga

[ ]:
```
# Tulis jawaban dibawah ini
...
```

Klik untuk melihat kunci jawaban

```
ip_smt_1 = 3.0
ip_smt_2 = 3.49
get_prestasi_ol = True

ip_smt_1 >= 3.5 or ip_smt_2 >=3.5 and get_prestasi_ol == True
```

---

# 6 Branching

Bahasa pemrograman python juga dapat melakukan conditional branching, yaitu ketika eksekusi program harus memilih ke salah satu cabang berdasarkan syarat suatu nilai boolean yang bernilai true atau false.

- Branching atau percabangan adalah salah satu bentuk kontrol pada suatu program.
- Percabangan berguna untuk mengatur jalannya program sesuai dengan suatu kondisi yang terpenuhi.
- Percabangan juga mampu membuat sebuah program menentukan suatu tindakan sesuai dengan logika/kondisi yang diberikan
- Dalam pemrograman Python, terdapat tiga keywords untuk melakukan percabangan, yaitu if, elif, dan else.

**Syntax Branching** Berikut merupakan syntax dari statement if, elif, dan else:

## 6.1 ## Case Study 2 : Pengelompokan Kelompok umur Penduduk

Sebuah Aplikasi Sensus memiliki fitur untuk mengkategorikan penduduk kelompok umur . berikut kategori kelompok umur

| Nama Kelompok Umur | Ambang batas umur |
|---|---|
| Balita | 0-5 tahun |
| Kanak-kanak | 6-11 tahun |
| Remaja | 12-25 |
| Dewasa | 26-45 |
| Lansia | >46 |

Buatlah program branching lebih dari 2 kondisi menggunakan variabel umur berdasarkan semua kelompok umur.

[11]:
```
umur = int(input('umur : '))

if  umur < 5:
    print("balita")
elif umur <= 11:
    print("kanak-kanak")
elif umur <= 25:
    print("remaja")
elif umur <= 45:
    print("dewasa")
else:
    print("lansia")
```

```
umur : 78
lansia
```

## 6.2 ## Case Study 3 : Menyiapkan Pengaturan Penyiraman Menggunakan Sensor Suhu dan Kelembapan

Sebuah Sensor yang dipasang pada sebuah perkebunan memiliki aturan untuk penyiraman berdasarkan suhu dan kelembapan, sensor dapat mendeteksi suhu dan kelembapan dengan penilaian sebagai berikut

| Penilaian Suhu (temp_type) | Penilaian Kelembapan (moisture_type) |
|---|---|
| Cold | Dry |
| Normal | Moist |
| Hot | Wet |

Alat Penyiram air akan menyala secara otomatis dengan beberapa lama durasi penyiraman yang berbeda berdasarkan aturan kombinasi antara penilaian suhu dan penilaian kelembapan. Aturan-aturan yang telah ditentukan sebagai berikut:

| Penilaian Suhu & Kelembapan | Durasi Penyiraman |
|---|---|
| Cold - Dry | 15 menit |
| Cold - Moist | 15 menit |
| Cold - Wet | 5 menit |
| Normal - Dry | 15 menit |
| Normal - Moist | 10 menit |
| Normal - Wet | 5 menit |
| Hot - Dry | 15 menit |
| Hot - Moist | 10 menit |
| Hot - Wet | 5 menit |
| None | Perangkat Error |

### 6.2.1 Problem 3A

Buatlah nested branch kondisi untuk Cold - Dry dengan aturan penyiraman yg telah ditentukan

[12]:
```
temp_type = "Cold"
moisture_type = "Dry"

if temp_type == "Cold":
  if moisture_type == "Dry":
    print("Watering on 15 minutes")
```

```
Watering on 15 minutes
```

### 6.2.2 Problem 3B

Buatlah nested branch kondisi untuk Normal - Dry, Normal - Moist, Hot - Dry, dan Hot - Moist dengan aturan penyiraman yg telah ditentukan dengan input Hot dan Dry

[ ]:
```
# Tulis jawaban dibawah ini
...
```

Klik untuk melihat kunci jawaban

```
# Tulis jawaban dibawah ini
temp_type = "Hot"
moisture_type = "Dry"

if temp_type == "Normal":
    if moisture_type == "Dry":
        print("Penyiraman 15 menit")
    elif moisture_type == "Moist":
        print("Penyiraman 10 menit")
elif temp_type == "Hot":
    if moisture_type == "Dry":
        print("Penyiraman 15 menit")
  elif moisture_type == "Moist":
        print("Penyiraman 10 menit")
```

---

## 7 List dan Tuples

- Tuple adalah tempat untuk menyimpan lebih dari satu objek dalam suatu urutan.
- List, serupa dengan tuple namun bersifat mutable yaitu elemennya dapat diubah.

[13]:
```
# contoh tuple
t = (1,2,3,4,5)
print(t)
```

```
(1, 2, 3, 4, 5)
```

[14]:
```
# contoh tuple dengan tipe data str
t = tuple('DataMining')
print(t)
```

```
('D', 'a', 't', 'a', 'M', 'i', 'n', 'i', 'n', 'g')
```

Selain itu, elemen-elemen dalam tuple dapat diakses menggunakan tanda bracket '[ ]'

[15]:
```
q = t[4]
print(q)
```

```
M
```

[16]:
```
# Contoh List
list1 = [1, 2, 3, 4, 5]
list1
```

[16]:
```
[1, 2, 3, 4, 5]
```

[17]:
```
# contoh
list2 = list('DataMining')
print(list2)
```

```
['D', 'a', 't', 'a', 'M', 'i', 'n', 'i', 'n', 'g']
```

Pada list, dapat ditambahkan elemen baru dengan menggunakan fungsi append()untuk menambah elemen ke bagian paling belakang, atau fungsi insert() untuk menambah elemen ke bagian yang diinginkan.

[18]:
```
#contoh
list2.append('B')
print(list2)
```

```
['D', 'a', 't', 'a', 'M', 'i', 'n', 'i', 'n', 'g', 'B']
```

[19]:
```
list2.insert(5, 'B')
print(list2)
```

```
['D', 'a', 't', 'a', 'M', 'B', 'i', 'n', 'i', 'n', 'g', 'B']
```

Selain menambahkan elemen, juga dapat menghapus elemen yang ada pada list

[20]:
```
#contoh
list2.pop(1)
print(list2)
```

```
['D', 't', 'a', 'M', 'B', 'i', 'n', 'i', 'n', 'g', 'B']
```

[21]:
```
#contoh
list2.remove('a')
print(list2)
```

```
['D', 't', 'M', 'B', 'i', 'n', 'i', 'n', 'g', 'B']
```

List juga dapat digunakan untuk mengurutkan data dengan menggunakan fungsi sort()

[22]:
```
# Contoh
list3 = [8,3,9,2,5,4,1,6,7]
list3.sort()
print(list3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[23]:
```
# atau diurutkan dari terbesar ke terkecil
list3.sort(reverse=True)
print(list3)
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Untuk tipe data str, mengurutkan elemen-elemen berdasarkan panjang str tersebut

[24]:
```
# contoh
list4 = ['abc', 'de', 'f','ghi','jklmn','op']
list4.sort(key=len, reverse=True)
print(list4)
```

```
['jklmn', 'abc', 'ghi', 'de', 'op', 'f']
```

Salah satu hal yang sering dilakukan terhadap data adalah mengambil subset (sebagian) dari data. Untuk hal ini digunakan notasi slicing (pemotongan), yaitu [start : stop] di samping list.

[25]:
```
#contoh
list5 = ['a', 'b', 'c', 'd', 'e','f','g']
list5[0:6]
```

[25]:
```
['a', 'b', 'c', 'd', 'e', 'f']
```

[26]:
```
list5[: 2]
```

[26]:
```
['a', 'b']
```

[27]:
```
list5[3 : ]
```

[27]:
```
['d', 'e', 'f', 'g']
```

Suatu list juga dapat terdiri dari banyak list.

[28]:
```
#Contoh
list6 = [[1,2,3],[4,5,6],[7,8,9]]
print(list6)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[31]:
```
# Untuk mengakses seluruh isi suatu list yang terdiri dari beberapa list
n = 0
for x in list6:
    n = n + 1
    print('List', n ,'=', x)
```

```
List 1 = [1, 2, 3]
List 2 = [4, 5, 6]
List 3 = [7, 8, 9]
```

Pada dasarnya, sebuah List akan berguna saat diperlukan tempat untuk menyimpan serangkaian nilai yang bisa diubah-ubah. Namun, hal ini kurang efisien jika terdapat ribuan atau puluhan ribu elemen dalam list, karena Python harus menggeser semua elemen untuk memberi ruang bagi elemen baru yang masuk.

# 8 Dictionary

Salah satu struktur data lain yang juga penting adalah tipe data dict. Hampir mirip dengan list, dict juga mutable, namun dict digunakan untuk menyimpan pasangan key-value (kunci dan nilai isinya).

[32]:
```
# Contoh
MK = {"senin": "Ekonomi, Peng. Data Mining", "selasa": "Praktikum, Asuransi␣
↪Kesehatan", "rabu": "Asuransi Umum, Dana Pensiun"}
MK.items()
```

[32]:
```
dict_items([('senin', 'Ekonomi, Peng. Data Mining'), ('selasa', 'Praktikum,
Asuransi Kesehatan'), ('rabu', 'Asuransi Umum, Dana Pensiun')])
```

[33]:
```
MK['selasa']
```

[33]:
```
'Praktikum, Asuransi Kesehatan'
```

[34]:
```
num_dict = {1: "one", 2: "two", 1: "three"}

# kita akan mengganti value pada key 1 menjadi 'ayam'
num_dict[1] = 'ayam'

print(num_dict)
```

```
{1: 'ayam', 2: 'two'}
```

# 9 Looping

What is a Looping ?

Mungkin dari kita pernah mendapat sanksi dari guru sewaktu masih sekolah untuk menulis kalimat yang sama berulang-ulang sebanyak 50 atau bahkan 100 kali. Pasti memakan waktu yang lama dan pastinya akan capek. Di bahasa pemrograman kita bisa melakukannya dengan satu kedipan mata.

`Looping` atau **perulangan** adalah metode di dalam sebuah bahasa pemrograman untuk melakukan suatu pekerjaan seperti menampilkan data, memproses data atau pekerjaan yang lainnya yang sifatnya **repetitif** atau **berulang** (**loop**). Misalnya seperti: - Menambahkan angka 100 kali - Menulis kalimat yang sama sebanyak 50 kali - Mencari channel TV yang bagus - Fotocopy sebuah KTP sebanyak 5 lembar - dll

Pada bahasa pemrograman Python ada **dua** jenis `looping`, yaitu - `for` - `while`

Perbedaan antara `for` dan `while` adalah: - Ketika menggunakan `for`, kita sudah **mengetahui** objek apa yang akan di `loop`. - Ketika menggunakan `while`, perulangan ini memiliki **kondisi yang harus dipenuhi** dan akan berhenti ketika kondisi sudah tidak terpenuhi, oleh karena itu, `while` belum diketahui banyak perulangannya.

## 9.1 For Loop

Biasanya jika kita ingin mencetak sesuatu kita menggunakan keyword print. Namun ini akan menjadi tidak efisien jika kita print kata yang berulang kali. Misalnya seperti print 10 bilangan awal secara berurutan. Jadi kode yang kita pakai

```
print(1)
print(2)
print(3)
...
print(10)
```

Bagaimana jika ingin print 50 atau 100 lagi? Pasti akan capek dan hal ini tidak efisien.

Kita bisa otomatisasi print hal yang berulang tersebut dengan `for` loop.

Struktur perulangan `for` adalah sebagai berikut

```
for iterator_variable in sequence_name:
    Statements
    . . .
    Statements
```

- keyword `for` menandakan awal dari `for` loop.
- `iterator_variabel` adalah variabel yang kita gunakan untuk menyimpan nilai pada saat kita melakukan for loop
- keyword `in` memberi tahu variabel iterator untuk mengulang elemen dalam urutan
- `sequence_name` adalah tipe data yang memiliki urutan, seperti list, set, tuple, dictionary dll.

[35]:
```
for angka in range(5):
    print(angka)
```

```
0
1
2
3
4
```

[36]:
```
for angka in range(5,11):
    print(angka)
```

```
5
6
7
8
9
10
```

[37]:
```
for angka in range(0, 10, 2):
    print(angka)
```

```
0
2
4
6
8
```

[38]:
```
for angka_1 in range (1,3):
    for angka_2 in range (1,6):
        pertambahan = angka_1 + angka_2
        print(f'{angka_1} + {angka_2} = {pertambahan}')
    print('---')
```

```
1 + 1 = 2
1 + 2 = 3
1 + 3 = 4
1 + 4 = 5
1 + 5 = 6
---
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
2 + 4 = 6
2 + 5 = 7
---
```

[39]:
```
tuple_kota = ('Surabaya', 'Malang', 'Bandung')
for kota in tuple_kota:
    print(kota)
```

```
Surabaya
Malang
Bandung
```

[40]:
```
gaji_dict = {'Dika':4_000_000, 'Rendy':3_5000_000, 'Satria':5_000_000}

for key,value in gaji_dict.items():
  print(f'Nama {key} | Gaji Rp {value}')
```

```
Nama Dika | Gaji Rp 4000000
Nama Rendy | Gaji Rp 35000000
Nama Satria | Gaji Rp 5000000
```

## 9.2 While Loop

Jika perulangan `for` sudah kita ketahui objek yang akan di loop dan banyaknya looping. Semisal kita akan melakukan looping sebanyak 10x. Maka dalam while kita tidak mengetahui berapa banyak loop akan dijalankan.

### 9.2.1 While in general

Bentuk umum while adalah:

```
while <condition>:
    # Jika kondisi benar
    run_this_statement
```

Berarti selama kondisi masih terpenuhi maka akan terus mengeksekusi blok statement

`while` loop biasanya harus disertai dengan nilai yang selalu berubah.

[41]:
```
# coba jalankan code dibawah ini
counter = 0
max = 5

while counter <= max: #jika counter masih kurang dari max, code akan tetap␣
 ↪berjalan
    print(f'Sekarang angka {counter}')
    counter += 1 #counter akan ditambahkan 1 agar perulangan bisa berhenti
```

```
Sekarang angka 0
Sekarang angka 1
Sekarang angka 2
Sekarang angka 3
Sekarang angka 4
Sekarang angka 5
```

[42]:
```
angka = 5
while angka >= 0: # "angka" disini memenuhi kondisi while, maka looping akan␣
 ↪berjalan
    print(angka)
    angka -= 1 # variabel angka akan dikurangi dengan 1 sampai kondisi di atas␣
 ↪tidak terpenuhi
```

```
5
4
3
2
1
0
```

[43]:
```
# Coba jalankan kode dibawah ini
# kita akan membuat while - else dari angka 1 - 3

numb = 1
```

```
while numb <= 3: # jika nilai numb masih kurang sama dengan tiga maka proses␣
 ↪akan terus berlangsung
    print(f'Nilai numb sekarang {numb}')
    numb += 1
else: # else akan di eksekusi karena nilai numb sudah lebih dari 3
    print(f"Nilai numb sekarang {numb} sudah lebih dari 3.")
```

```
Nilai numb sekarang 1
Nilai numb sekarang 2
Nilai numb sekarang 3
Nilai numb sekarang 4 sudah lebih dari 3.
```

---

## 9.3 Function

### 9.3.1 Membuat Sebuah Fungsi

```
def <nama_fungsi>(parameters):
    kode
```

[44]:
```
# fungsi dengan parameter
def identitas(nama, asal, umur):
    print(f'{nama} berasal dari {asal} berumur {str(umur)} tahun')

# memanggil fungsi
identitas('Astri', '22', 'Malang')
```

```
Astri berasal dari 22 berumur Malang tahun
```

[45]:
```
def diskon(promo, harga):
    if promo == True:
        potongan = 50/100
        total_harga = int(harga * potongan)
        print(total_harga)

    else:
        print(harga)
```

[46]:
```
diskon(True, 100000)
```

```
50000
```

[49]:
```
diskon(False, 100000)
```

```
100000
```

[50]:
```
# fungsi dengan keyword arguments
def luas_lingkaran(jari_jari, PI = 3.14):
```

```
    print(f"Luas lingkaran dari jari-jari {jari_jari} adalah␣
 ↪{PI*jari_jari*jari_jari}")

# memanggil fungsi
luas_lingkaran(4)
luas_lingkaran(8)
luas_lingkaran(12)
```

```
Luas lingkaran dari jari-jari 4 adalah 50.24
Luas lingkaran dari jari-jari 8 adalah 200.96
Luas lingkaran dari jari-jari 12 adalah 452.15999999999997
```

# 10 Lambda Function

Pada Python, kita juga dapat membuat fungsi sederhana tanpa harus diberi nama tertentu (hanya berupa statement). Misalnya suatu fungsi sederhana yang hanya digunakan satu kali. Untuk ini Python menyediakan kata kunci lamda yang artinya adalah fungsi anonim, tanpa nama.

### 10.0.1 Perbedaan Fungsi Lambda dan Fungsi Biasa

| Fungsi Lambda | Fungsi biasa |
|---|---|
| Tidak perlu mendefinisikan nama fungsi (anonymous) | Harus mendefinisikan nama fungsi |
| Berisi baris tunggal yang dapat mengembalikan beberapa nilai | Dapat berisi satu maupun lebih baris didalam blok fungsi |
| Kode program pendek dan singkat | Kode program yang dibangun lebih panjang dibanding fungsi lambda |
| Cocok untuk operasi pendek/manipulasi data | Cocok untuk kasus yang memerlukan banyak baris kode |
| Kode program pada fungsi lambda terkadang dapat lebih sulit dipahami | Lebih mudah dipahami karena dapat menambah keterangan dan deskripsi fungsi agar mudah dibaca |

```
lambda <parameters>: <expression>
```

Untuk membuat fungsi lambda dibutuhkan tiga bagian utama: * `lambda` : keyword untuk membuat fungsi lambda. * `parameters` : berisi param

[51]:
```
#Fungsi yang berisi perhitungan kuadrat
kuadrat = lambda num : num * num

kuadrat(2)
```

[51]: 4

[52]:
```
profit = lambda pendapatan, pengeluaran: pendapatan - pengeluaran

profit(1200000, 935000)
```

[52]:
```
265000
```

## 11 Import

Python juga menyediakan berbagai modul serta package yang dapat dipanggil sewaktu-waktu. Modul dapat berupa modul standar python, atau modul yang dibuat sendiri. Perintah yang digunakan untuk memanggil modul tersebut adalah import.

[53]:
```
# Contoh
import math as m
y = m.ceil(49.5)
print(y)
```

```
50
```

## 12 Library

Library pada Python merupakan sebutan untuk kode program tambahan yang digunakan untuk kebutuhan tertentu. Python mempunyai lebih dari 140.000 library yang dikembangkan melalui open source project. Beberapa contoh library pada Python adalah * NumPy * Pandas * Seaborn * Pyplot * Scikit - learn * MLXtend * Keras

## 13 NumPy

Meskipun Python telah memiliki kemampuan untuk mengolah data dalam bentuk list dan tuple, namun hal ini masih kurang efisien untuk mengolah data untuk multidimension array (array yang berdimensi banyak) dibandingkan dengan bahasa pemrograman lain seperti C++ atau Java.

Sehingga, dibuatlah suatu library baru yaitu Numerical Python atau sering dikenal dengan NumPy.

Array dalam NumPy sebenarnya mirip dengan List dan Tuple, bedanya setiap elemen dalam array haruslah bertipe sama. Array satu dimensi dapat dianggap sebagai satu baris variabel yang sama, sementara array dua dimensi dapat dilihat sebagai tabel dengan banyak baris dan banyak kolom. Pada Python untuk menyatakan dimensi suatu array digunakan istilah Rank.

[54]:
```
# contoh
import numpy as np
x = np.array([1,3,5,7,9,11,13,17,23,29])
print(x)
```

```
[ 1  3  5  7  9 11 13 17 23 29]
```

[55]:
```
#untuk mengecek ada berapa banyak elemen yang terdapat dalam array
print(x.size)
```

```
10
```

[56]:
```
#mengambil elemen tertentu
x[4:8]
```

[56]:
```
array([ 9, 11, 13, 17])
```

[57]:
```
#Mengubah elemen tertentu
x[7] = 0
print(x)
```

```
[ 1  3  5  7  9 11 13  0 23 29]
```

[58]:
```
# Contoh array 2-dimensi
y = np.array([[1,2,3],[4,5,6]])
print(y)
```

```
[[1 2 3]
 [4 5 6]]
```

[59]:
```
print(y.shape)
```

```
(2, 3)
```

Salah satu alasan penggunaan NumPy adalah untuk efisiensi dalam pengelolaan memori komputer dibandingkan dengan list.

[80]:
```
#menghitung pv dari tabel
f = np.array([100000,400000,600000])      #input data
i = np.array([0.03,0.06,0.08])
n = np.array([2,4,6])

def pv(f,i,n):     #Present value
    v = f / (1+i) ** n            #persamaan present value
    return v

print(pv(f, i, n))
```

```
[ 94259.59091338 316837.46529521 378101.77612986]
```

## 14 Pandas

Pandas dapat membuat sebuah file ke dalam tabel virtual menyerupai spreadsheet. Pandas juga berfungsi mengolah suatu data seperti Teknik join, distinct, group by, agregasi dan teknik lainnya. Kelebihan dari Pandas adalah dapat membaca file dari berbagai format seperti .txt, .csv, .xlsx, dan .tsv

[71]:
```
import pandas as pd
import numpy as np
df = pd.read_excel("/content/Pertemuan 1_PDM.xlsx")
df.head()
```

[71]:

| | No | NIU | Nama | Prodi |
|---|---|---|---|---|
| 0 | 1 | 492729 | AMALIA NUR ZAHRO | S1 STATISTIKA |
| 1 | 2 | 492786 | Immanuella Rere Kusumawardhani | S1 STATISTIKA |
| 2 | 3 | 492812 | Adam Adhitya Prayoga | S1 ILMU AKTUARIA |
| 3 | 4 | 492912 | REVALDY HAZZA DANISWARA | S1 ILMU AKTUARIA |
| 4 | 5 | 493047 | Gabriella Yoanda Pelawi | S1 STATISTIKA |

## 15 Pyplot dan Seaborn

Salah satu library yang dapat digunakan dalam melakukan visualisasi data alah Pyplot (yang disediakan di dalam library lain bernama Matplotlib).

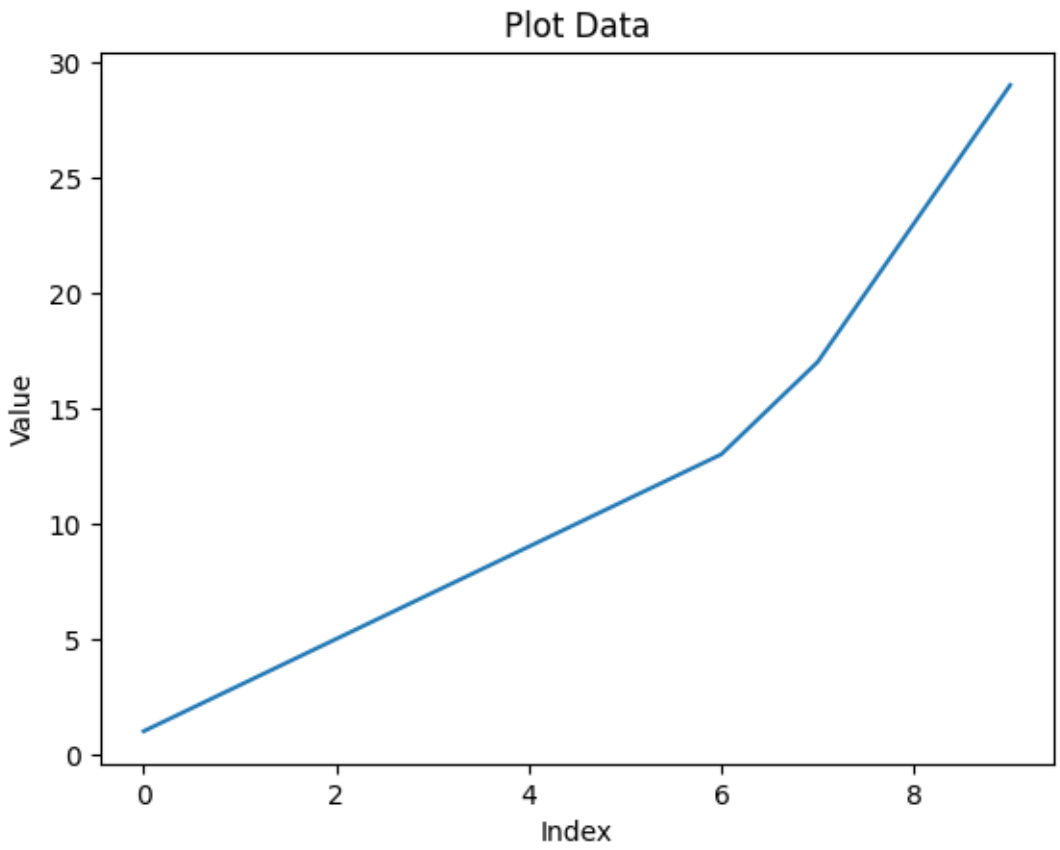
Dengan Matplotlib dapat digunakan untuk menampilkan histogram, bar chart, line chart, dll.

Selain itu, library lainnya adalah seaborn yang merupakan turunan dari Matplotlib. Seaborn dapat digunakan untuk membuat grafik statistik misalnya heatmap dan boxplot

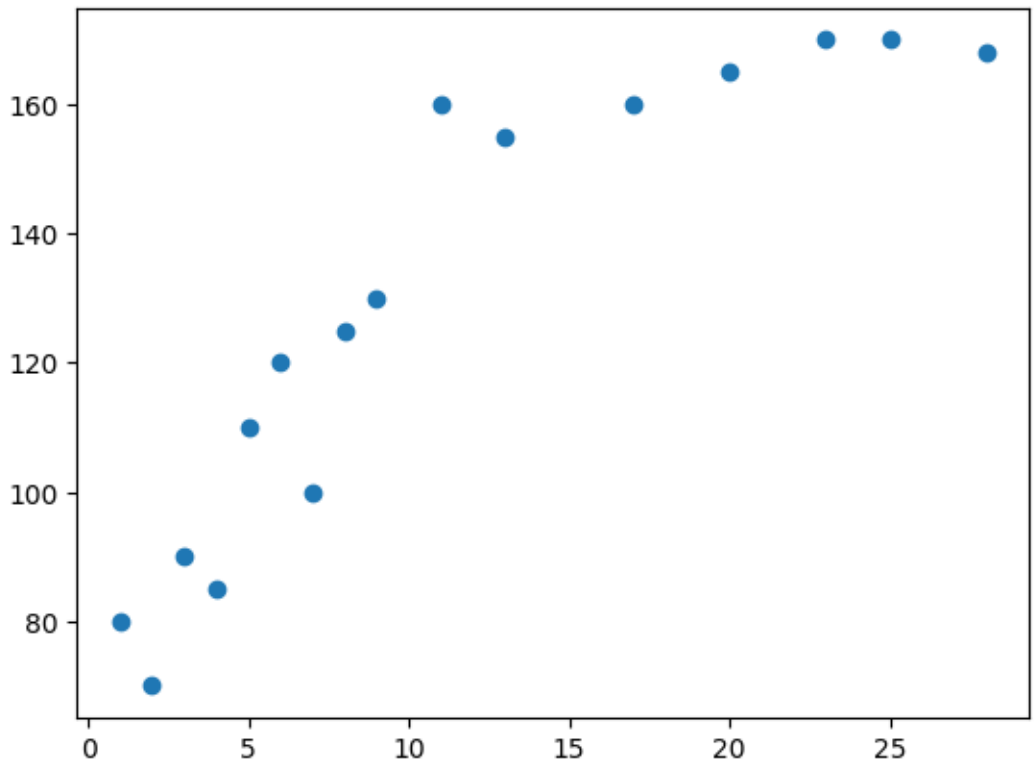
[81]:
```
import numpy as np
import matplotlib.pyplot as plt

data = np.array([1, 3, 5, 7, 9, 11, 13, 17, 23, 29])
plt.plot(data)  # Plot data langsung dari array

plt.xlabel('Index')  # Label sumbu x
plt.ylabel('Value')  # Label sumbu y
plt.title('Plot Data')  # Judul plot
plt.show()
```




[82]:
```
# Contoh
import numpy as np
import matplotlib.pyplot as plt
usia = np.array([1,2,3,4,5,6,7,8,9,11,13,17,20,23,25,28])
tinggi = np.array([80,70,90,85,110,120,100,125,130,160,155,160,165,170,170,168])
plt.scatter(usia,tinggi)
plt.show()
```




## 16 Scikit - learn

Saat ini, Python merupakan salah satu bahasa pemrograman yang sangat populer untuk keperluan Mechine Learning. Salah satu alasan utamanya adalah karena adanya library scikit-learn. Library ini menyediakan banyak modul untuk pembuatan model, baik supervised maupun unsupervised. Berikut beberapa modul yang sering digunakan.

[83]:
```
from tabulate import tabulate
#Membuat tabel
data = [["Linear Regression", "sklearn.linear_model.LinearRegression",␣
 ↪'Prediksi Numerik, supervised'],
       ["Logistic Regression", 'sklearn.linear_model.
 ↪LogisticRegression','klasifikasi,supervised'],
       ["Decision tree", 'sklearn.tree.
 ↪DecisionTreeClassifier','Klasifikasi,Supervised'],
       ["Random Forest", 'sklearn.ensemble.
 ↪RandomForestClassifier','Klasifikasi,Supervised'],
      ["Naive Bayes",'sklearn.naive_bayes',"klasifikasi,supervised"],
      ['K-Means','sklearn.cluster.KMeans','Klusterisasi,Unsupervised'],
      ['Neural Network','sklearn.neural_network','Regresi dan klasifikasi,␣
 ↪supervised'],
```

```
       ['SVM','sklearn.svm','Regresi dan klasifikasi, supervised']]

#Nama kolom
col_names = ["Algoritma", "Nama kelas untuk diimpor", "Penggunaan"]

print(tabulate(data, headers= col_names))
```

```
Algoritma            Nama kelas untuk diimpor                 Penggunaan
-------------------  ---------------------------------------
-----------------------------------
Linear Regression    sklearn.linear_model.LinearRegression    Prediksi Numerik,
supervised
Logistic Regression  sklearn.linear_model.LogisticRegression
klasifikasi,supervised
Decision tree        sklearn.tree.DecisionTreeClassifier
Klasifikasi,Supervised
Random Forest        sklearn.ensemble.RandomForestClassifier
Klasifikasi,Supervised
Naive Bayes          sklearn.naive_bayes
klasifikasi,supervised
K-Means              sklearn.cluster.KMeans
Klusterisasi,Unsupervised
Neural Network       sklearn.neural_network                   Regresi dan
klasifikasi, supervised
SVM                  sklearn.svm                              Regresi dan
klasifikasi, supervised
```

Selain itu, Scikit-learn juga menyediakan banyak fasilitas untuk melakukan manipulasi data, beberapa diantaranya adalah

[84]:
```
data1 = [['sklearn.model_selection','untuk memilih training dan test dataset'],
       ['sklearn.metrics','untuk melakukan pengukuran terhadap berbagai kinerja␣
 ↪model'],
       ['sklearn.preprocessing','untuk melakukan berbagai transformasi terhadap␣
 ↪data mentah'],
       ['sklearn.feature_extraction','untuk mengambil feature dari text dan␣
 ↪gambar']]

#Nama kolom
col_names = ["Algoritma","Penggunaan"]

print(tabulate(data1, headers= col_names))
```

```
Algoritma                  Penggunaan
-------------------------
------------------------------------------------------------
sklearn.model_selection    untuk memilih training dan test dataset
sklearn.metrics            untuk melakukan pengukuran terhadap berbagai kinerja
```

```
model
sklearn.preprocessing       untuk melakukan berbagai transformasi terhadap data
mentah
sklearn.feature_extraction  untuk mengambil feature dari text dan gambar
```