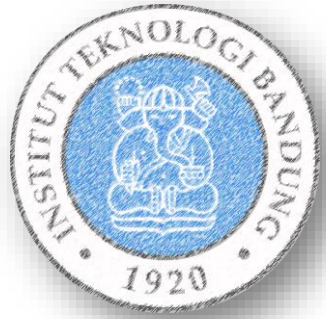


KU1102/Pengenalan Komputasi Dasar Pemrograman dengan Python 3

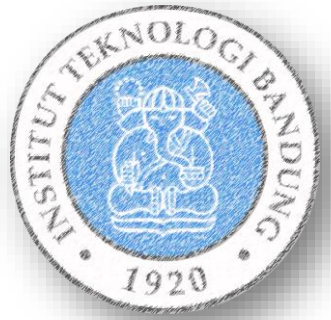
Tim Penyusun Materi Pengenalan Komputasi
Institut Teknologi Bandung © 2019



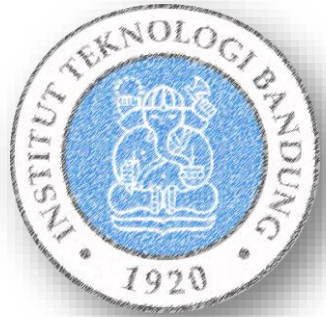


Python

- Bahasa programming tingkat tinggi, direlease oleh Guido van Rossum pada tahun 1991
- Mendukung berbagai paradigma pemrograman. Dalam kuliah ini, hanya akan menggunakan paradigma procedural.
- Interpreter yg tersedia pada beragam sistem operasi:
 - Indentasi untuk menandai blok program
 - **case sensitive** → perbedaan huruf besar dan kecil berpengaruh
- Python adalah bahasa pemrograman yang **strongly and dynamic typed**
 - **Strongly typed**: jika sebuah variabel sudah di-*assign* dengan value bertipe tertentu, maka hanya dioperasikan sesuai type value tersebut
 - **Dynamic typed**: type variabel dapat diubah pada saat run-time



Struktur Dasar Program Prosedural

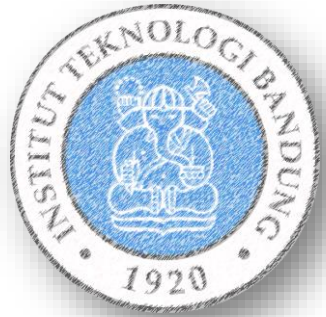


Struktur Dasar Algoritma

Program <JudulProgram>
{ Spesifikasi Program }

KAMUS
{ Deklarasi type, variabel, konstanta, fungsi, prosedur }

ALGORITMA
{ Deretan langkah algoritmik untuk penyelesaian persoalan }
{ Ditulis dengan pseudocode atau flowchart }

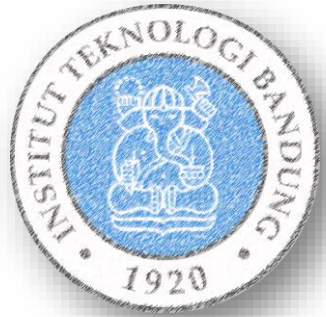


Struktur Dasar Program Python

```
# Program <JudulProgram>
# Spesifikasi Program

# KAMUS
# Penjelasan dalam bentuk komentar
# Deklarasi variabel, konstanta, fungsi, prosedur

# ALGORITMA
# Deretan langkah algoritmik untuk penyelesaian persoalan
```



Program Pertama

- Buatlah program untuk menuliskan “Hello, World!” ke layar.

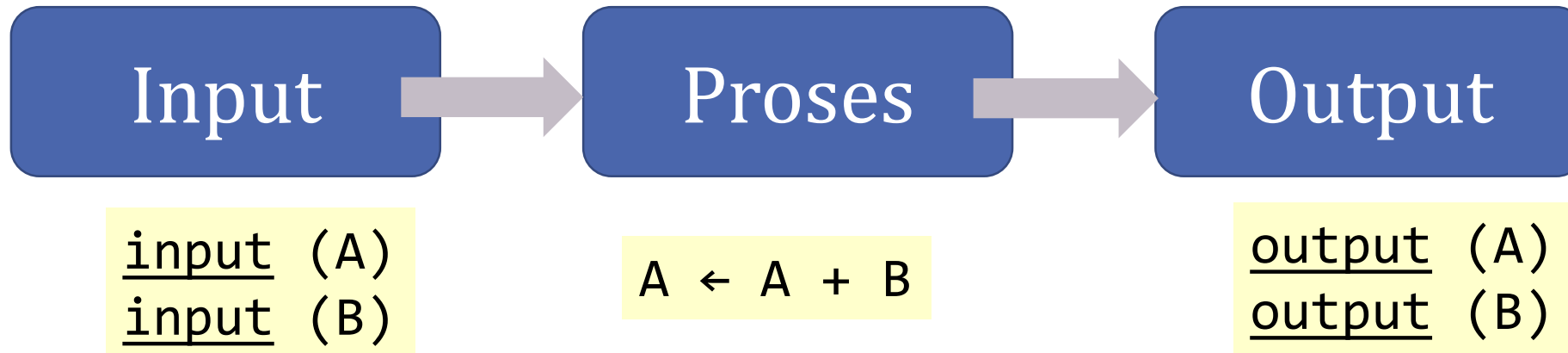
print adalah perintah untuk mencetak teks ke layar/monitor

```
# Program HelloWorld
# Mencetak Hello, World! ke layar

# KAMUS
# belum diperlukan

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

Input – Proses – Output



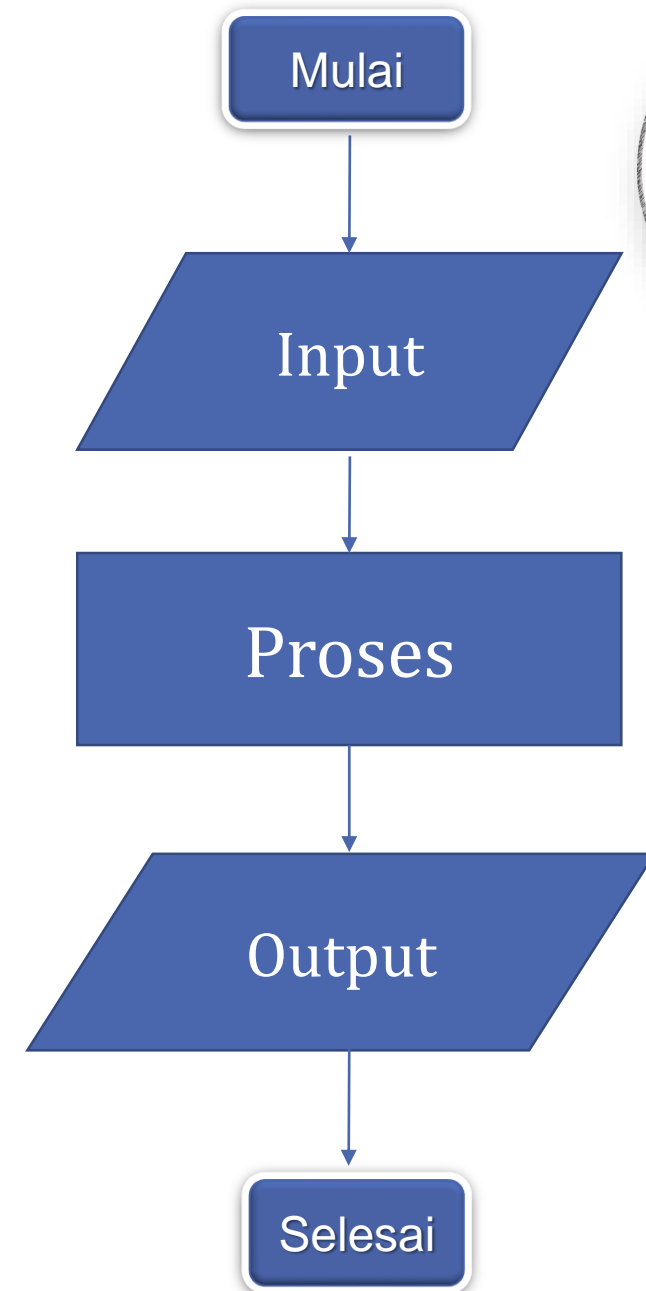
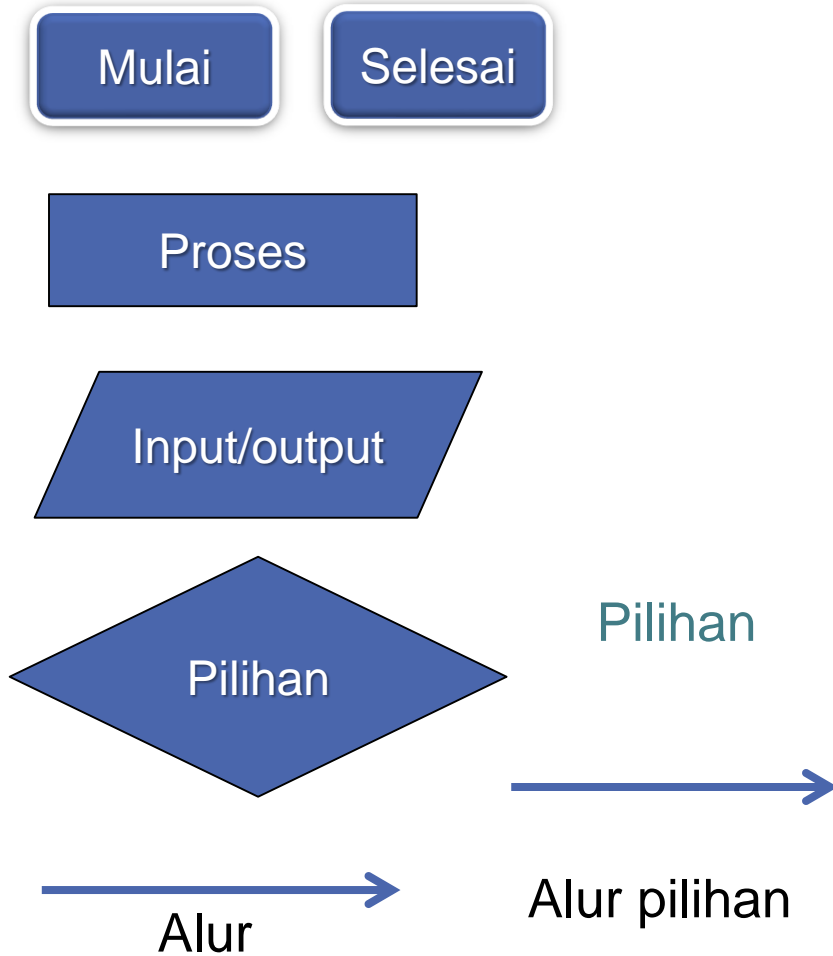
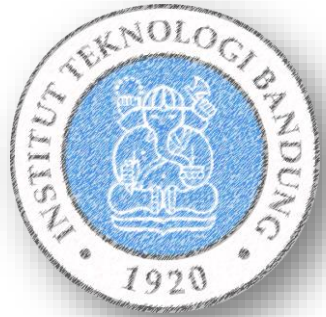
Python

```
A = int(input())  
B = int(input())
```

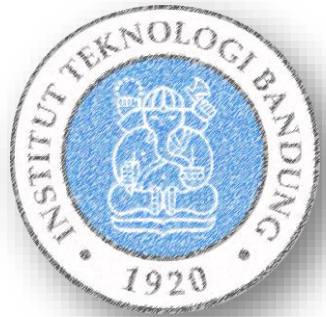
```
A = A + B
```

```
print(A)  
print(B)
```

Flow Chart



Struktur Dasar Program



Program Test

{ Spesifikasi Program: menghitung $A + B$ }

KAMUS

{ Deklarasi variabel }

$A, B : \text{integer}$

ALGORITMA - Notasi Algoritmik

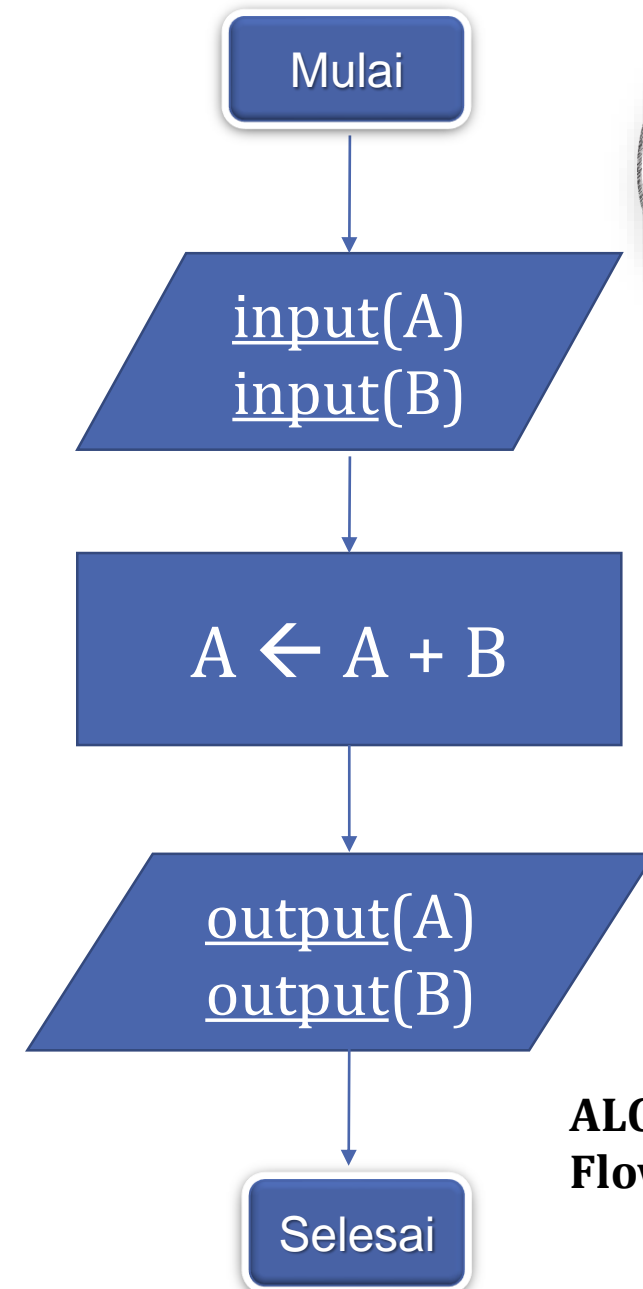
input(A)

input(B)

$A \leftarrow A + B$

output(A)

output(B)



**ALGORITMA -
Flowchart**

Contoh Program Python

```
# Program Test
# Spesifikasi : Menghitung nilai A dan B

# KAMUS
# A : int
# B : int

# ALGORITMA
A = int(input()) # input
B = int(input())

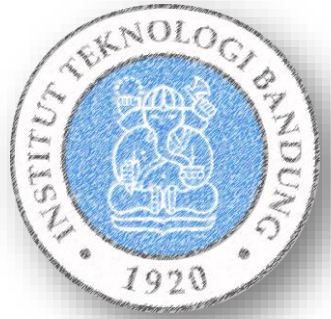
A = A + B        # proses

print(A)         #output
print(B)
```

Judul Program + spesifikasi, dituliskan dalam komentar

KAMUS: deklarasi variabel A dan B (dalam komentar)

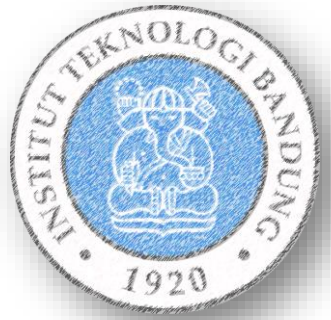
ALGORITMA: Input, Proses, Output



Komentar

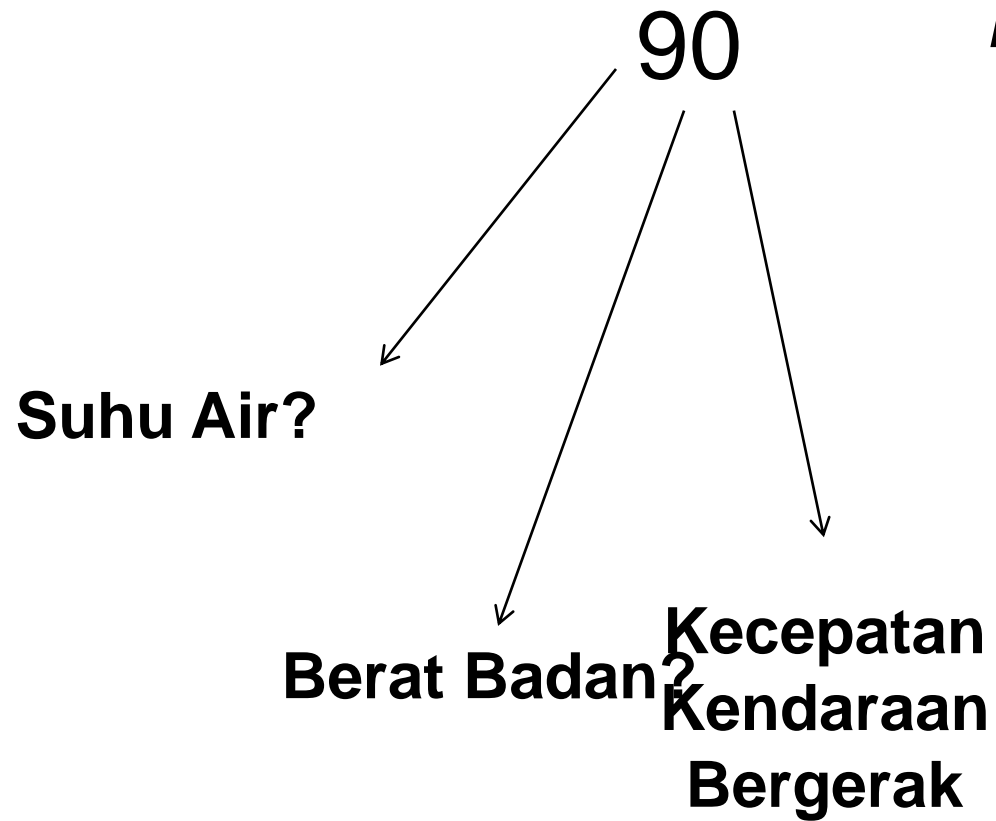
- Dalam bahasa pemrograman komentar adalah bagian program yang tidak dieksekusi
 - Bagian ini hanya digunakan untuk memberikan penjelasan suatu langkah, rumus ataupun bisa hanya berupa keterangan
- Dalam Python komentar dituliskan per baris diawali dengan #
- Contoh:

ini komentar



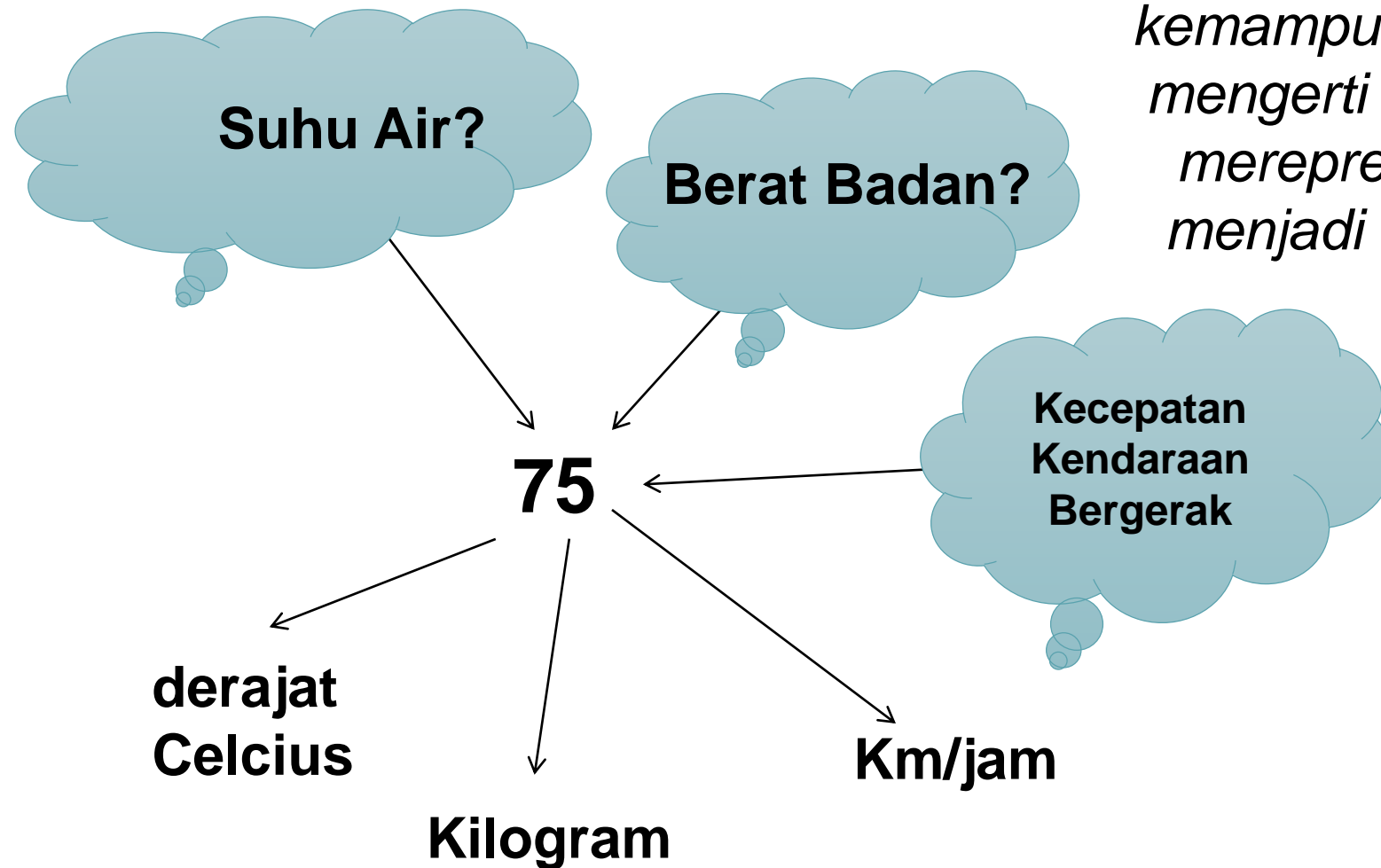
Data

Abstraksi Data

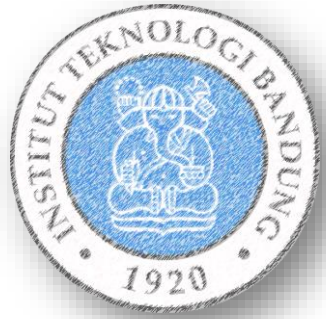


*kemampuan kita untuk
menginterpretasikan
suatu data dengan
konteks masalahnya*

Persoalan Abstraksi Data

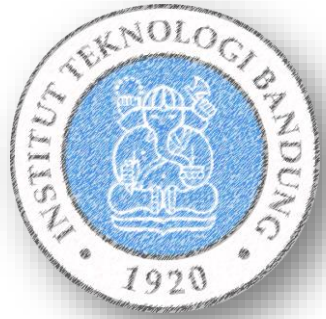


*kemampuan kita untuk
mengerti konteks dan
merepresentasikan
menjadi bentuk lain.*



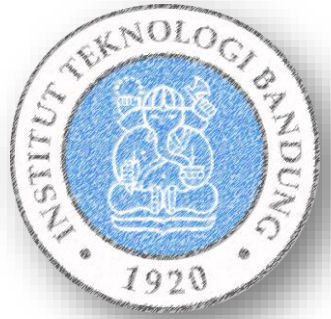
Bagian Kamus

- Bagian **Kamus** dipakai untuk mendeklarasikan nama-nama yang digunakan dalam program
- Nama-nama merepresentasikan **data** yang digunakan dalam program
- Python adalah bahasa pemrograman yang ***loosely typed***
 - Tidak perlu mendeklarasikan secara eksplisit tipe data dari variabel
- Namun demikian...
 - Dalam menggunakan variabel tetap harus diketahui dengan baik tipe data apa didefinisikan terhadap variabel tersebut
 - Untuk itu, bagian KAMUS tetap harus dinyatakan walaupun hanya dalam bagian komentar



Tipe Data (1)

- Setiap data memiliki jenis yang berbeda-beda
 - Data **umur** seseorang berbeda dengan data **nama**
 - Data umur dibentuk dari kumpulan angka
 - Data nama dibentuk dari serangkaian huruf
 - Untuk setiap jenis data juga memiliki rentang (range) yang berbeda
 - Data umur rentangnya antara 1 sampai 100 (bila diasumsikan bahwa umur seseorang tidak lebih dari 100).
 - Data nama rentangnya mulai dari 1 sampai 50 (bila di anggap nama tidak ada yang melebihi 50 huruf)



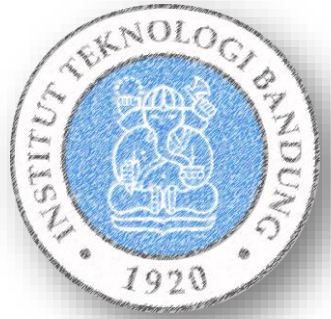
Tipe Data (2)

- Nilai yang diperbolehkan untuk variabel tergantung pada tipe data-nya
- Tipe data mendefinisikan himpunan nilai-nilai tertentu, misalnya:
 - Tipe data integer : himpunan nilai yang terdiri atas bilangan bulat (negatif, 0, positif)
 - Tipe data boolean: himpunan nilai yang terdiri atas nilai true dan false

Tipe Data Dasar/Primitif

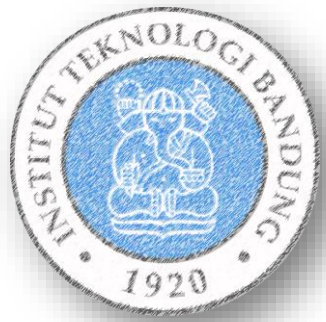
- Disediakan oleh bahasa pemrograman

Python	Domain Nilai
bool	Nilai boolean: True ; False
numbers	Nilai-nilai numerik. Jenis nilai numerik: <ul style="list-style-type: none">• int : integer/bilangan bulat bertanda (+/-). Contoh: 1; -144; 999; 0• float : floating point (real). Contoh: 3.14; 4.01E+1• complex : bilangan kompleks → tidak akan digunakan di kelas ini
string	Kumpulan karakter/huruf, ditandai dengan kutip tunggal atau kutip ganda. Contoh: 'xcxcx'
char	Character: karakter/huruf, ditandai dengan kutip tunggal; Contoh: 'A'; '#'; 'b'



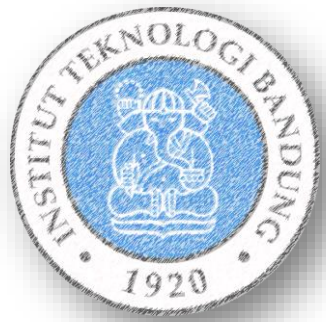
Contoh Penentuan Tipe Data Variabel

- Umur → Integer contoh: 25; 44; 35
- Kota → String, contoh: "Jakarta"; "Bandung"
- Nama → String, contoh: "Budi"; "Ali"
- Suhu → Integer atau float, contoh: 37.5; 100
- Luas → Integer atau float, contoh: 400; 43.5
- BeratBadan → Integer atau float, contoh: 60.5; 75
- NIM → Integer atau string?, contoh: 15812001



Variabel (1)

- **Variabel** digunakan menyimpan suatu nilai yang ber-"tipe data" tertentu sesuai dengan deklarasi
- Merepresentasikan suatu makna di dunia nyata yang ingin diolah dalam program, misalnya:
 - **Sum** : jumlah beberapa angka
 - **Max** : nilai maksimum
- Penggunaan variabel:
 - deklarasi (supaya nama dikenal dan diketahui tipe datanya),
 - inisialisasi dan manipulasi nilai



Variabel (2)

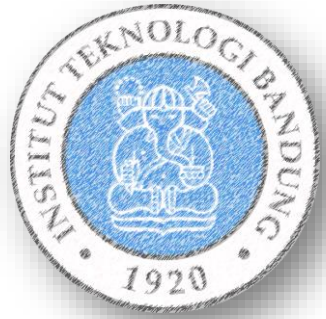
- Contoh deklarasi dan inisialisasi variabel:

Python

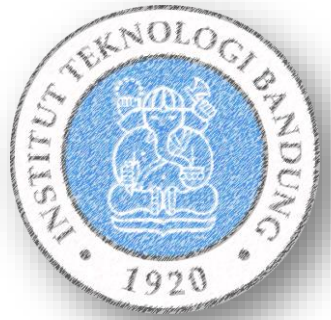
```
# KAMUS
# i : int
# A : int

# ALGORITMA
...
i = 100
A = i * 50
....
```

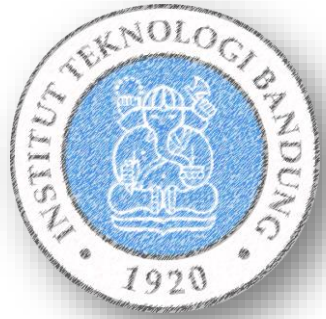
Membuat Nama Variabel yang Benar dan “baik”



- Nama variabel harus dimulai dengan huruf dan dapat diikuti dengan huruf lagi dan angka
 - Tidak boleh ada karakter lain, kecuali: *underscore* (`_`)
- Dalam nama variabel tidak boleh dipisahkan oleh spasi
- Cari nama variabel yang bisa dimengerti dan tidak membingungkan
 - Contoh: **sum** adalah untuk jumlah, bertipe integer. Jangan gunakan untuk data bertipe lain
- Python adalah bahasa yang ***case sensitive***: Kesalahan penulisan huruf besar dan kecil menyebabkan error



Assignment dan Input/Output



Pemberian Nilai

- Suatu besaran (dengan tipe tertentu), misalnya variabel, yang telah dikenal dapat diberi **nilai/harga**
- Pemberian nilai:
 - Pemberian nilai langsung atau disebut sebagai ***assignment***
 - Contoh: **A = 10**
 - Dibaca dari piranti masukan (perintah input)
 - Contoh: **A = input()**

Assignment

- **Assignment:** Pemberian nilai suatu variabel
- Ruas kiri harus **variable**
- Ruas kanan harus **ekspresi/nilai/variabel** yang sudah jelas nilainya

Python

`<RuasKiri> = <RuasKanan>`

Contoh:

`i = 10`

`Nama = "Maya"`

`X = i + 10`

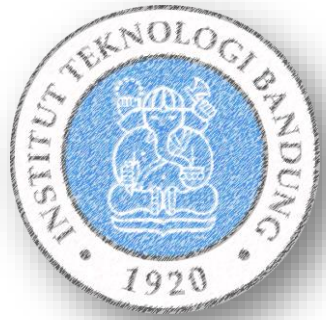
Nilai X di-
assign dengan
ekspresi

Input/Output (1)

- Perintah **input**: pemberian nilai **variabel** dari piranti masukan, misal: keyboard → dibaca atas masukan dari pengguna
- Perintah di Python: **input('<perintah>')**
 <perintah> dapat diganti dengan kalimat pengantar input
- Contoh:

```
A = input()                # A bertipe string
B = input("Masukkan angka =") # B bertipe string
C = int(input())           # C bertipe integer
D = float(input("Masukkan angka =")) # D bertipe float
```

Type checking: memastikan nilai yang dimasukkan dalam type yang tepat (gunakan type conversion)



Type Conversion

Beberapa fungsi *type conversion* yang penting diketahui:

No.	Function & Description
1	int(x) Mengkonversi x menjadi integer
2	float(x) Mengkonversi x menjadi nilai floating point (real)
3	str(x) Mengkonversi objek x menjadi representasi stringnya
4	chr(x) Mengkonversi sebuah integer x menjadi character
5	ord(x) Mengkonversi sebuah character x menjadi nilai integernya

Input/Output (2)

- Perintah **output**: penulisan nilai (variabel/konstanta/hasil ekspresi) ke piranti keluaran, misal: monitor
- Perintah di python: **print**
- Contoh:

print(A) # menulis isi variabel A ke layar

print("Hello") # menulis Hello ke layar

print(A * 4) # menulis hasil perkalian A*4

print("Hello World!" + str(a)) # menulis Hello World! <nilai a>

Mengkonversi nilai a (bertipe lain) menjadi string
+ adalah operator konkatenasi string

Latihan

- Tentukan untuk setiap baris (yang diberikan nomor dalam komentar) dari potongan program Python berikut, manakah yang merupakan assignment yang tepat.
- Jika tidak tepat, berikan alasannya.

```
# Program Latihan
# Latihan type data dan assignment

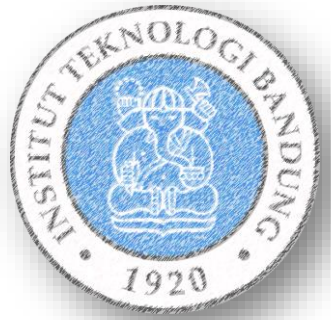
# KAMUS
# IA : int
# FA, FB : float
# SA, SB : string
# BA : bool
# CA, CB : char

# ALGORITMA
IA = 10                # (1)
FA = 3.45              # (2)
FB = 4.567             # (3)
FB = IA                # (4)

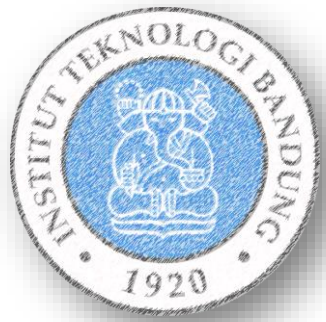
SA = "ITBJAYA"         # (5)
SA = SB                # (6)

CA = 'C'               # (7)
CA = "MAJUTERUS"      # (8)

BA = True              # (9)
BA = "#"               # (10)
```



Ekspresi

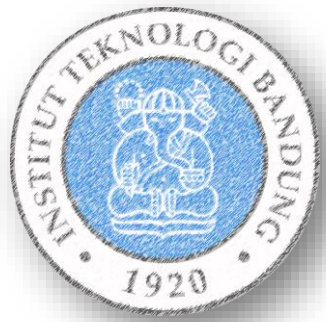


Ekspresi

- **Ekspresi** adalah kombinasi dari satu atau lebih variabel, konstanta, operator, dan fungsi yang bermakna menurut aturan suatu bahasa pemrograman dan menghasilkan suatu nilai dalam suatu type tertentu
- **Operator** adalah suatu fungsi standar yang disediakan dalam bahasa pemrograman untuk melakukan beberapa hal dasar seperti perhitungan aritmatika, logika, dan relasional.
- Struktur umum ekspresi [biner]: `<operan1> <operator> <operan2>`
- Hasil dari operasi bergantung pada tipe data operan
- Operan dapat berupa nilai, variable, konstanta, atau ekspresi lain

Jenis Ekspresi

- Jenis ekspresi menurut *arity* dari operator:
 - Ekspresi **biner**: bentuk dasarnya adalah operasi dengan 2 operan
 - Contoh: $A + 5$
 - Ekspresi **uner**: bentuk dasarnya adalah operasi dengan 1 operan
 - Contoh: `not (found)`
- Jenis ekspresi menurut tipe data yang dihasilkan:
 - Ekspresi **aritmatika**: operan bertipe numerik (`int/float`) dan menghasilkan nilai numerik
 - Ekspresi **relasional**: operan bertipe numerik (`int/float`) dan menghasilkan nilai `bool/logika`
 - Ekspresi **logika**: operan bertipe `bool/logika` dan menghasilkan nilai `bool/logika`

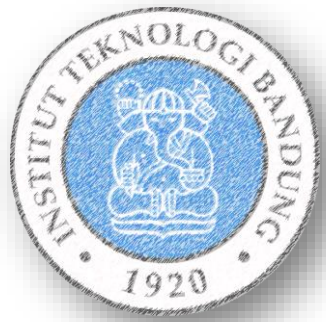


Operator Tipe Dasar (1)

Operator Aritmatika

Jika **a = 10** dan **b = 21**, maka:

Operator	Description	Type Operan	Example
+	Penjumlahan: menambahkan nilai kedua operan	int, float	$a + b = 31$
-	Pengurangan: mengurangi nilai operan kiri dengan nilai operan kanan	int, float	$a - b = -11$
*	Perkalian: mengalikan nilai kedua operan	int, float	$a * b = 210$
//	Pembagian bulat: Jika operan adalah int, maka hasil operasi adalah pembagian bulat	int	$b // a = 2$
/	Pembagian riil: Jika operan adalah float, maka hasil operasi adalah pembagian bilangan float	int, float	$b / a = 2.1$
%	Modulo: sisa hasil pembagian bulat	int	$b \% a = 1$
**	Pangkat: mengangkat operan kiri dengan operan kanan	int, float	$10 ** 2 = 100$

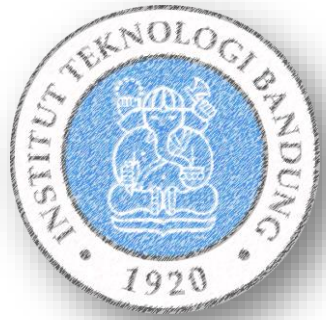


Operator Tipe Dasar (2)

Operator Relasional

Jika **a = 10** dan **b = 21**, maka:

Operator	Description	Type Operan	Example
==	Jika nilai kedua operan sama, maka menghasilkan true (tidak berlaku untuk bilangan riil)	int, char, string, bool	(a == b) menghasilkan false
!=	Jika nilai kedua operan tidak sama, maka menghasilkan true	int, float, char, string, bool	(a != b) menghasilkan true
>	Jika nilai operan kiri lebih besar dari operan kanan, maka menghasilkan true	int, float, char, string	(a > b) menghasilkan false
<	Jika nilai operan kiri lebih kecil dari operan kanan, maka menghasilkan true	int, float, char, string	(a < b) menghasilkan true
>=	Jika nilai operan kiri lebih besar dari atau sama dengan operan kanan, maka menghasilkan true	int, float, char, string	(a >= b) menghasilkan false
<=	Jika nilai operan kiri lebih kecil dari atau sama dengan operan kanan, maka menghasilkan true	int, float, char, string	(a <= b) menghasilkan true

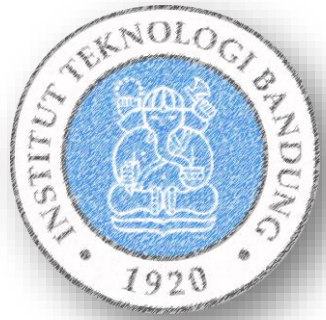


Operator Tipe Dasar (3)

Operator Logika

Jika **a = true** dan **b = false**, maka:

Operator	Description	Type Operan	Example
and	Logika AND: Jika kedua operan bernilai true, maka menghasilkan true.	bool	(a and b) menghasilkan false
or	Logika OR: Jika setidaknya salah satu dari kedua operan bernilai true, maka menghasilkan true.	bool	(a or b) menghasilkan false
not	Logika NOT/negasi: Untuk membalik nilai logika dari operannya.	bool	not(a) menghasilkan false

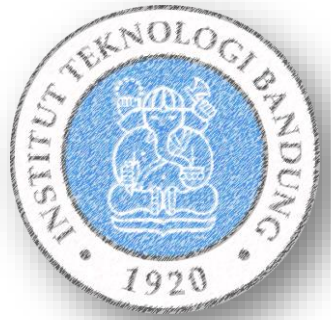


Operator Tipe Dasar (4)

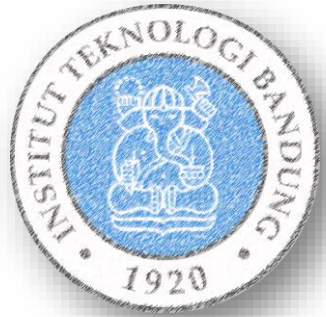
Operator Assignment

Jika **a = 10** dan **b = 21**, maka:

Operator	Description	Type Operan	Example
<op>=	<op> adalah + - * / % Meringkas operasi: A = A <op> B menjadi A <op>= B	int, float	a+=b; maka a = 31 (setara a = a + b) a*=b; maka a = 210 (setara a = a * b)



Aksi Sekuensial

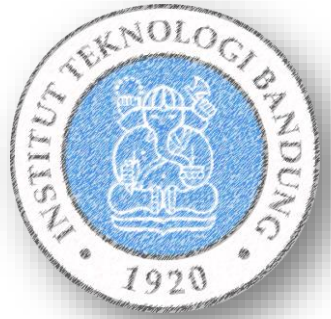


Struktur Dasar Algoritma

Program <JudulProgram>
{ Spesifikasi Program }

KAMUS
{ Deklarasi type, variabel, konstanta, fungsi,
prosedur }

ALGORITMA
{ Deretan langkah algoritmik untuk penyelesaian
persoalan }
{ Ditulis dengan pseudocode atau flowchart }

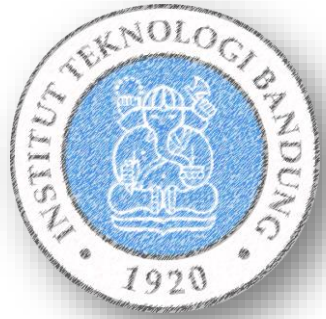


Struktur Dasar Program Python

```
# Program <JudulProgram>
# Spesifikasi Program

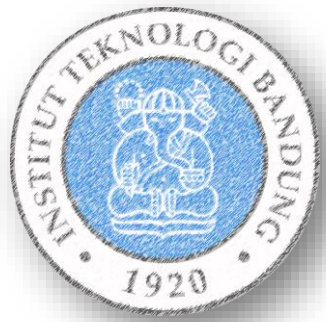
# KAMUS
# Penjelasan dalam bentuk komentar
# Deklarasi type, variabel, konstanta, fungsi, prosedur

# ALGORITMA
# Deretan langkah algoritmik untuk penyelesaian # persoalan
```



Bagian Algoritma dari Program

- Adalah bagian program dalam bentuk teks algoritmik yang berisi instruksi atau pemanggilan aksi
- Teks algoritmik tsb. dapat berupa:
 - Perintah dasar: Input/Output, assignment
 - Perintah perintah yang berurutan
 - Analisis kasus (jika-maka)
 - Pengulangan... dll.
- Dalam Bahasa Python, setiap instruksi ditulis per baris
 - Jika lebih dari 1 instruksi dituliskan pada satu baris, maka setiap instruksi dipisahkan oleh titik koma (;)
 - Contoh: `nama = input(); print(nama)`



Aksi Sekuensial

- **Aksi sekuensial:** sederetan instruksi primitif dan/atau aksi yang akan dilaksanakan (dieksekusi) oleh komputer berdasarkan urutan penulisannya
- Setiap aksi akan mengubah status dari program
 - Jadi setiap aksi sekuensial harus ada awal dan akhir.
 - Dengan kata lain, suatu program harus dimulai dan suatu ketika harus berakhir
- Instruksi ditulis terurut sesuai penulisan per baris
- Perhatikan bahwa:
 - ada program yang akan berubah jika urutan baris instruksinya berubah
 - dan ada juga program yang tidak berubah jika urutan baris instruksinya berubah

Urutan instruksi tidak mengubah hasil eksekusi...

```
# Program Test
# KAMUS
# i : int
# x : float
```

```
# ALGORITMA
i = int(input())
x = 100.75
```

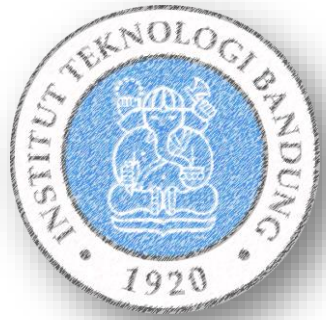
```
print(x)
print(i*2)
```

```
# Program Test
# KAMUS
# i : int
# x : float
```

```
# ALGORITMA
x = 100.75
i = int(input())
```

```
print(x)
print(i*2)
```

Hasil eksekusi **tidak berubah**, walaupun urutan instruksi diubah



Urutan instruksi mengubah hasil eksekusi...

```
# Program Test
# KAMUS
# i : int
# x : float

# ALGORITMA
i = int(input())
x = 100.75
```

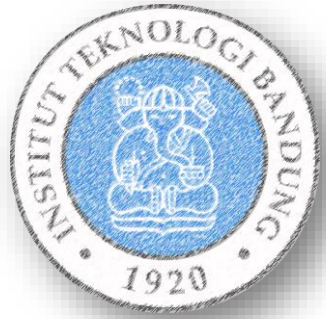
```
print(x)
print(i*2)
```

```
# Program Test
# KAMUS
# i : int
# x : float

# ALGORITMA
i = int(input())
x = 100.75
```

```
print(i*2)
print(x)
```

Hasil eksekusi **berubah** karena urutan instruksi diubah



Blok Program (1)

- Sederetan instruksi yang dieksekusi secara sekuensial dikelompokkan dalam blok program
- Dalam Python, satu blok program ditandai dengan indentasi yang semakin menjorok ke dalam
- Dalam 1 blok program dimungkinkan ada blok program lain yang berada lebih di dalam (*inner block*)
- Jika instruksi berada dalam 1 blok, maka indentasi harus rapi. Jika tidak, akan *error*.

Blok Program (2)

```
a = int(input("Masukkan angka = "))
if (a > 50):
    print ("Hello World!")
    print ("bye")
else: # a <= 50
    print ("Hello Darling!")
    print ("bye bye")
```

OK!

```
a = int(input("Masukkan angka = "))
if (a > 50):
    print ("Hello World!")
    print ("bye")
else: # a <= 50
    print ("Hello Darling!")
    print ("bye bye")
```

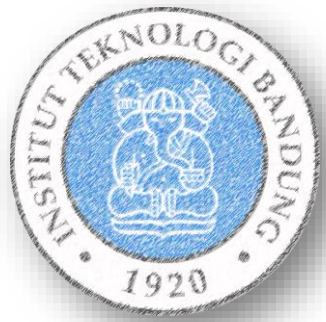
Error!

Contoh-2:

- Baris ke-7 s.d. 13 dalam 1 blok (*outer block*)
- Baris ke-9 s.d. 10 dalam 1 blok (*inner block*)
- Baris ke-12 s.d. 13 dalam 1 blok (*inner block*)

```
1  # Program Test
2
3  # KAMUS
4  # a : int
5
6  # ALGORITMA
7  a = int(input("Masukkan angka = "))
8  if (a > 50):
9      print ("Hello World!")
10     print ("bye")
11 else: # a <= 50
12     print ("Hello Darling!")
13     print ("bye bye")
14
```

Instruksi if-then-else... coming soon



Contoh-1. Roda Pak Pit

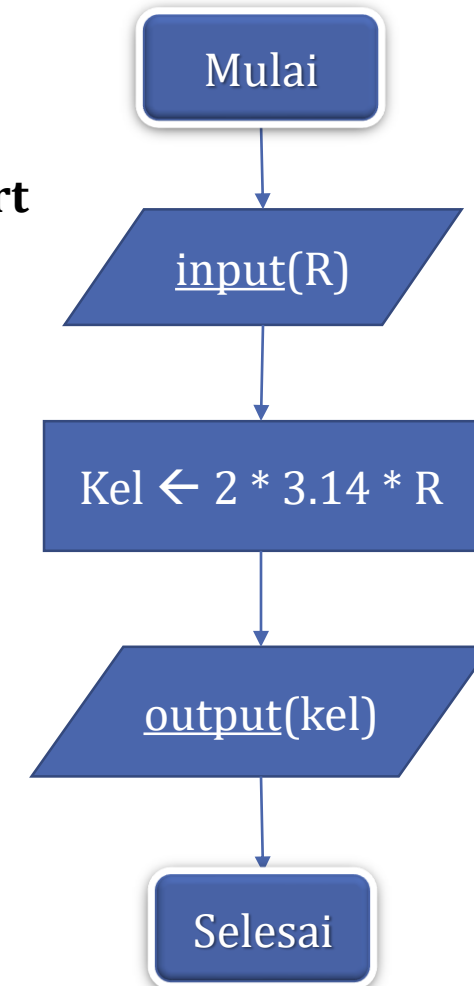
- Pak Pit, seorang pengusaha bengkel sepeda, memberikan tarif untuk setiap roda sepeda yang diperbaikinya berdasarkan keliling dari roda sepeda.
- Untuk itu, ia mengukur jari-jari sepeda, yaitu panjang dari pusat roda sampai tepi roda.
- Buatlah program yang menampilkan hasil perhitungan keliling lingkaran berdasarkan masukan nilai jari-jari.
- Rumus menghitung keliling lingkaran: $2 \pi r$
 - r adalah panjang jari-jari

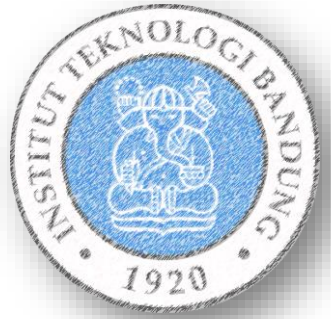
Contoh-1: Pseudocode + Flowchart

Pseudocode

```
input(R)  
Kel  $\leftarrow$  2 * 3.14 * R  
output(R)
```

Flowchart





Contoh-1: Python

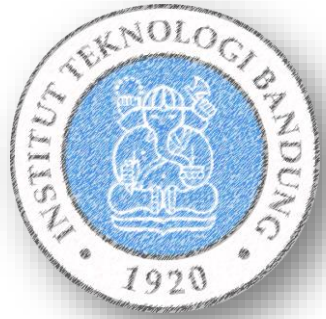
```
# Program KelilingLingkaran
# Menghitung keliling lingkaran berdasarkan masukan jari-jari

# KAMUS
# R : float
# Kel : float

# ALGORITMA
R = float(input())

Kel = 2 * 3.14 * R

print(Kel)
```

Contoh-2. Tinggi Rata-Rata

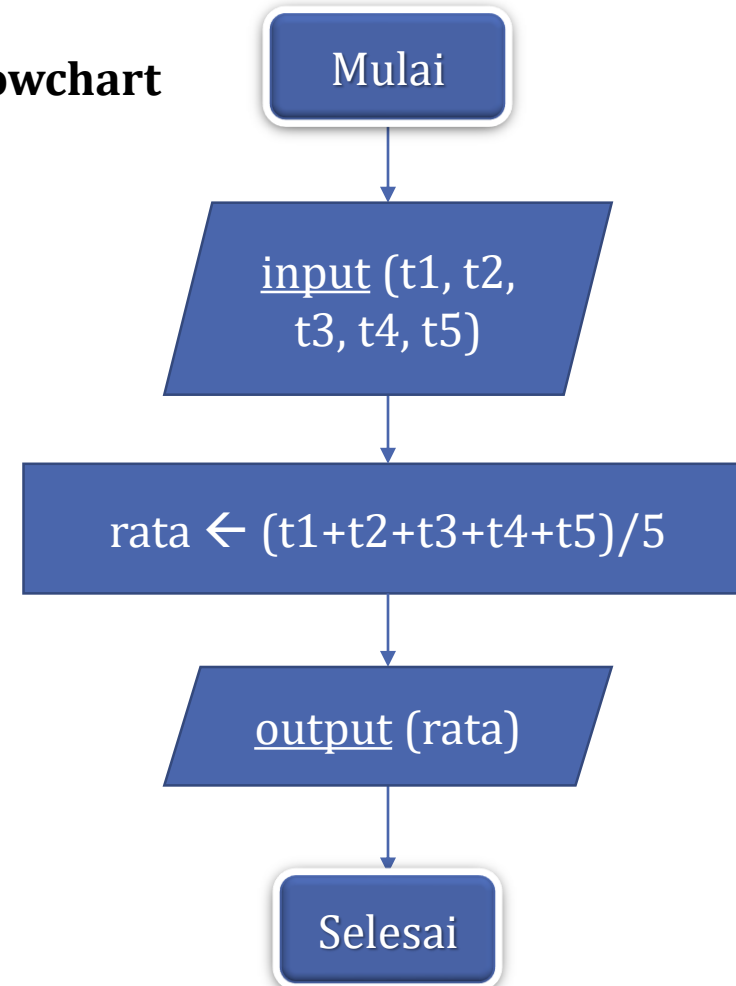
- Pak Guru menyeleksi 5 orang anak yang akan masuk ke tim basket sekolah. Ia ingin mengetahui tinggi badan rata-rata mereka.
- Buat program menghitung rata-rata dari tinggi badan 5 anak
 - Program akan menerima masukan data tinggi badan untuk 5 orang anak
 - Selanjutnya program menampilkan tinggi rata-rata dari ke lima anak tersebut

Contoh-2: Pseudocode + Flowchart

Pseudocode

```
input(t1, t2, t3, t4, t5)  
rata ← (t1+t2+t3+t4+t5)/5  
output(rata)
```

Flowchart





Contoh-2: Python

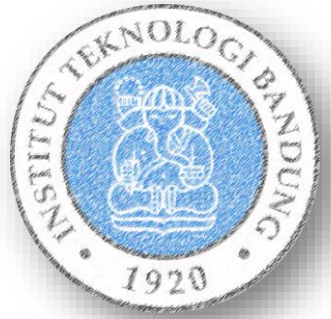
```
# Program TinggiRataRata
# Menerima tinggi 5 siswa dan menghitung rata-ratanya

# KAMUS
# t1, t2, t3, t4, t5 : float
# rata : float

# ALGORITMA
t1 = float(input())
t2 = float(input())
t3 = float(input())
t4 = float(input())
t5 = float(input())

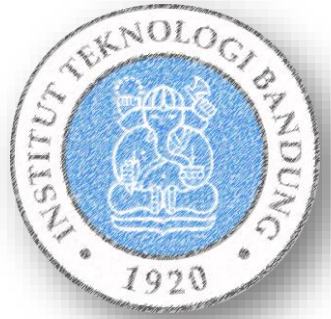
rata = (t1 + t2 + t3 + t4 + t5)/5

print (rata)
```



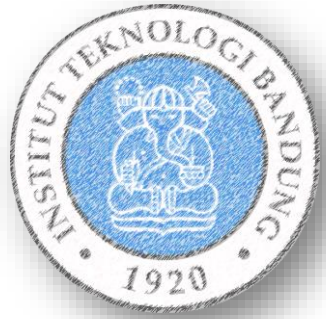
Latihan

- Untuk soal-soal berikut berlatihlah untuk membuat:
 - Flowchart atau Pseudocode (silakan pilih, atau ditentukan oleh dosen kelas)
 - Program Python yang bersesuaian



Latihan-1: Hitung Jarak

- Dalam Fisika, jarak (s) dapat dihitung berdasarkan kecepatan (v) dan waktu tempuh (t), yaitu: $s = v * t$
- Buatlah program untuk menghitung jarak (dalam m) berdasarkan masukan kecepatan (dalam m/s) dan waktu (dalam s)



Latihan-2. Umbul-Umbul Segitiga

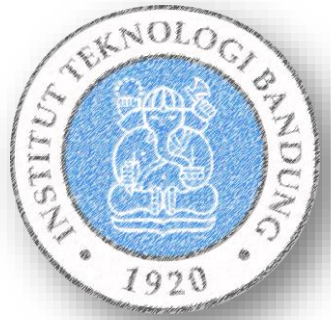
- Bu Tuti adalah seorang pengusaha umbul-umbul yang terkenal di kotanya. Dia membuat berbagai umbul-umbul dari berbagai bentuk, termasuk segitiga.
- Untuk setiap umbul-umbul segitiga, Bu Tuti menetapkan harga umbul-umbul berdasarkan luasnya. Untuk bisa menghitung luas umbul-umbul, Bu Tuti memerlukan tinggi dan alas umbul-umbul.
- Buatlah program yang menerima masukan tinggi dan alas dan menghasilkan luas umbul-umbul segitiga.
- Rumus luas segitiga: $\text{luas} = \frac{1}{2} * \text{alas} * \text{tinggi}$

Latihan-3. Toko Kelereng

- Sebuah toko menjual kelereng. Berikut adalah tabel harga kelereng berdasarkan warnanya:

Warna kelereng	Harga 1 butir (dalam ratusan rupiah)
Merah	10
Hijau	15
Kuning	20

- Seorang anak membeli kelereng sejumlah m kelereng merah, h kelereng hijau, dan k kelereng kuning. Asumsikan $m \geq 0$, $h \geq 0$, $k \geq 0$.
- Hitunglah berapa yang harus dibayarkan anak itu.



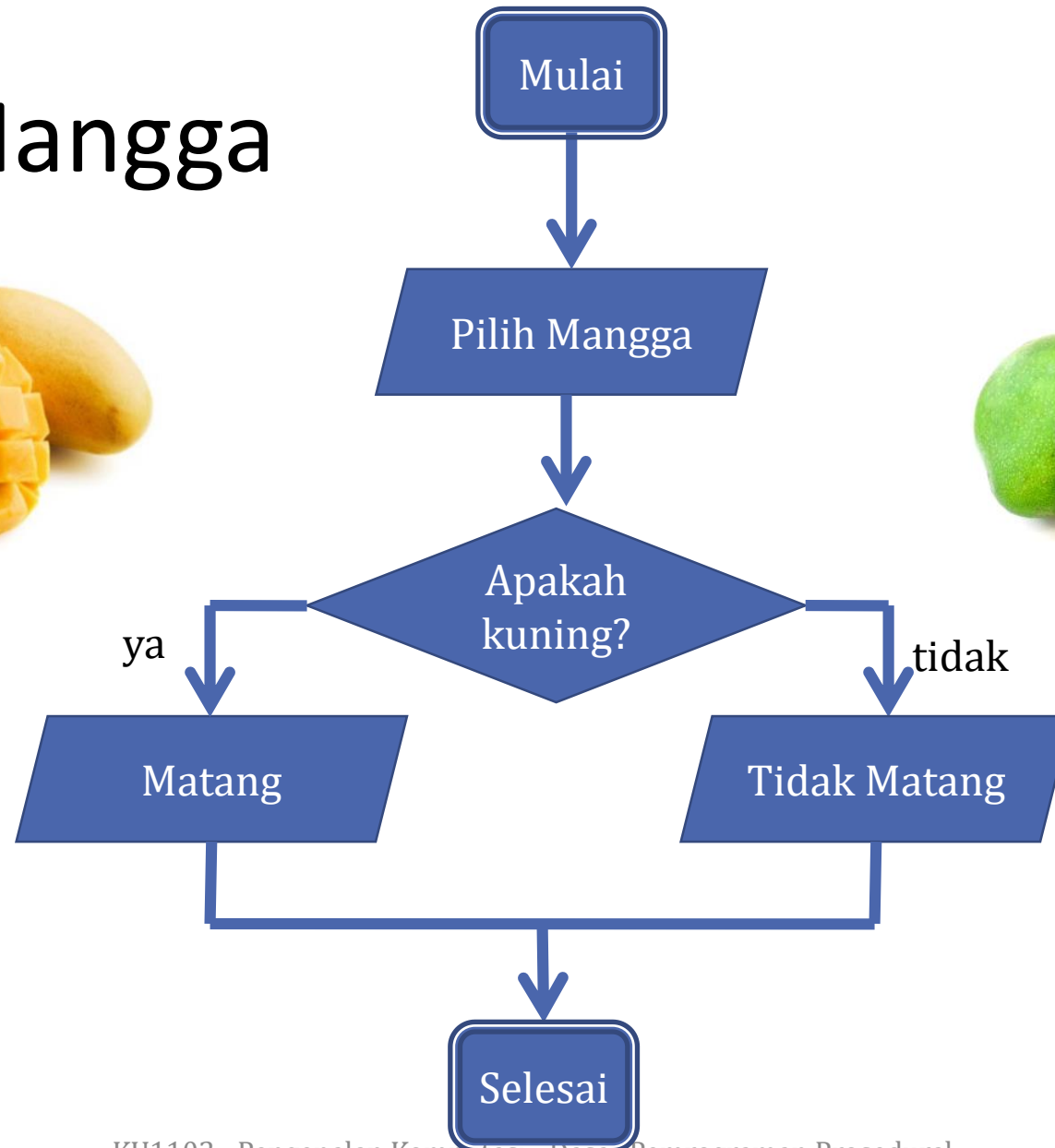
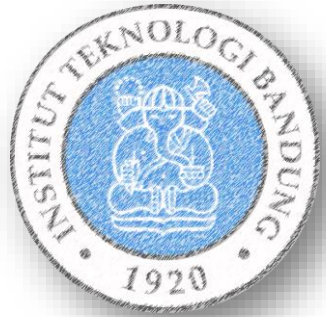
Analisis Kasus

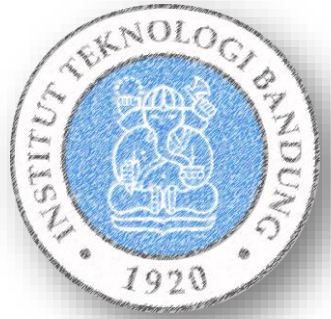
Contoh-1: Memilih Mangga

- Analisis kasus dapat digunakan dalam kehidupan sehari-hari, contoh: memilih mangga
- Mangga yang sudah matang dan siap dimakan adalah mangga yang berwarna kuning
- Jika tidak berwarna kuning maka tidak matang



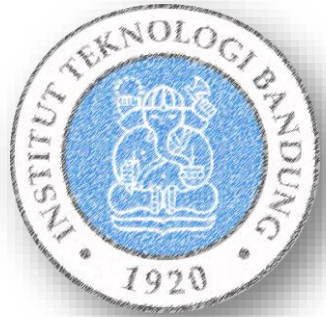
Flowchart Memilih Mangga





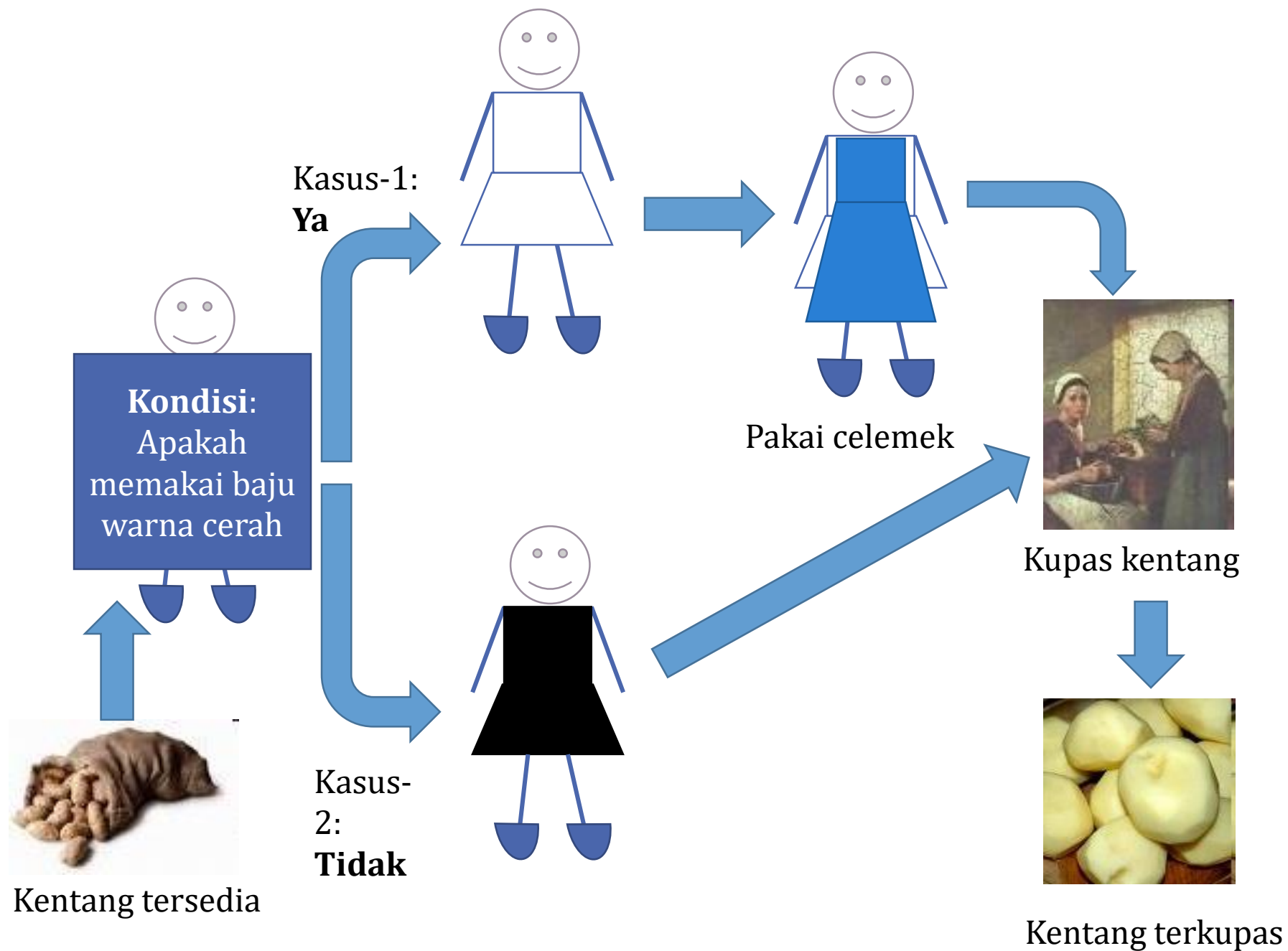
Memilih Mangga - Pseudocode

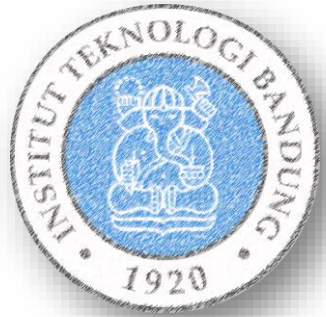
```
PilihMangga  
if (ApakahKuning? = true) then  
    Matang  
else { ApakahKuning? = false}  
    Tidak Matang
```



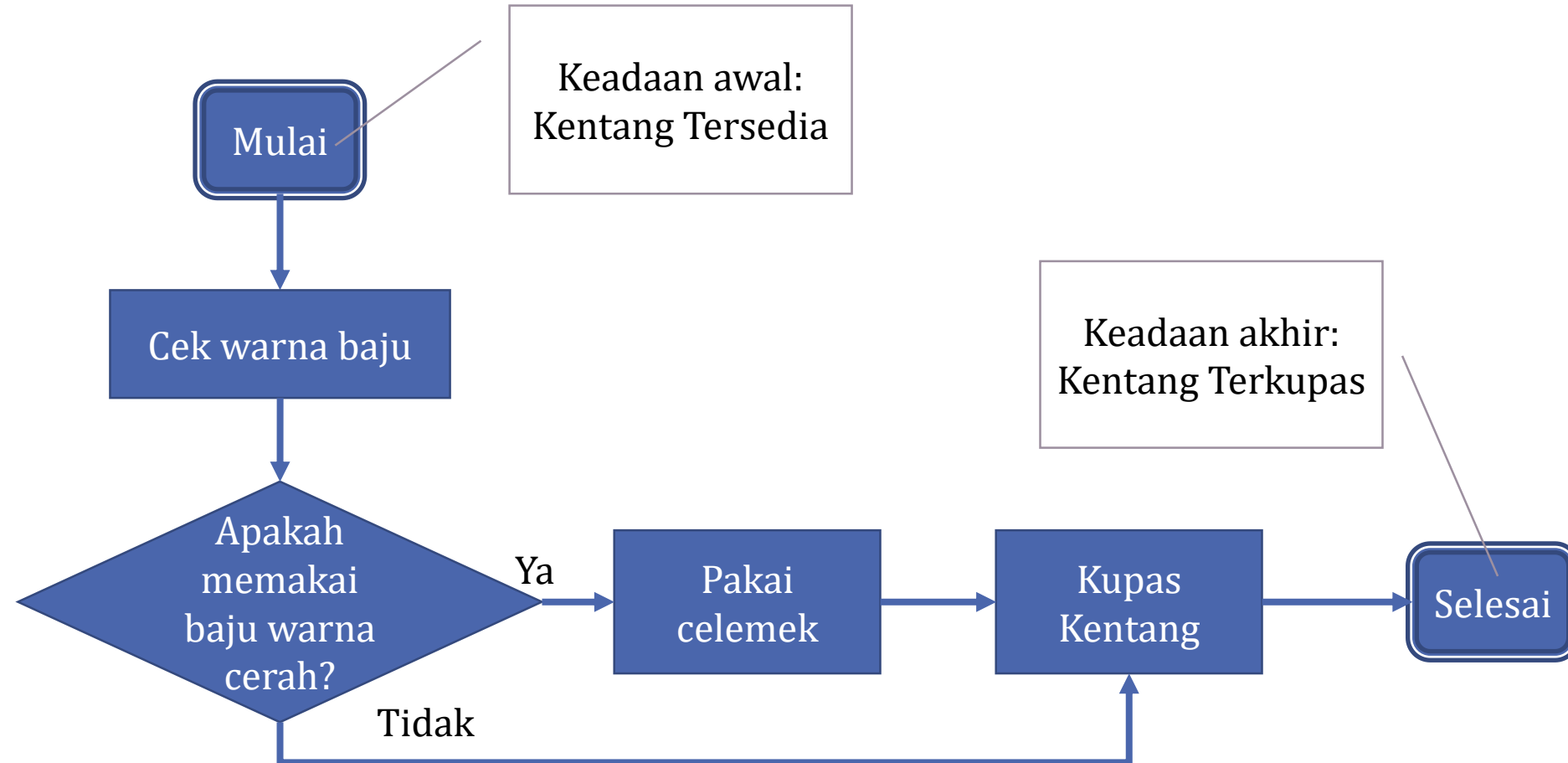
Contoh-2: Menyiapkan kentang untuk makan malam

- Berdasarkan pengamatan, ada hari-hari di mana ibu memakai celemek ketika mengupas kentang, tapi ada hari-hari lain yang tidak
 - Setelah diamati, ternyata jika ibu sedang memakai baju berwarna cerah, maka ibu memakai celemek → takut bajunya terlihat kotor 😊
 - Jika tidak (memakai baju berwarna gelap), maka ibu tidak memakai celemek

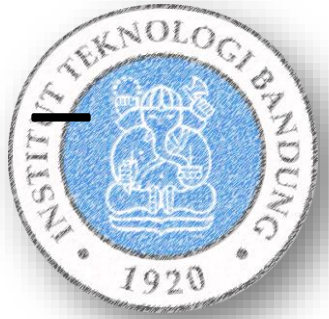




Flowchart: Menyiapkan kentang untuk makan malam



Menyiapkan kentang untuk makan malam – Pseudocode



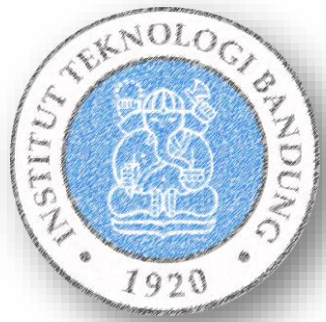
CekWarnaBaju

if (ApakahBajuWarnaCerah? = ya) then

 PakaiCelemek

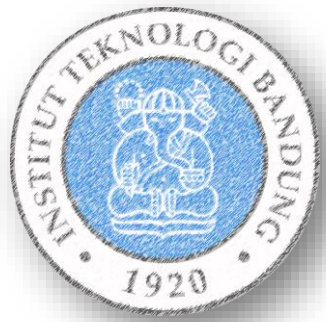
{ else : ApakahBajuWarnaCerah? = tidak, tidak melakukan apa-apa }

KupasKentang



Analisis Kasus (1)

- Memungkinkan kita membuat teks yang sama, namun menghasilkan eksekusi berbeda
- Sering disebut **percabangan / kondisional**
 - Dari satu langkah ada pilihan (bercabang) ke beberapa langkah
- Terdiri atas:
 - **Kondisi:** ekspresi yang menghasilkan true dan false
 - **Aksi:** statement yang dilaksanakan jika kondisi yang berpasangan dengan aksi dipenuhi



Analisis Kasus (2)

- Analisis kasus harus memenuhi 2 kriteria:
 - **COMPLETE**: semua kasus terdefinisi secara lengkap
 - **DISJOINT**: tidak ada kasus yang tumpang tindih/overlapped
- Contoh: Diberikan sebuah bilangan bulat, misalnya A, nyatakan apakah bilangan tersebut adalah bilangan positif, negatif, atau nol
- Ada 3 kasus yang *complete* dan *disjoint*:
 - $A > 0$
 - $A < 0$
 - $A = 0$
 - Tidak ada kasus lain yang bisa ddefiniskan dan ketiga kasus tersebut tidak tumpang tindih



Sintaks Umum

Python

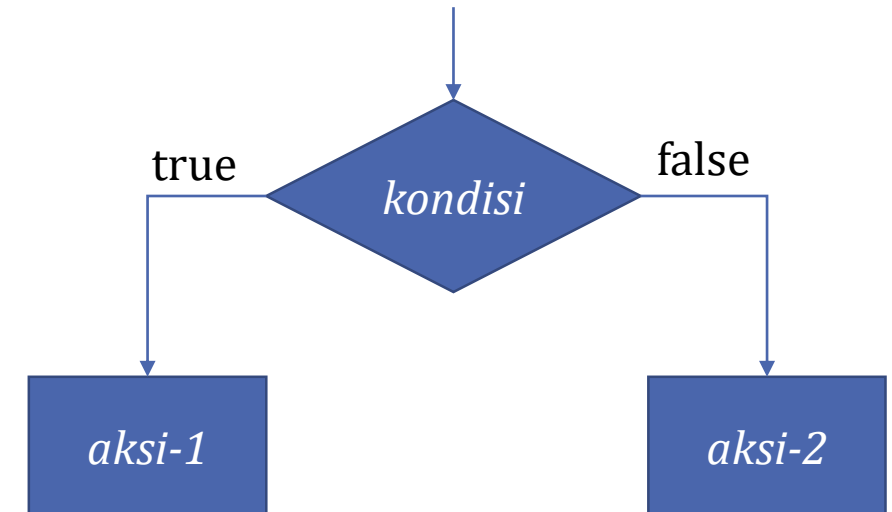
```
if ( kondisi ):  
    aksi-1  
else: # kondisi = false  
    aksi-2
```

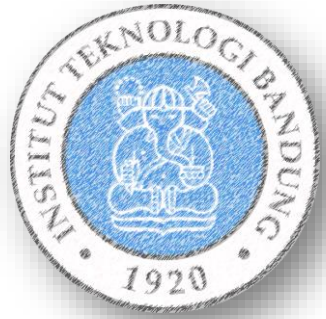
Jika aksi-1 atau aksi-2 terdiri dari lebih dari 1 instruksi, perhatikan bahwa indentasi harus rapi

Pseudocode

```
if ( kondisi ) then  
    aksi-1  
else { kondisi=false }  
    aksi-2
```

flowchart





Jenis Analisis Kasus (dalam Python)

Satu Kasus

```
if ( kondisi ):
    aksi-1

# jika kondisi=false
# tidak didefinisikan aksi
```

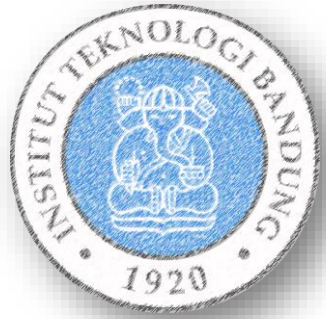
Dua Kasus [Komplementer]

```
if ( kondisi ):
    aksi-1
else: # kondisi=false
    aksi-2
```

Banyak Kasus

```
if (kondisi-1):
    aksi-1
elif (kondisi-2):
    aksi-2
elif (...):
    # kondisi-3 ... dst
    ...
else: # kondisi-n
    aksi-n
```

Pseudocode dan flowchart silakan disesuaikan atau lihat contoh-contoh berikut



Contoh-3: Apakah bilangan positif [Contoh Satu Kasus]

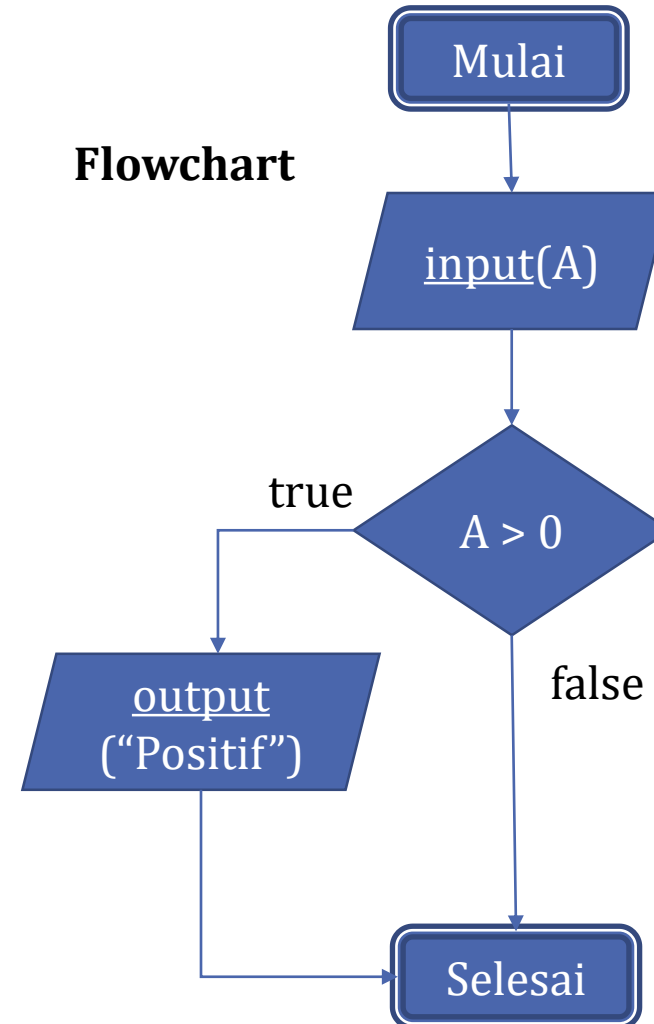
- Diberikan sebuah bilangan bulat, misalnya A, nyatakan apakah bilangan tersebut adalah bilangan positif atau bukan
- **Kondisi: Apakah $A > 0$?**
- **Kasus:**
 - Jika **ya**, maka: tuliskan “Positif”
 - Jika **tidak**, tidak dilakukan apa pun

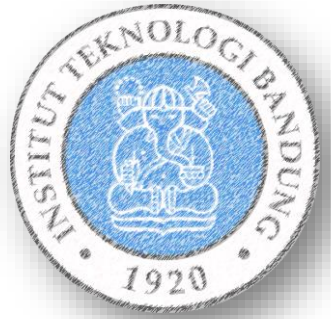
Contoh-3: Pseudocode + Flowchart

Pseudocode

```
input(A)  
if (A > 0) then  
    output("Positif")  
{ else: tidak dilakukan apa pun }
```

Flowchart





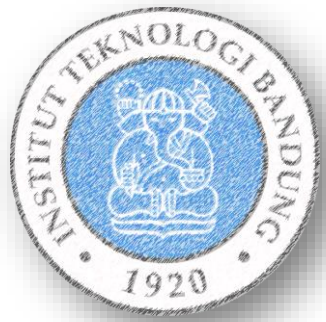
Contoh-3: Python

```
# Program CetakPositif
# Input A; jika A >= 0, cetak "positif"

# KAMUS }
# A : int

# ALGORITMA
A = int(input())

if (A >= 0):
    print("positif")
# else: tidak dilakukan apa pun
```



Contoh-4: Genap atau Ganjil?

[Contoh Dua Kasus Komplementer]

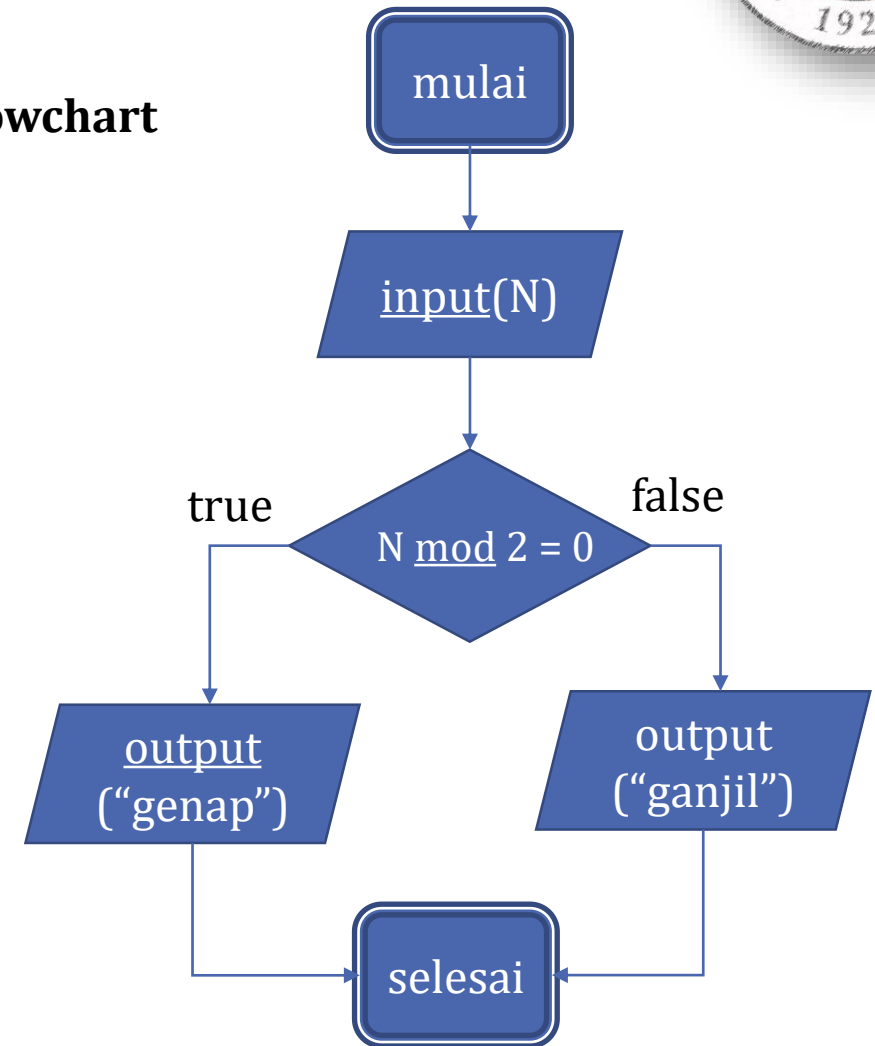
- Buatlah program yang menerima masukan sebuah integer positif (asumsikan masukan pasti benar), misalnya N, kemudian tentukan apakah bilangan tersebut genap atau ganjil.
- N adalah bilangan genap jika $N \bmod 2 = 0$; jika $N \bmod 2 = 1$, maka N adalah bilangan ganjil
 - Tidak ada kasus lain.
- Kasus:
 - Jika $N \bmod 2 = 0$ maka cetak “genap”
 - Jika tidak ($N \bmod 2 = 1$), maka cetak “ganjil”

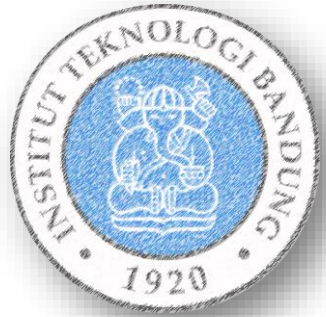
Contoh-4: Pseudocode + Flowchart

Pseudocode

```
input(N)  
if (N mod 2 = 0) then  
    output("genap")  
else { N mod 2 = 1 }  
    output("ganjil")
```

Flowchart





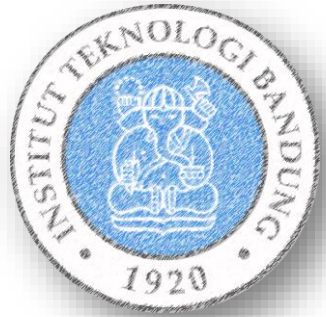
Contoh-4: Python

```
# Program GenapGanjil
# Input N>0. Jika N genap, cetak "genap"
# Jika tidak, cetak "N ganjil"

# KAMUS
# N : int

# ALGORITMA
N = int(input()) # Asumsi N > 0

if (N % 2 == 0):
    print("genap")
else: # N % 2 == 1
    print("ganjil")
```



Contoh-5: Positif, Negatif, atau Nol?

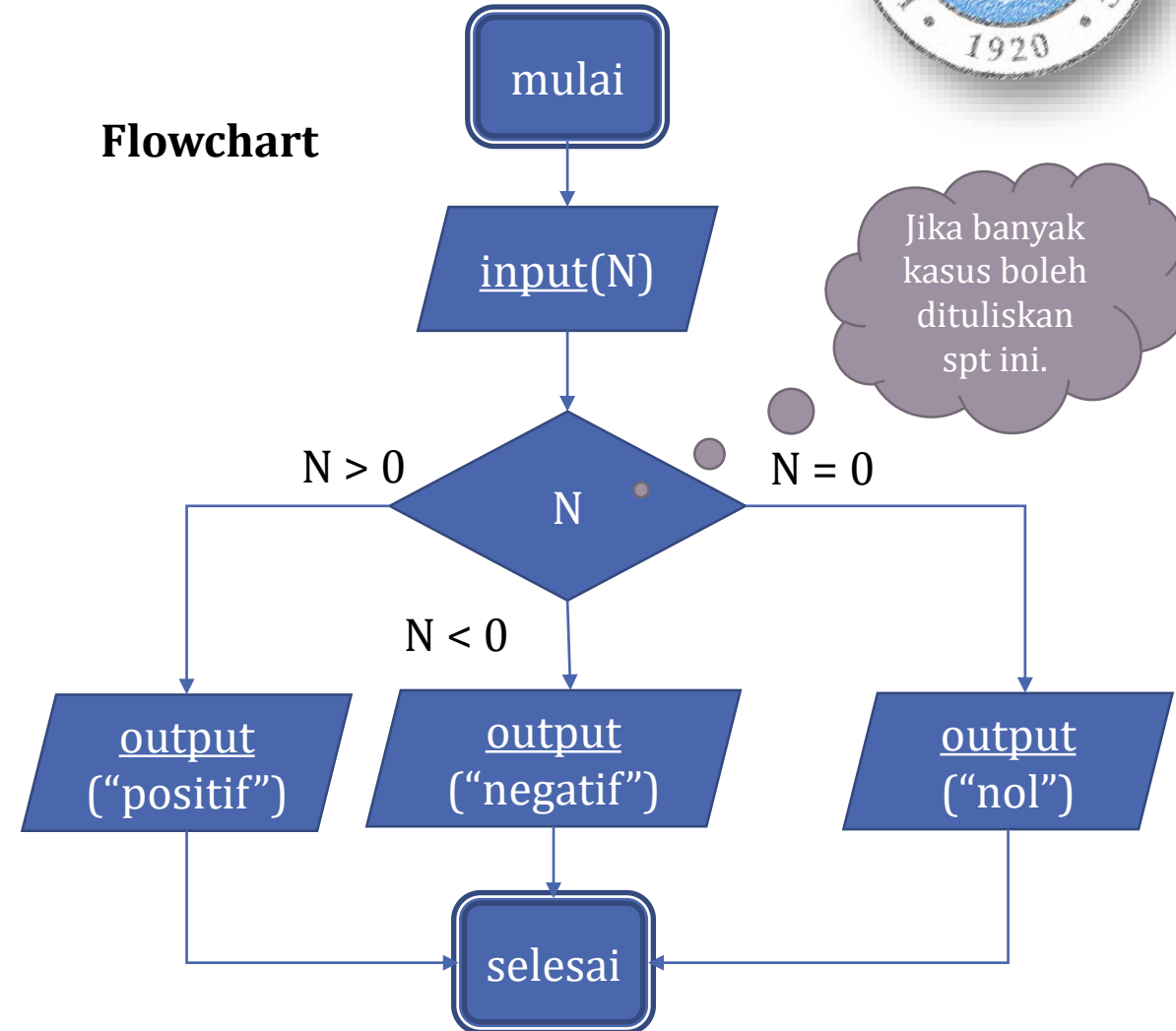
- Buatlah program yang menerima masukan sebuah integer, misalnya N, dan menentukan apakah N adalah bilangan bulat positif, negatif, atau nol
- Kasus:
 - Jika $N > 0$; cetak “positif”
 - Jika $N < 0$, cetak “negatif”
 - Jika $N = 0$; cetak “nol”

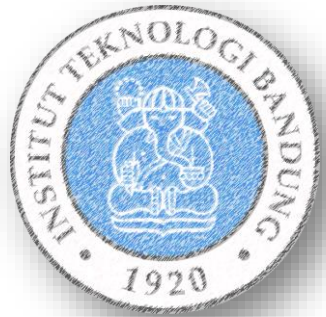
Contoh-5: Pseudocode + Flowchart

Pseudocode

```
input(N)  
if (N > 0) then  
    output("genap")  
else if (N < 0) then  
    output("negatif")  
else { N = 0 }  
    output("nol")
```

Flowchart





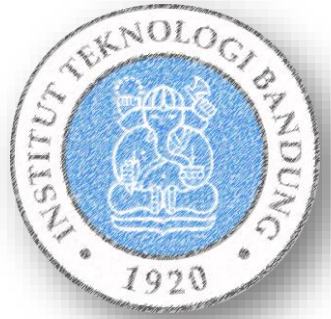
Contoh-5: Python

```
# Program Bilangan
# Input N. Tentukan apakah N positif, negatif, atau nol.

# KAMUS
# N : float

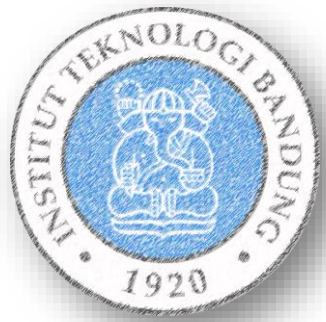
# ALGORITMA
N = int(input())

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



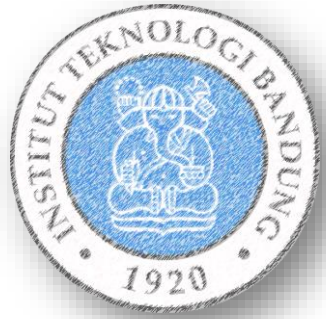
Latihan Soal

- Untuk soal-soal berikut berlatihlah untuk membuat:
 - Flowchart atau Pseudocode (silakan pilih, atau ditentukan oleh dosen kelas)
 - Program Python yang bersesuaian



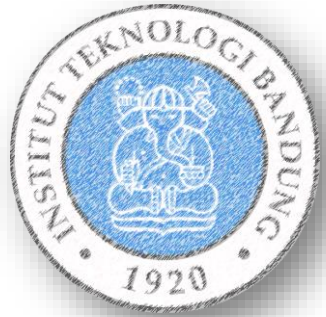
Latihan-1: Maksimum 2 bilangan

- Buatlah sebuah program yang membaca masukan 2 buah bilangan bulat, misalnya A dan B, dan tuliskan di antara kedua bilangan tersebut mana yang paling besar
- Kasus:
 - Jika $A > B$, maka bilangan terbesar = A
 - Jika $A < B$, maka bilangan terbesar = B
 - Jika $A = B$, maka bilangan terbesar adalah A atau B (berarti output akan sama seperti salah satu dari 2 kasus di atas)
- Apakah ini 3 kasus atau hanya 2 kasus?



Latihan-2. Wujud Air

- Buatlah sebuah program yang menerima suhu air (dalam derajat celcius) dan menuliskan wujud air ke layar yaitu **beku**, **cair**, atau **uap**.
 - Jika suhu air ≤ 0 derajat, maka tuliskan “beku”
 - Jika suhu air > 0 dan < 100 derajat, maka tuliskan “cair”
 - Jika suhu air ≥ 100 , maka tuliskan “uap”

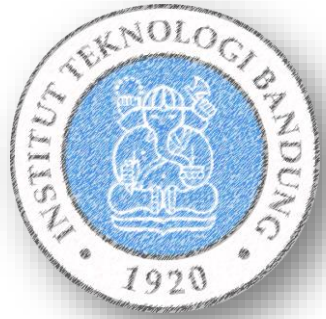


Latihan-3. Total Hambatan Seri

- Buatlah program yang menerima 3 buah hambatan (R1, R2, R3) dan menghasilkan hambatan total (RT) jika dirangkai seri.

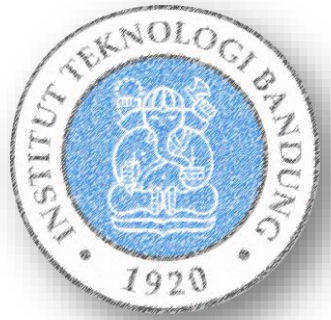
$$\mathbf{RT = R1 + R2 + R3}$$

- R1, R2, dan R3 tidak boleh bernilai negatif. Jika satu saja hambatan bernilai negatif, maka total hambatan tidak bisa dihitung dan tuliskan ke layar pesan kesalahan “Hambatan total tidak bisa dihitung”.

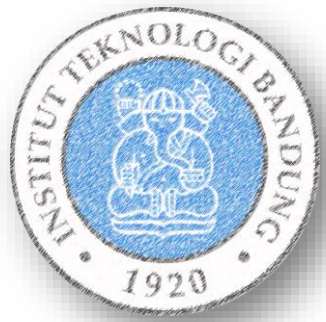


Latihan-4. Ranging 3 Bilangan

- Diberikan 3 buah integer yang dibaca dari keyboard, misalnya A, B, C. Asumsikan bahwa ketiga bilangan tersebut **berbeda**.
- Tuliskan ke layar ketiga bilangan tersebut dalam urutan dari yang terbesar sampai yang terkecil.
- Contoh: $A = 1$, $B = -1$, $C = 2$
Maka tertulis di layar: 2 1 -1

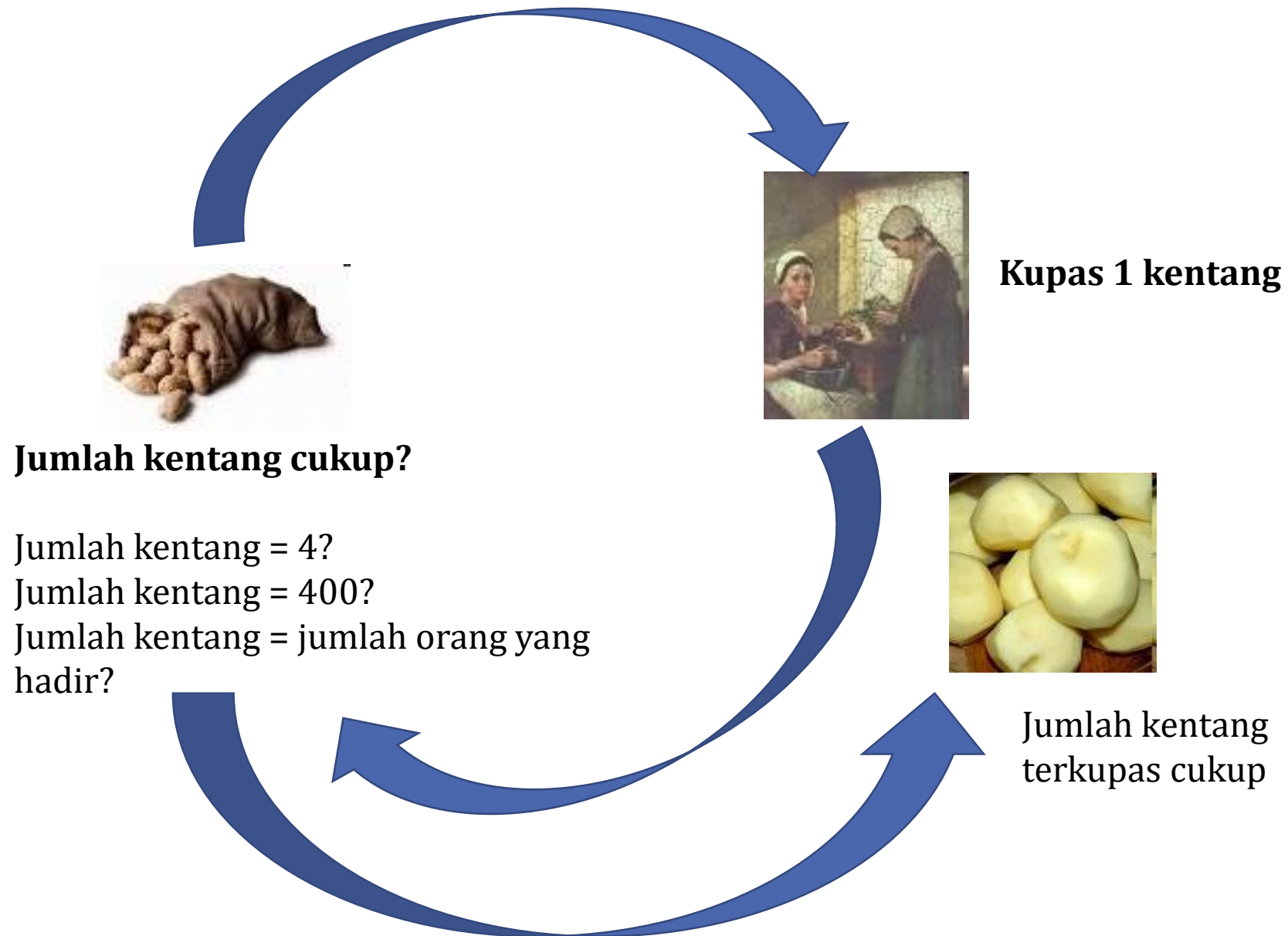


Pengulangan



Menyiapkan kentang untuk makan malam

- Asumsi: jumlah kentang tersedia tidak terbatas
- Pada suatu hari Ibu hanya mengupas kentang hanya 4 buah karena hanya anggota keluarga saja yang makan malam
- Pada hari yang lain, Ibu mengundang mahasiswa KU1102 sejumlah 400 orang untuk makan malam di rumahnya sehingga ibu mengupas 400 kentang untuk semua orang
- Hari yang lain, ibu tidak tahu berapa jumlah orang yang akan makan malam
 - Setiap selesai mengupas 1 kentang, dicek apakah jumlah cukup atau tidak

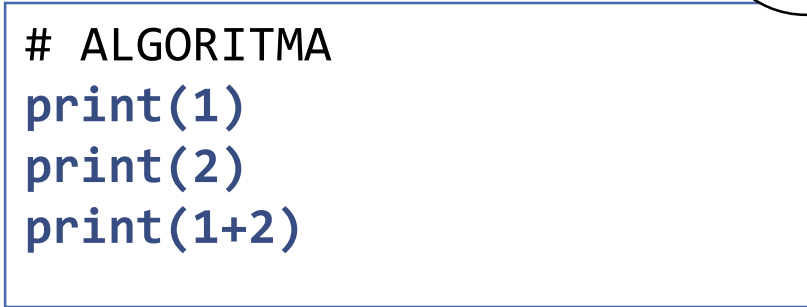


Menulis 1 dan 2

- Tuliskan program yang menuliskan angka 1 dan 2 dan selanjutnya 1+2 ke layar
- Contoh keluaran:



1
2
3




```
# ALGORITMA  
print(1)  
print(2)  
print(1+2)
```

Menulis 1 s.d. 3

- Tuliskan program yang menuliskan angka 1 s.d. 3 dan selanjutnya 1+2+3 ke layar
- Contoh keluaran:

1
2
3
6



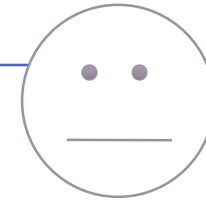
```
# ALGORITMA  
print(1)  
print(2)  
print(3)  
print(1+2+3)
```

Menulis 1 s.d. 10

- Tuliskan program yang menuliskan angka 1 s.d. 10 dan selanjutnya $1+2+3+\dots+10$ ke layar
- Contoh keluaran:

```
1
2
3
4
5
6
7
8
9
10
55
```


```
# ALGORITMA
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
... #lanjutkan sendiri!!
print(10)
print(1+2+3+4+5+6+7+8+9+10)
```



Menulis 1 s.d. 100

- Tuliskan program yang menuliskan angka 1 s.d. 100 dan selanjutnya $1+2+3+\dots+100$ ke layar
- Contoh keluaran:

```
1
2
3
4
5
6
7
8
9
10
... // lanjutkan sendiri!!
```



```
# ALGORITMA
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
... #lanjutkan sendiri!!
print(100)
print(1+2+3+4+5+6+7+8+9+10+ ... #lanjutkan sendiri!!)
```


Bagaimana kalau...

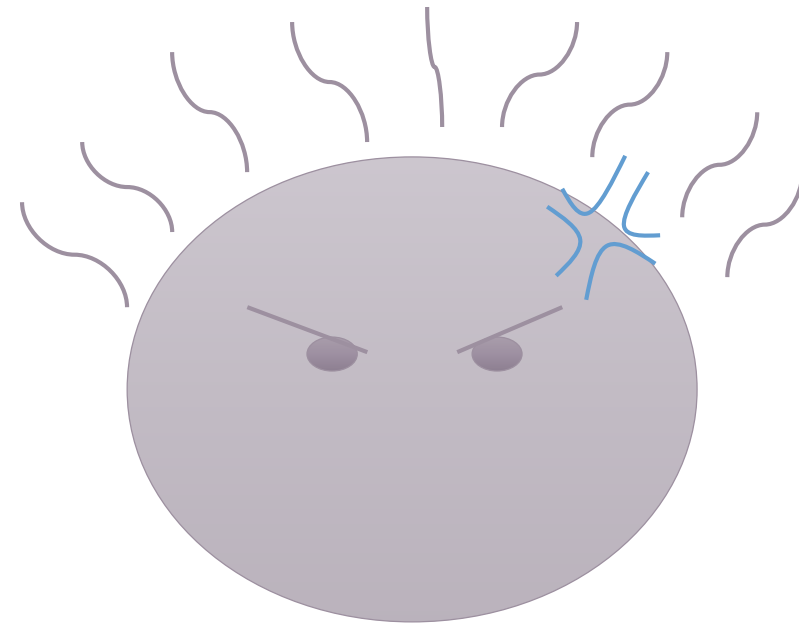
Anda diminta menulis dan menjumlahkan...

1 s.d. 1000 ???

1 s.d. 10000 ???

1 s.d. 1000000 ???

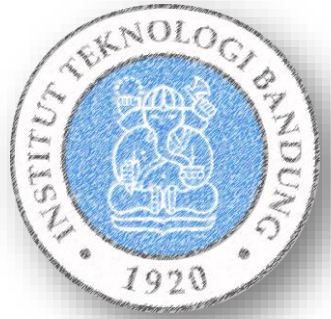
....



Pengulangan: Latar Belakang

- Melakukan suatu instruksi, bahkan aksi, secara berulang-ulang
 - Komputer: memiliki performansi yang sama
 - Manusia: punya kecenderungan untuk melakukan kesalahan (karena letih atau bosan)





Pengulangan (*Looping*)

- Elemen:
 - **Kondisi pengulangan:** ekspresi logik
 - **Badan pengulangan:** aksi yang diulang
- Jenis-jenis notasi pengulangan di Python:
 - Berdasarkan pencacah: **for**
 - Berdasarkan kondisi mengulang di awal: **while**

Contoh-1

- Tuliskan program yang menerima masukan sebuah integer misalnya N dan menuliskan angka 1, 2, 3, ... N dan menuliskan $1+2+3+\dots+N$ ke layar.
- Asumsikan $N > 0$.
- Contoh:

N = 1

Tampilan di layar:

1

1

N = 5

Tampilan di layar:

1

2

3

4

5

15

N = 10

Tampilan di layar:

1

2

3

4

5

6

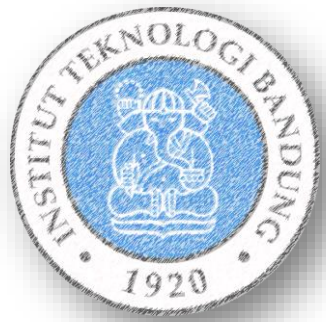
7

8

9

10

55



Berdasarkan Pencacah (for)

- Pengulangan dilakukan berdasarkan *range* harga suatu variabel *pencacah* (dalam contoh sebelumnya **i**)
 - *Range* harga pencacah yang diproses adalah dari **hmin** ke **hmaks**
- Pencacah harus suatu variabel dengan type yang terdefinisi suksesor dan predesesornya, misalnya integer
- **Aksi** akan dilakukan selama nilai pencacah masih berada dalam *range* yang ditentukan
- Harga pencacah di-*increment*, setiap kali **Aksi** selesai dilakukan
 - Karena itulah, nilai akhir *range* harus ditulis **hmaks+1** (agar **hmaks** tetap diproses)

Berdasarkan Pencacah (for)

Python

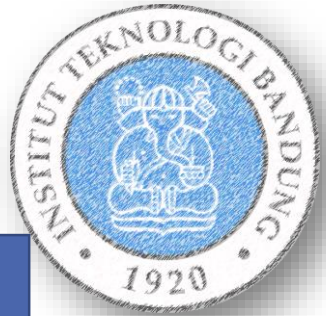
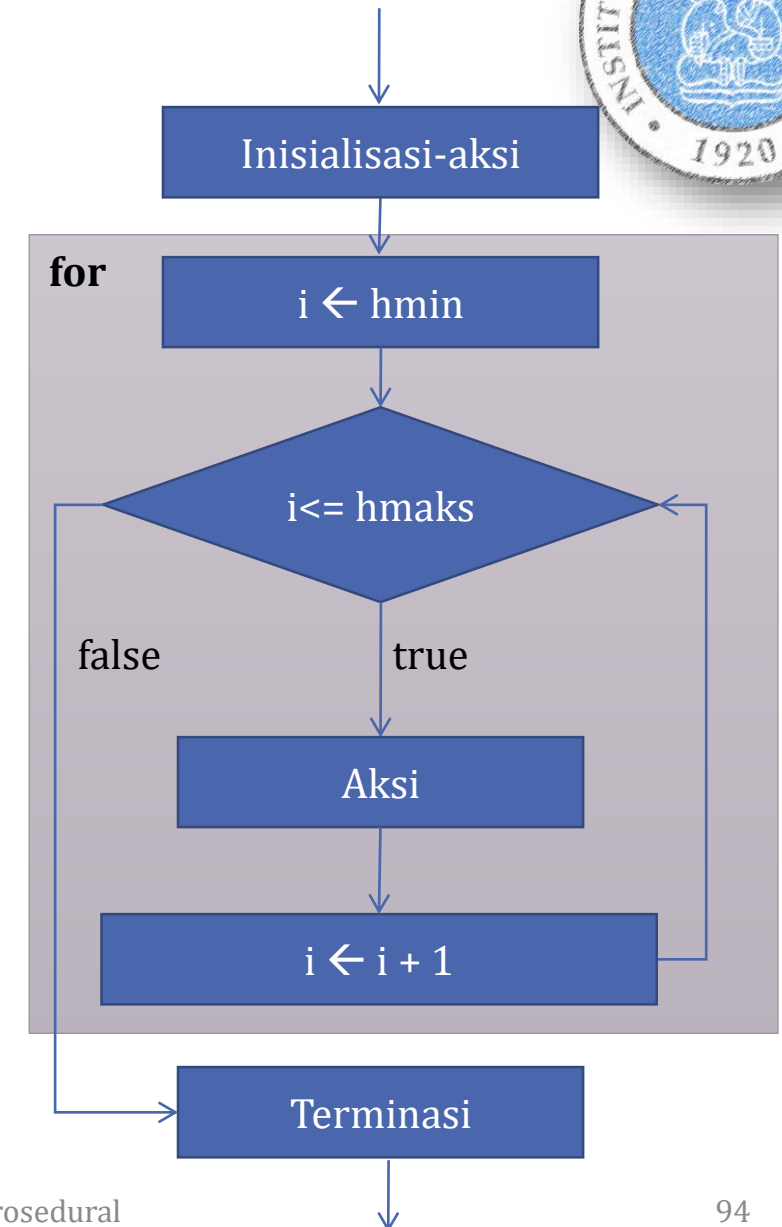
```
Inisialisasi-aksi  
for i in range(hmin, hmaks+1):  
    Aksi  
Terminasi
```

i adalah variabel pencacah (bisa diganti variabel lain)
hmin = nilai *i* di awal *loop*; *hmaks* = nilai *i* terakhir yang diproses;
nilai *i* ketika keluar *loop* adalah *hmaks*+1
Setiap berulang, *i* di-*increment* (ditambah 1)

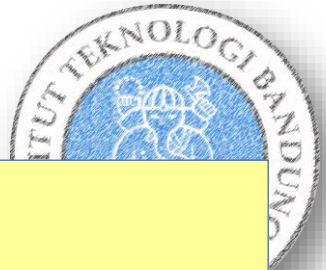
pseudocode

```
Inisialisasi-aksi  
i traversal [hmin..hmaks]  
    Aksi  
Terminasi
```

flowchart



Contoh-1: for



```
# Program JumlahAngka
# Menghitung 1+2+3+...+N. Asumsi N > 0

# KAMUS
# N : int
# i, sum : int

# ALGORITMA
N = int(input())      # Inisialisasi
sum = 0               # Inisialisasi

for i in range(1,N+1):
    print(i)          # Aksi
    sum = sum + i     # Aksi

print(sum)            # Terminasi
```

Mencacah Mundur

Python

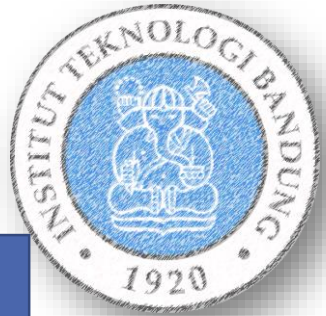
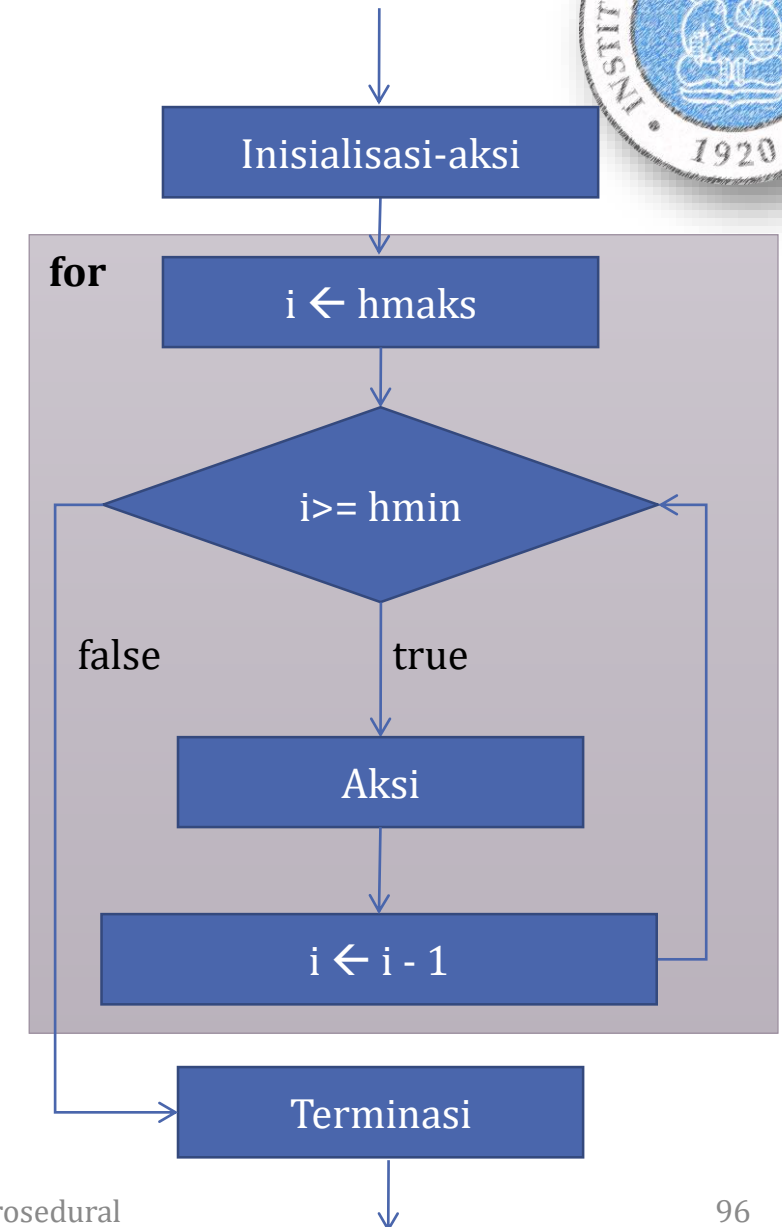
```
Inisialisasi-aksi  
for i in range(hmaks, hmin-1, -1):  
    Aksi  
Terminasi
```

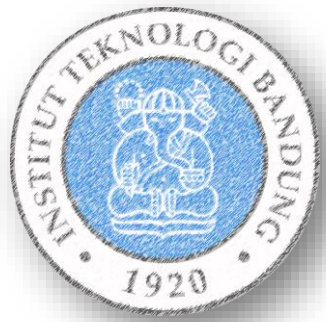
i adalah variabel pencacah (bisa diganti variabel lain)
hmaks = nilai *i* di awal *loop*; *hmin* = nilai *i* terakhir yang diproses;
nilai *i* ketika keluar *loop* adalah *hmin*-1
-1: Setiap berulang, *i* di-*decrement* (dikurangi 1)

pseudocode

```
Inisialisasi-aksi  
i traversal [hmaks..hmin]  
    Aksi  
Terminasi
```

flowchart





Pengulangan Berdasarkan Kondisi Mengulang di Awal (**while**)

- **Aksi** akan dilakukan selama **kondisi-mengulang** masih dipenuhi (berharga **true**)
- Pengulangan ini berpotensi untuk menimbulkan **Aksi** “kosong” (Aksi tidak pernah dilakukan sama sekali)
 - Karena pada *test* yang pertama, **kondisi-mengulang** langsung tidak dipenuhi (berharga **false**) sehingga langsung ke luar *loop*

Kondisi Mengulang di Awal (while)

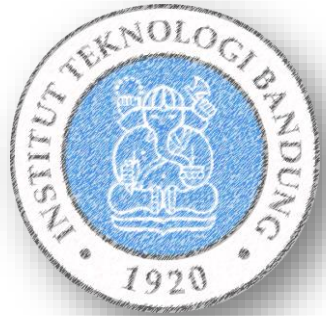
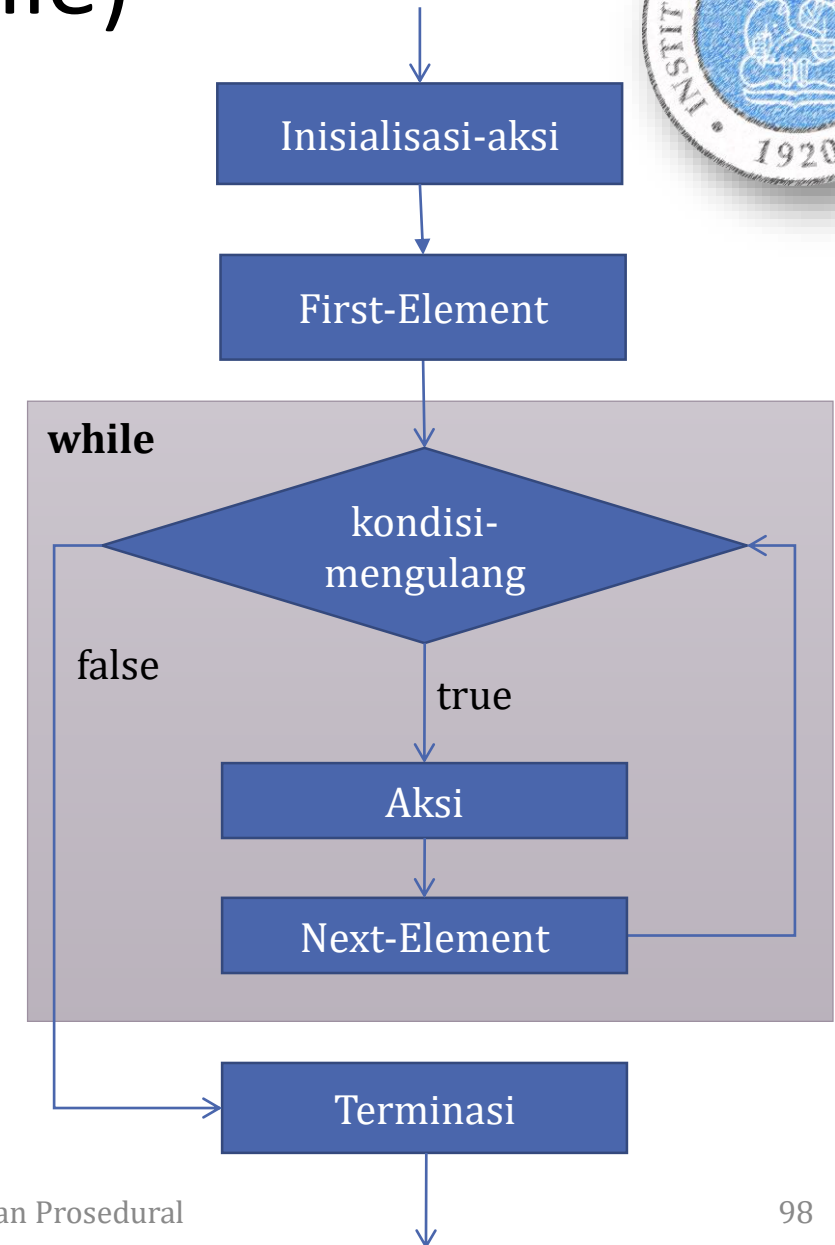
Python

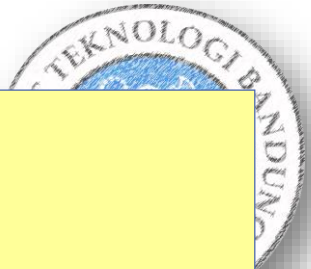
```
Inisialisasi-aksi  
First-Element  
while (kondisi-mengulang):  
    Aksi  
    Next-Element  
# kondisi-mengulang=false  
Terminasi
```

pseudocode

```
Inisialisasi-Aksi  
First-Element  
while (kondisi-mengulang) do  
    Aksi  
    Next-Element  
{ kondisi-mengulang = false }  
Terminasi
```

flowchart



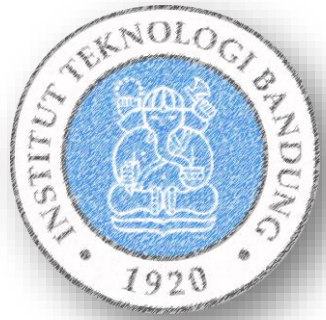


Contoh-1: while

```
# Program JumlahAngka
# Menghitung 1+2+3+...+N Asumsi N > 0

# KAMUS
# N : int
# i, sum : int

# ALGORITMA
N = int(input())           # Inisialisasi
sum = 0                    # Inisialisasi
i = 1                      # First-Element
while (i <= N):             # Kondisi-mengulang
    print(i)                # Aksi
    sum = sum + i           # Aksi
    i = i + 1               # Next-Element
# i > N
print(sum)                  # Terminasi
```



Contoh-2

- Buatlah program yang menerima masukan 10 buah bilangan integer (dari keyboard) dan menuliskan ke layar jumlah total ke-10 integer tersebut.
- Contoh:

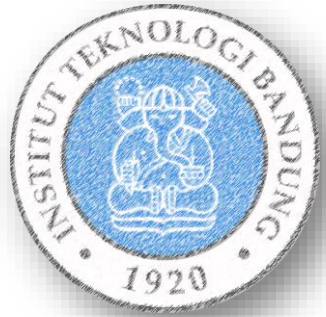
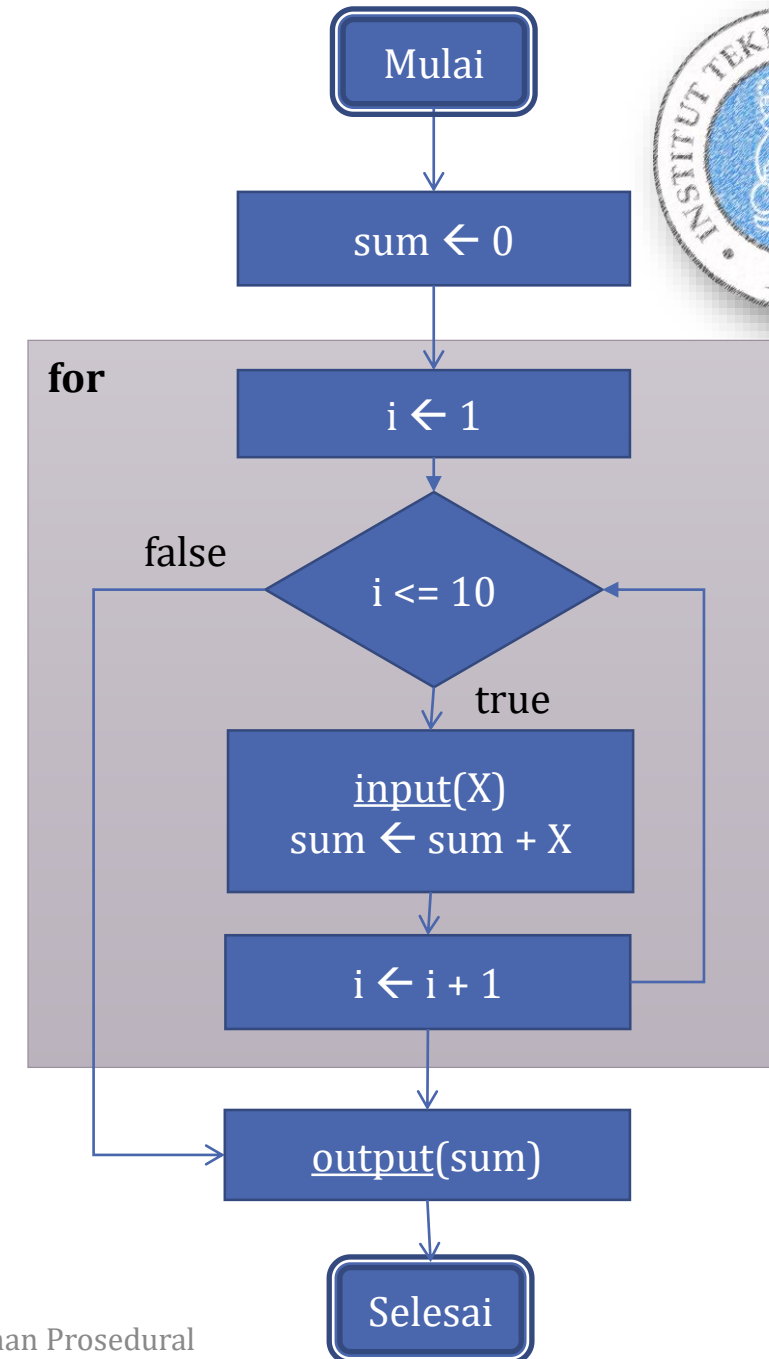
Masukan	Tampilan di Layar
2	18
1	
0	
-9	
7	
13	
2	
2	
1	
-1	

Contoh-2: for Pseudocode+Flowchart

pseudocode

```
sum ← 0 { inisialisasi }  
i traversal [1..10]  
    input(X)      { aksi }  
    sum ← sum + X { aksi }  
output(sum) { terminasi }
```

flowchart



Contoh-2: for Python

```
# Program Jumlah10Angka
# Menerima masukan 10 buah integer dan
# menjumlahkan totalnya

# KAMUS
# N, i, sum : int

# ALGORITMA
sum = 0                                # Inisialisasi

for i in range(1, 11):
    N = int(input())                  # Aksi
    sum = sum + N                     # Aksi

print(sum)                           # Terminasi
```

Contoh-2: Diskusi

- Paling tepat menggunakan **for**:
 - Karena berapa kali Aksi harus diulang diketahui secara pasti, yaitu 10x → berarti *range* harga pencacah untuk pengulangan diketahui secara pasti, yaitu dari 1..10 (nilai terakhir pencacah ketika keluar *loop* = 11)
- Kurang tepat menggunakan **while** karena tidak ada kemungkinan kasus “kosong”
 - **while** lebih tepat digunakan jika ada kemungkinan Aksi tidak pernah dilakukan sama sekali (kasus kosong) → dalam hal ini, Aksi pasti dilakukan, minimum 1 kali

Contoh-3

- Buatlah program yang membaca sejumlah bilangan integer dari keyboard sampai pengguna memasukkan angka -999 (angka -999 tidak termasuk bilangan yang diolah).
- Tuliskan berapa banyak bilangan yang dimasukkan, nilai total, dan rata-rata semua bilangan
- Jika dari masukan pertama sudah menuliskan -999, maka tuliskan pesan “Tidak ada data yang diolah”
- Petunjuk: Gunakan pengulangan **while**

No	Input	Output
1	<u>-1</u> <u>12</u> <u>-6</u> <u>10</u> <u>2</u> <u>-999</u>	Banyak bilangan = <u>5</u> Jumlah total = <u>17</u> Rata-rata = <u>3.40</u>
2	<u>-999</u>	<u>Tidak ada data yang diolah</u>

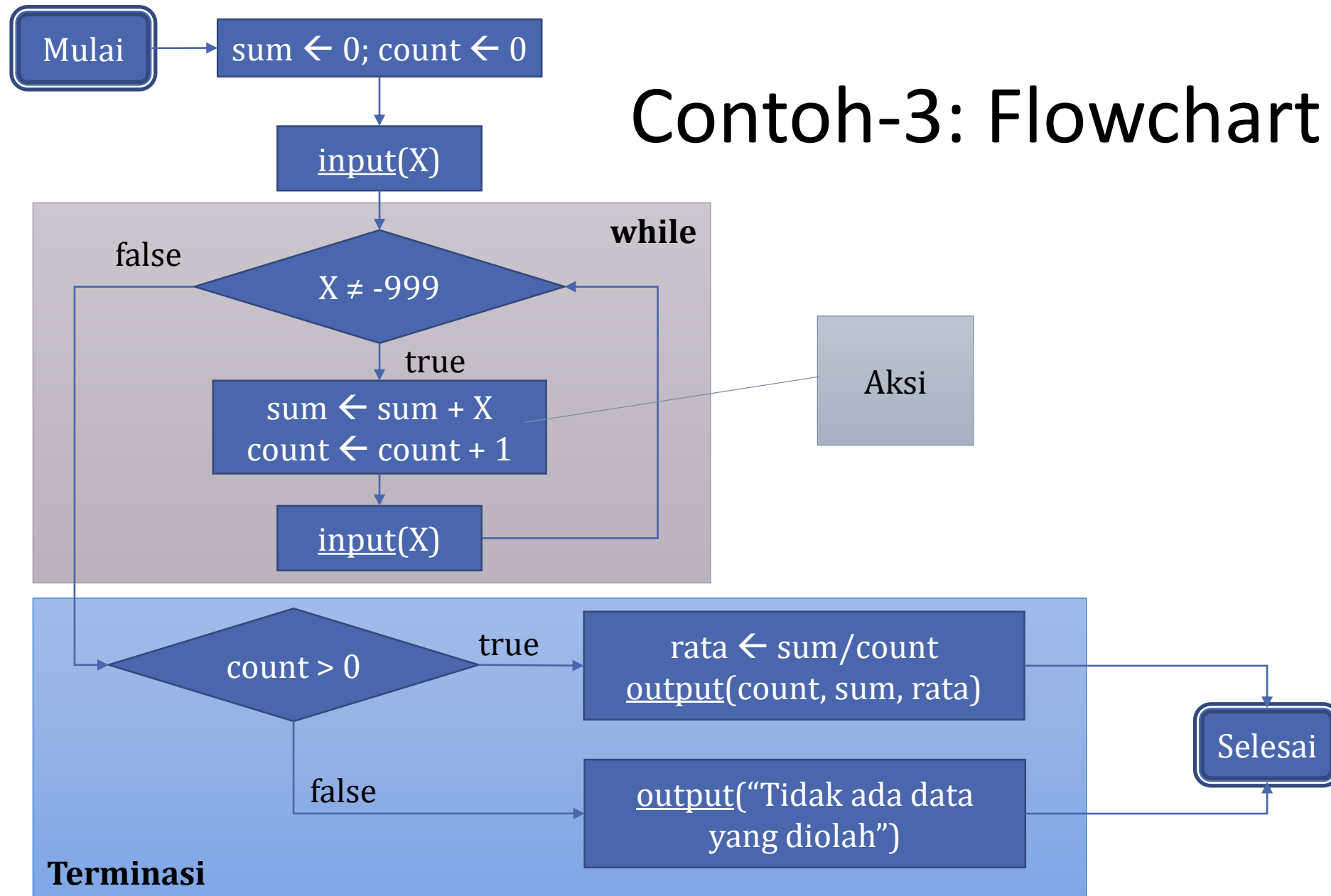
Contoh-3: Pseudocode

```
sum ← 0; count ← 0 { Inisialisasi }

while (X != -999) do
    sum ← sum + X
    count ← count + 1
    input(X)
{ X = -999 }

{ Terminasi }
if (count > 0) then
    rata ← sum/count
    output(count,sum,rata)
else { count = 0 }
    output("Tidak ada data yang diolah")
```

Contoh-3: Flowchart



Contoh-3: Python

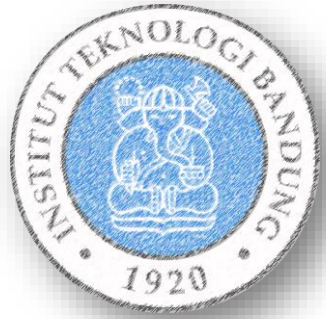
```
# Program RataBilangan
# Menerima masukan sejumlah bilangan integer sampai pengguna
# memasukkan -999 dan dan menampilkan banyak bilangan, total, dan
# rata-ratanya

# KAMUS
# X, count, sum : int
# rata : float

# ALGORITMA
sum = 0; count = 0      # Inisialisasi

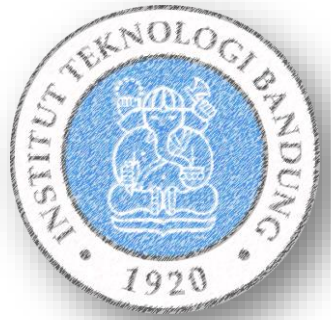
X = int(input())        # First-Elmt
while (X != -999):
    count = count + 1    # Aksi
    sum = sum + X
    X = int(input())     # Next-Elmt
# X = -999

# Terminasi
if (count > 0):
    print("Banyaknya bilangan = " + str(count))
    print("Jumlah total = " + str(sum))
    rata = sum/count
    print("Rata-rata = " + str(rata))
else:
    print ("Tidak ada data yang diolah")
```

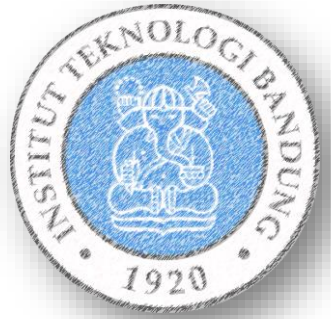


Contoh-3: Diskusi

- Pengulangan menggunakan **while** paling tepat karena:
 - Ada kemungkinan Aksi tidak pernah dilakukan sama sekali (kasus kosong), yaitu jika nilai X yang pertama kali dimasukkan user adalah -999 (lihat contoh ke-2)
- **For** tidak tepat digunakan karena tidak terdefinisi *range* nilainya



Latihan Soal



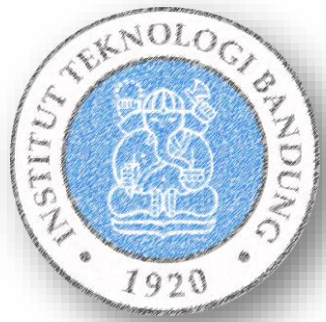
Latihan

- Untuk semua soal berikut:
 - Berlatihlah untuk membuat program Python dengan notasi pengulangan yang terbaik atau yang diminta
 - Buatlah juga *flowchart/pseudocode* (tergantung yang diminta oleh dosen kelas)

Latihan 1

- Buatlah algoritma/program yang membaca sebuah nilai integer positif, misalnya N, dan menjumlahkan (serta menampilkan) semua bilangan kelipatan 5 antara 1 s.d. N.
- Contoh:

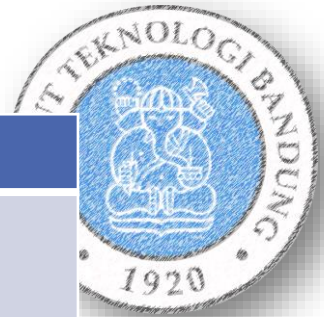
No	Input N	Output	Keterangan
1	5	5	Hanya ada 1 bilangan kelipatan 5 antara 1 s.d. 5, yaitu 5
2	26	75	Bilangan kelipatan 5 antara 1 s.d. 26 adalah 5, 10, 15, 20, 25 $5+10+15+20+25 = 75$
3	4	0	Tidak ada bilangan kelipatan 5 antara 1 s.d. 4



Latihan 2

- Buatlah algoritma/program yang membaca ada berapa banyak mahasiswa di kelas, misalnya N (Asumsi: $N > 0$, tidak perlu diperiksa)
- Selanjutnya, bacalah N buah character yang merepresentasikan nilai tugas KU1102. Nilai tugas yang mungkin adalah: 'A', 'B', 'C', 'D', 'E', 'F'. Asumsikan masukan nilai selalu benar.
- Jika mahasiswa mendapatkan nilai: 'A', 'B', 'C', atau 'D', maka mahasiswa dinyatakan lulus; sedangkan jika mendapat 'E' atau 'F' maka mahasiswa dinyatakan tidak lulus.
- Tuliskan ke layar berapa banyak mahasiswa yang lulus dan berapa yang tidak lulus.

Latihan 2: Contoh



No	Input N	Input nilai KU1102	Output
1	7	A B C A A E D	Lulus = 6 Tidak lulus = 1
2	5	A B B A A	Lulus = 5 Tidak lulus = 0
3	5	E E E E E	Lulus = 0 Tidak lulus = 5

Latihan 3

- Buatlah algoritma/program untuk membaca sekumpulan bilangan bulat (integer) positif. Pembacaan data diakhiri jika pengguna memasukkan nilai negatif.
- Selanjutnya, cetaklah berapa banyak bilangan genap dan ganjil.
- 0 adalah bilangan genap.

No	Input	Output	Keterangan
1	7 8 0 9 10 -1	Genap = 3 Ganjil = 2	
2	4 10 6 -111	Genap = 3 Ganjil = 0	
3	5 17 -234	Genap = 0 Ganjil = 2	
4	-99	Genap = 0 Ganjil = 0	Tidak ada bilangan positif yang dimasukkan
Semua input bilangan negatif mengakhiri pembacaan data			

Latihan 4: Lagu Anak Ayam

- Masih ingatkah dengan lagu Anak Ayam??

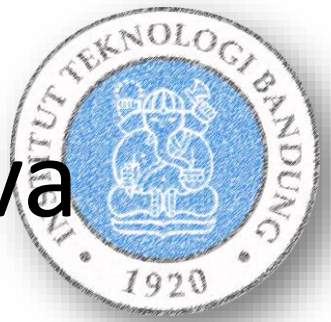
Anak ayam turunlah 5
Mati satu tinggalah 4
Mati satu tinggalah 3
Mati satu tinggalah 2
Mati satu tinggalah 1
Mati satu tinggal induknya

Anak ayam turunlah 1
Mati satu tinggal induknya

generalisasi

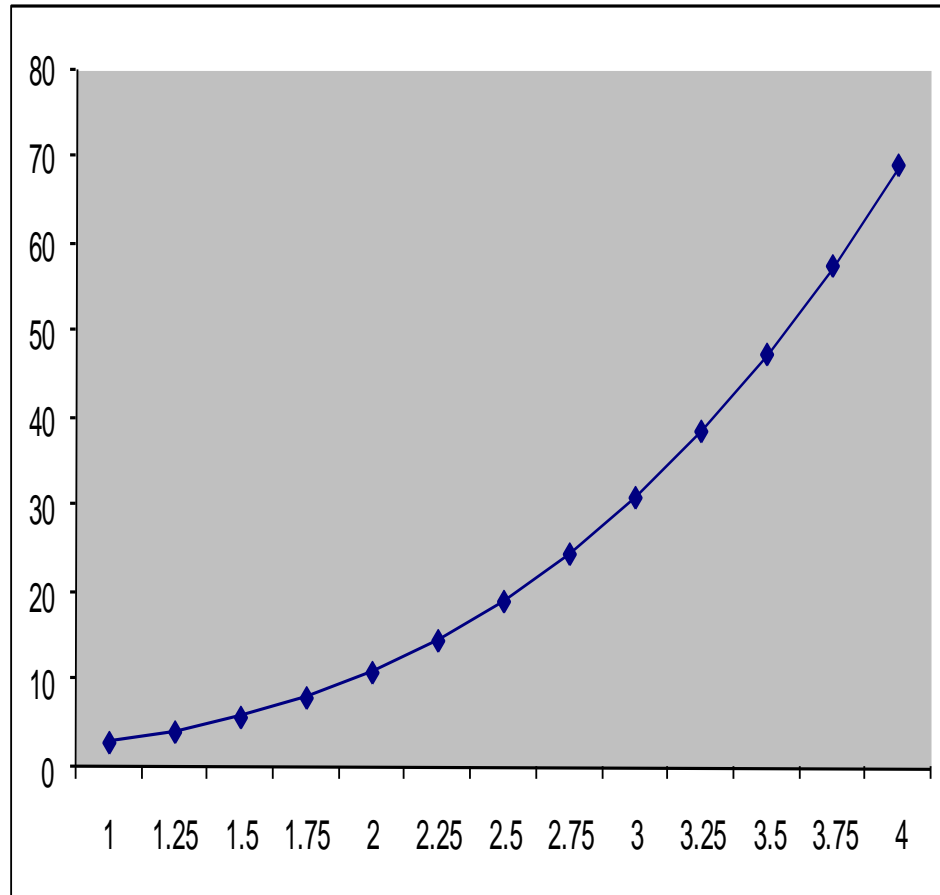
Anak ayam turunlah N
Mati satu tinggalah $N-1$
Mati satu tinggalah $N-2$
....
Mati satu tinggalah 1
Mati satu tinggal induknya

- Buatlah 3 versi program yang menerima masukan sebuah integer positif, misalnya N (asumsi $N > 0$), dan menuliskan lirik lagu Anak Ayam di atas dengan menggunakan perulangan **for**, **do-while**, dan **while**.
 - Berikan komentar, apakah masing-masing jenis pengulangan tepat untuk persoalan ini.



Latihan 5: Menghitung Luas di bawah kurva

- Untuk menghitung luas daerah dari suatu kurva yang dibentuk dengan rumus dapat dilakukan dengan menggunakan integral melalui menggunakan pendekatan numerik.
- Pendekatan numerik akan memotong-motong daerah dengan interval tertentu, kemudian dihitung luas masing-masing potongan daerah tersebut dengan menggunakan rumus trapesium secara berulang-ulang.
- Buatlah algoritma/program untuk persoalan berikut.



Asumsi:

$a < b$; $a \geq 0$; $b > 0$; $\delta > 0$

Untuk menghitung luas daerah yang dibangun dari rumus $f(x) = x^3 + x + 1$ dari $x = 1$ sampai $x = 4$ kita bisa memecah dengan suatu interval (misal 0.25).

Makin kecil interval, makin detil hasil yang diperoleh.

Luas daerah didapat dari menghitung luas semua trapesium hasil potongan berdasar interval.

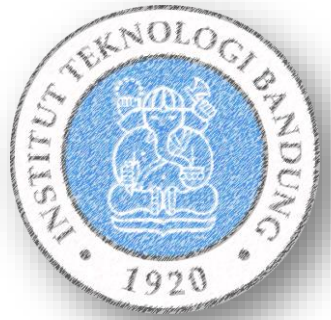
Tugas:

Buatlah algoritma/program untuk menghitung luas daerah yang dibangun dari rumus $f(x) = x^3 + x + 1$

dari $x=a$ sampai $x=b$ dengan interval δ , dengan a, b, δ merupakan masukan pengguna.

Latihan 6

- BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) Kota Bandung membutuhkan sebuah program untuk menghitung beberapa statistik dasar terkait suhu udara di Kota Bandung dalam 1 bulan, yaitu:
 - Suhu rata-rata
 - Suhu tertinggi
 - Suhu terendah
- Buatlah program yang menerima masukan jumlah hari dalam 1 bulan, misalnya N , (N bisa 28, 29, 30, atau 31 hari – asumsikan masukan benar), lalu menerima suhu udara dari hari ke-1 s.d. hari ke- N dalam bulan tersebut dan menuliskan statistik di atas.



Array dan Pemrosesannya

Kombinasi Pasangan Nama – 3 Nama

- Tuliskan program yang menerima 3 nama, lalu menampilkan semua kombinasi pasangan nama.
- Contoh keluaran:

Ali
Budi
Caca
Ali - Budi
Ali - Caca
Budi - Caca

```
# KAMUS
# nama1, nama2, nama3 : string

# ALGORITMA
nama1 = input()
nama2 = input()
nama3 = input()

print(nama1, " - ", nama2)
print(nama2, " - ", nama3)
print(nama3, " - ", nama1)
```



Kombinasi Pasangan Nama – 10 Nama

- Tuliskan program yang menerima 10 nama, lalu menampilkan semua kombinasi pasangan nama.
- Contoh keluaran:

Ali - Budi
Ali - Caca
...
Ina - Jaja

```
# KAMUS
# nama1,nama2,nama3,nama4,nama5 : string
# nama6,nama7,nama8,nama9,nama10 : string

# ALGORITMA
nama1 = input()
nama2 = input()
nama3 = input()
# ... Lanjutkan sendiri
nama10 = input()

print(nama1, " - ", nama2)
print(nama2, " - ", nama3)
print(nama3, " - ", nama4)
# ... Lanjutkan sendiri
print(nama10, " - ", nama1)
```



Bagaimana kalau...

Anda diminta menampilkan semua kombinasi pasangan nama yang mungkin dari ...

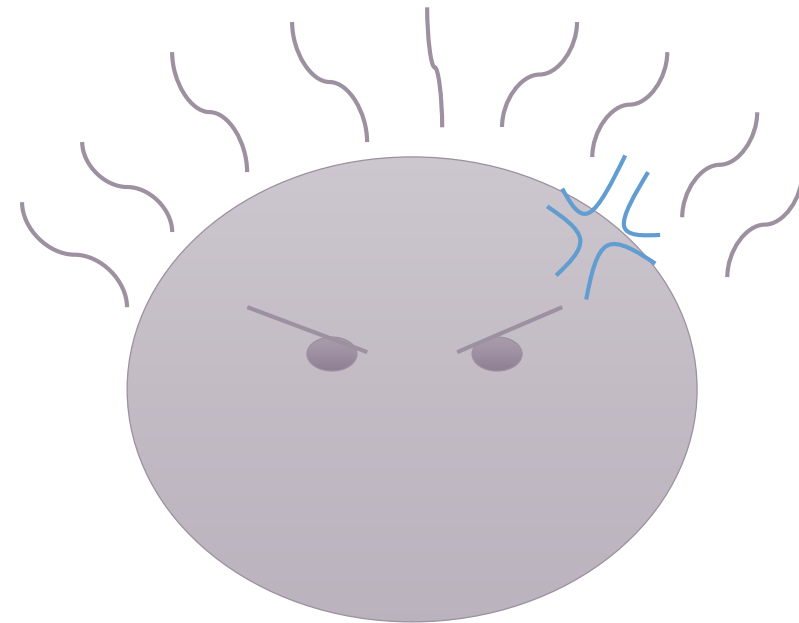
100 nama ???

1000 nama ???

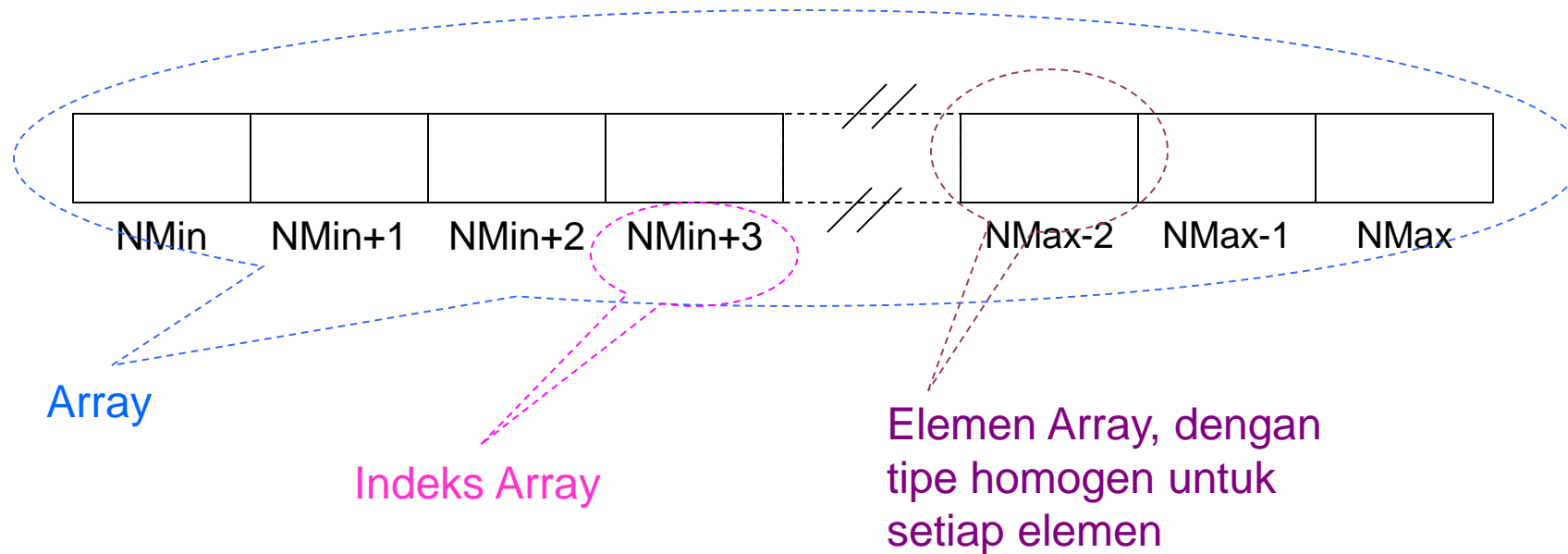
10000 nama ???

1000000 nama ???

....



Array / Tabel / Vektor / Larik



- **Array** mendefinisikan **sekumpulan** (satu atau lebih) elemen **bertipe sama**
- Setiap elemen tersusun secara teratur (kontigu) dan dapat diakses dengan menggunakan **indeks**
- Dalam Python, ada beberapa cara mendeklarasikan array → dalam kuliah ini, array didefinisikan menggunakan *collection type* **list**

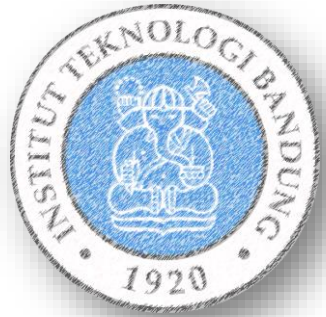
Deklarasi Array dalam Python (1)

Cara-1:

- Deklarasi variabel array sekaligus mendefinisikan isi array:

`<nama-var> = [<val0>, <val1>, <val2>, ..., <valn-1>]`

- Deklarasi array dengan nama `<nama-var>` dengan ukuran `n` dengan elemen `<val0>, <val1>, <val2>, ..., <valn-1>`
- Type elemen tergantung pada nilai elemen yang diberikan
- Elemen terurut berdasarkan indeks dari 0 s.d. `n-1`.



Deklarasi Array dalam Python (2)

- Contoh-1:

TabInt = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

9	8	7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7	8	9

Array bernama **TabInt** dengan setiap elemen bertipe **integer**, dengan ukuran **10** elemen, dengan alamat setiap elemen array (indeks) adalah dari **indeks ke-0 s.d. 9**

Deklarasi Array dalam Python (3)

- Jika belum diketahui nilai apa yang akan diberikan pada array, maka dapat diberikan suatu nilai default seragam terlebih dahulu
 - Contoh: Array berelemen integer: nilai elemen default = 0
- **Cara-2:** Mendeklarasikan array dan mengisi dengan nilai default:

```
<nama-var> = [<default-val> for i in range (<n>)]
```

 - Deklarasi array dengan nama `<nama-var>` dengan ukuran `<n>` dengan nilai setiap elemen `<default-val>`. `i` adalah variabel untuk loop pengisian nilai default ke tiap elemen array.
 - Type elemen tergantung pada type nilai `<default-val>`
 - Elemen terurut berdasarkan indeks dari 0 s.d. n-1.

Deklarasi Array dalam Python (4)

- Contoh-2: Array of integer

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

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

Array bernama **TabInt** dengan setiap elemen bertipe **integer** dan dengan nilai default elemen **0**, dengan ukuran **10** elemen dan setiap elemen array diakses dengan menggunakan **indeks ke-0 s.d. 9**

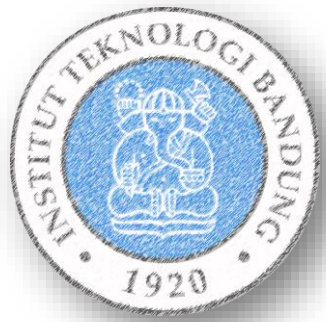
Deklarasi Array dalam Python (5)

- Contoh-3: Array of character

```
TabChar = ['*' for i in range(10)]
```

*	*	*	*	*	*	*	*	*	*
0	1	2	3	4	5	6	7	8	9

Array bernama **TabChar** dengan setiap elemen bertipe **char** dan dengan nilai default elemen *****, dengan ukuran **10** elemen dan setiap elemen array diakses dengan menggunakan **indeks ke-0 s.d. 9**



Mengakses Elemen Array dalam Python

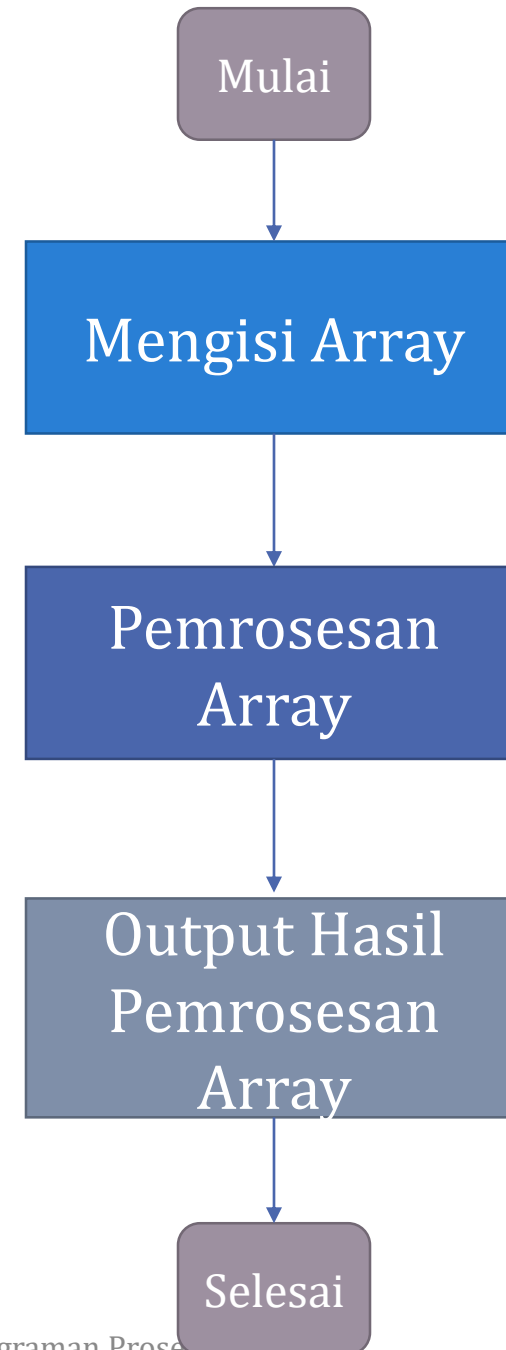
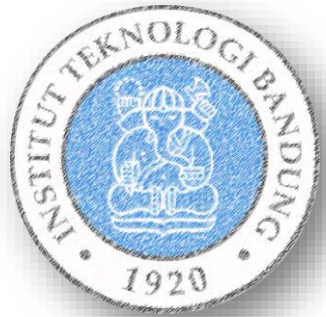
- Cara akses sebuah elemen: `<namatabel>[<indeks>]`
- Contoh: `TabInt = [1,2,4,-1,100,2,0,-1,3,9]`

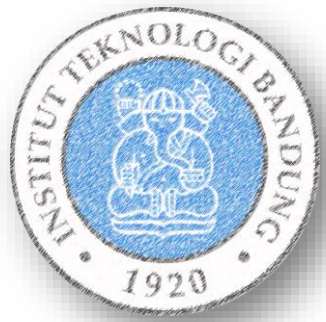
1	2	4	-1	100	2	0	-1	3	9
0	1	2	3	4	5	6	7	8	9

```
print(TabInt[5])           # akan tercetak: 2
x = TabInt[1] + TabInt[6]  # x = 2 + 0 = 2
TabInt[9] = 9              # Elemen array indeks 9 menjadi 9
TabInt[10] ???             # Berada di luar range, tidak terdefinisi!!
```

- **Perhatian: Tidak boleh** mengakses elemen dengan **indeks berada di luar definisi**.
 - Pada contoh di atas, misalnya: `TabInt[10]`, `TabInt[-1]`, dll

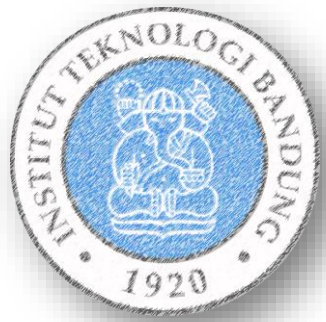
Pemrosesan Array





Pemrosesan Sekuensial pada Array (1)

- Pemrosesan **sekuensial** pada array adalah memproses setiap elemen array mulai dari elemen pada **indeks terkecil** s.d. **indeks terbesar** dengan menggunakan pengulangan (*loop*)
 - Setiap elemen array diakses secara langsung dengan indeks
 - *First element* adalah elemen array dengan indeks terkecil
 - *Next element* dicapai melalui suksesor indeks
 - Kondisi berhenti dicapai jika indeks yang diproses adalah indeks terbesar yang terdefinisi sebelumnya
- **Array tidak kosong**, artinya minimum memiliki 1 elemen

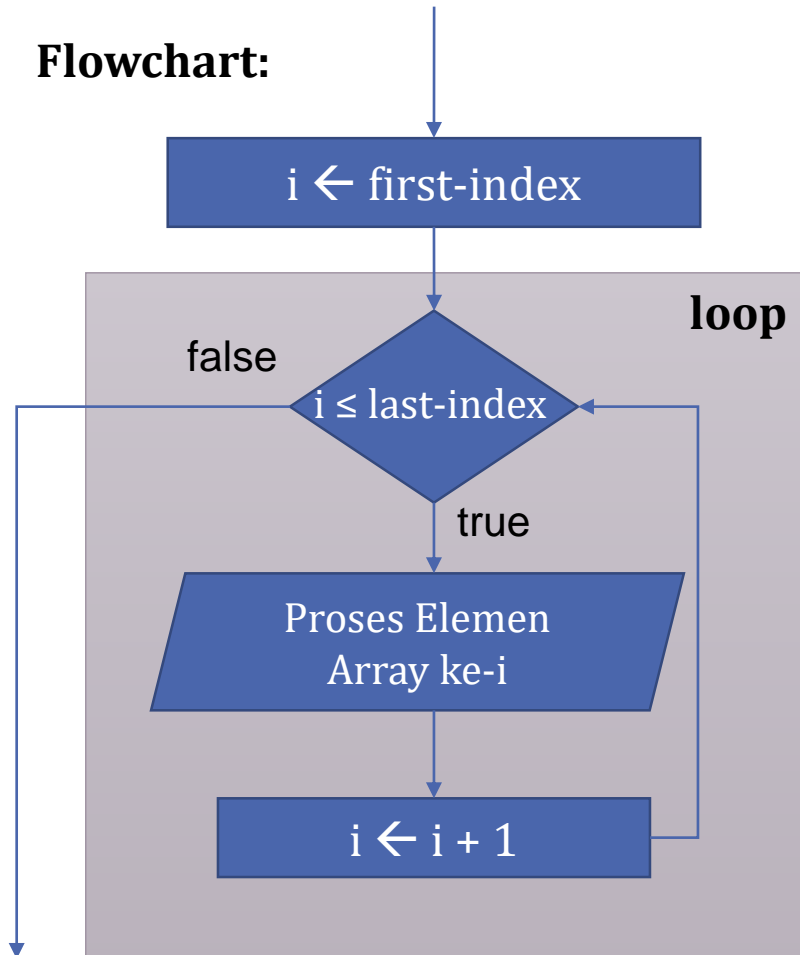


Pemrosesan Sekuensial pada Array (2)

- Contoh-contoh persoalan pemrosesan sekuensial pada array:
 - Mengisi array secara sekuensial
 - Mencetak elemen array
 - Menghitung nilai rata-rata elemen array
 - Mengalikan elemen array dengan suatu nilai
 - Mencari nilai terbesar/terkecil pada array
 - Mencari indeks di mana suatu nilai ditemukan pertama kali di array
 - ...

Flowchart + Pseudocode Umum Pemrosesan Sekuensial Array

Flowchart:



Pseudocode:

```
i traversal [first-index..last-index]
{ Proses elemen array ke-i }
...
```

Mengisi Array (1)

- Buatlah program yang **mendeklarasikan** sebuah **array of integer** (array dengan elemen bertipe integer) sebesar **10** buah dan mengisinya dengan nilai yang dibaca dari *keyboard*.
- **Hati-hati** untuk tidak mengakses elemen di luar batas indeks array!

```
# Program IsiArray
# Mengisi array dengan nilai dari
# pengguna

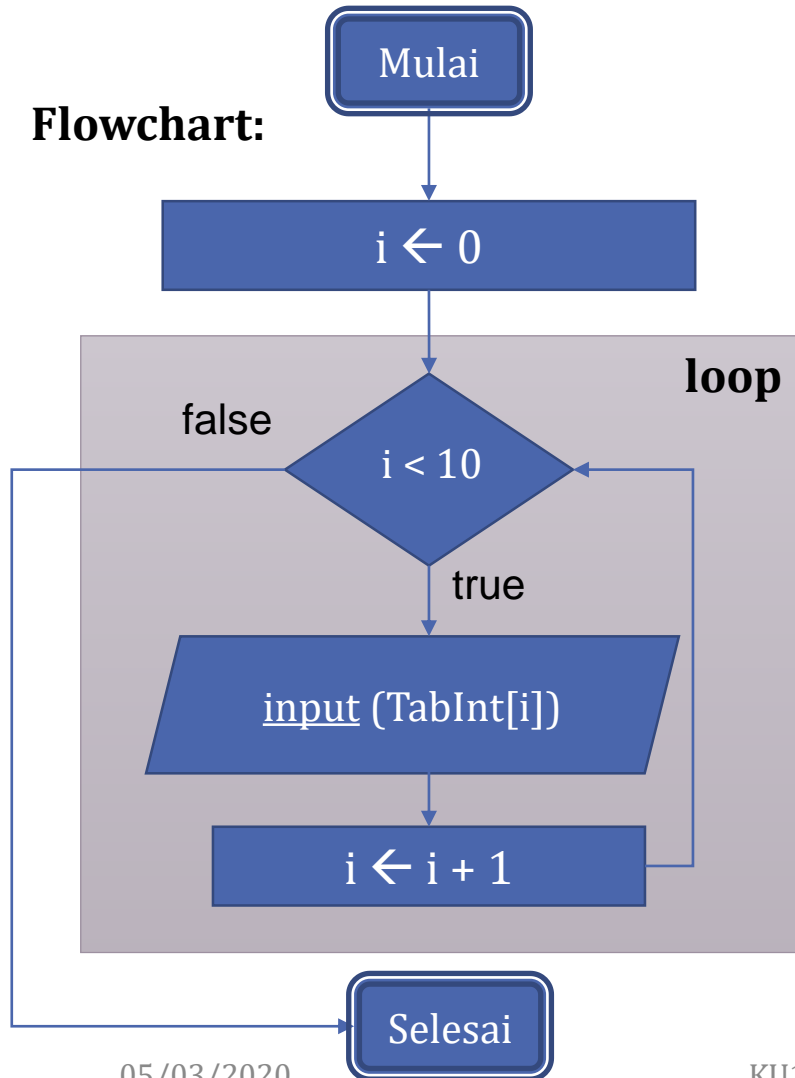
# KAMUS
# TabInt : array [0..9] of int
# i : int

# ALGORITMA
# Deklarasi array TabInt dan
# mengisinya dengan nilai default 0
TabInt = [0 for i in range(10)]

# Mengisi array dari pembacaan nilai
# dari keyboard
for i in range(0,10):
    TabInt[i] = int(input())
```

Mengisi Array (2): Flowchart + Pseudocode

Flowchart:



Pseudocode:

```
i traversal [0..9]  
  input(TabInt[i])
```

Menuliskan Isi Array (1)

- Buatlah program yang:
 - **mendeklarasikan** sebuah **array of integer** (array dengan elemen bertipe integer) sebesar **10** buah
 - **mengisinya** dengan nilai yang dibaca dari keyboard
 - **menuliskan** kembali apa yang disimpan dalam array ke layar
- **Hati-hati** untuk tidak mengakses elemen di luar batas indeks array!

```
# Program TulisArray
# Mengisi array dengan nilai dari
# pengguna dan menuliskan isinya ke
# layar

# KAMUS
# TabInt : array [0..9] of int
# i : int

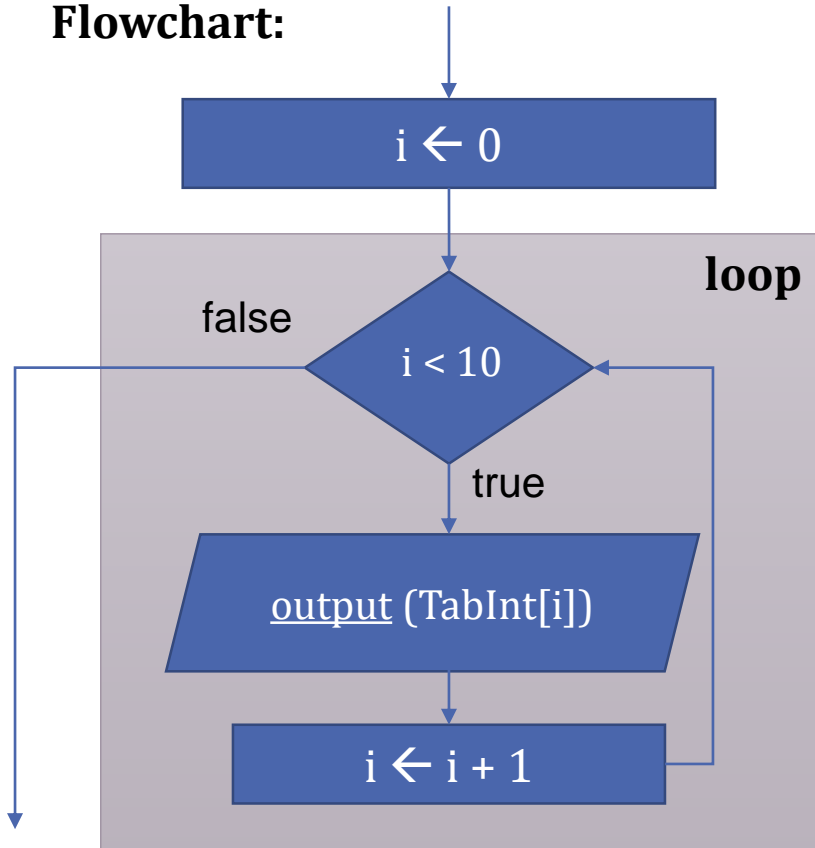
# ALGORITMA
# Deklarasi array TabInt dan
# mengisinya dengan nilai default 0
TabInt = [0 for i in range(10)]

# Mengisi array dari pembacaan nilai
# dari keyboard
for i in range(0,10):
    TabInt[i] = int(input())

# Mencetak isi array
for i in range(0,10):
    print(TabInt[i])
```


Menuliskan Isi Array (2)

Flowchart:



Pseudocode:

```
... { Bagian mengisi array }  
i traversal [0..9]  
  output(TabInt[i])
```

Bagian mengisi array buat sendiri sebagai latihan

Menghitung Rata-Rata (1)

- Buatlah program untuk menghitung rata-rata nilai elemen suatu array.
- Tahap:
 - Deklarasikan array, contoh array of integer ukuran 10
 - Isi elemen array
 - Jumlahkan semua elemen array
 - Bagi hasil penjumlahan elemen array dengan banyaknya elemen array dan tampilkan hasilnya

```
# Program AverageArray
# Menghitung nilai rata-rata elemen array

# KAMUS
# TabInt : array [0..9] of int
# i : int
# sum : int

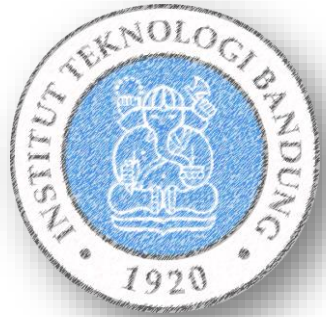
# ALGORITMA
# Deklarasi array TabInt dan mengisinya dengan
# nilai default 0
TabInt = [0 for i in range(10)]

# Mengisi array dari pembacaan nilai dari keyboard
for i in range(0,10):
    TabInt[i] = int(input())

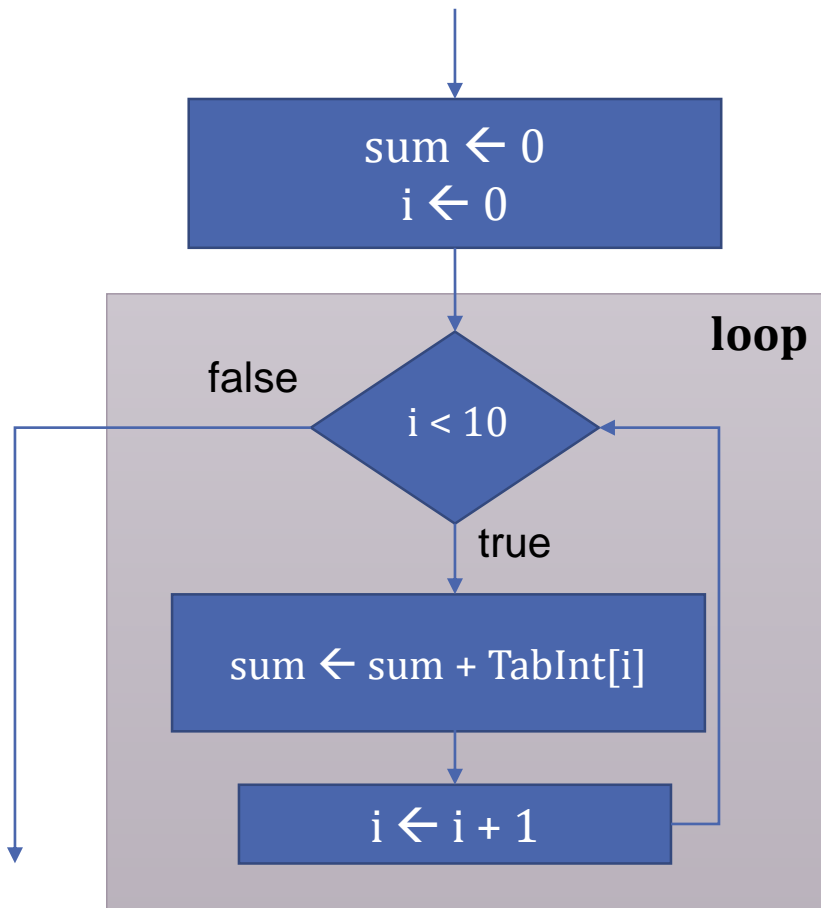
# Menjumlahkan elemen array
sum = 0
for i in range(0,10):
    sum = sum + TabInt[i]

# Menghitung nilai rata-rata dan menampilkannya
rata = sum/10
print ("Nilai rata-rata = " + str(rata))
```

Menghitung Rata-Rata (2)



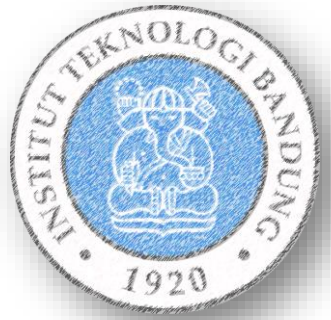
Flowchart – Bagian penjumlahan elemen:



Pseudocode – Bagian penjumlahan elemen:

```
... { Bagian mengisi array }  
sum ← 0  
i traversal [0..9]  
    sum ← sum + TabInt[i]  
...
```

Bagian yang lain silakan dibuat sebagai latihan.



Latihan

- Untuk setiap soal latihan, buatlah flowchart/pseudocode/kode program dalam Python

Latihan-1

- Buatlah sebuah program yang berisi sebuah array dengan elemen integer berukuran 20, misalnya **T**
- Anggaplah sudah ada bagian program yang digunakan untuk mengisi array **T**
 - Lihat slide sebelumnya untuk pengisian array dari keyboard
- Program menerima masukan sebuah integer, misalnya **X**
- Selanjutnya, program mengalikan semua elemen array **T** dengan **X** dan mencetak semua elemen **T** yang baru ke layar.

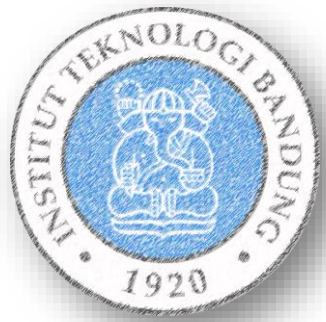
Contoh:

$T = [4, 1, 3, 4, 5, 6, 8, 9, 12, 30, -1, 0, 4, -1, 3, 10, 14, 6, 7, 0]$

$X = 3$

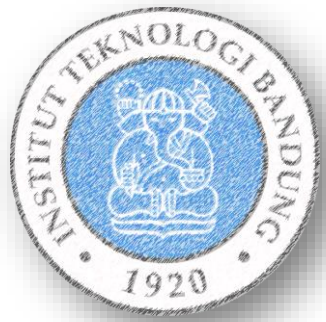
Setelah elemen **T** dikalikan **X**

$T = [12, 3, 9, 12, 15, 18, 24, 27, 36, 90, -3, 0, 12, -3, 9, 30, 42, 18, 21, 0]$



Latihan-2

- Nilai mahasiswa untuk suatu mahasiswa dinyatakan dalam bentuk huruf, yaitu A, B, C, D, dan E.
 - Di ITB ada nilai AB dan BC, tapi untuk menyederhanakan persoalan kedua nilai tersebut diabaikan
- Sebuah program menerima data nilai 50 mahasiswa di sebuah kelas dalam bentuk indeks huruf seperti di atas dan disimpan dalam sebuah **array of character**.
- Tentukanlah **berapa banyak** mahasiswa yang lulus dan berapa yang tidak lulus. Mahasiswa dinyatakan lulus jika mendapatkan nilai A, B, atau C. Selebihnya, tidak lulus.



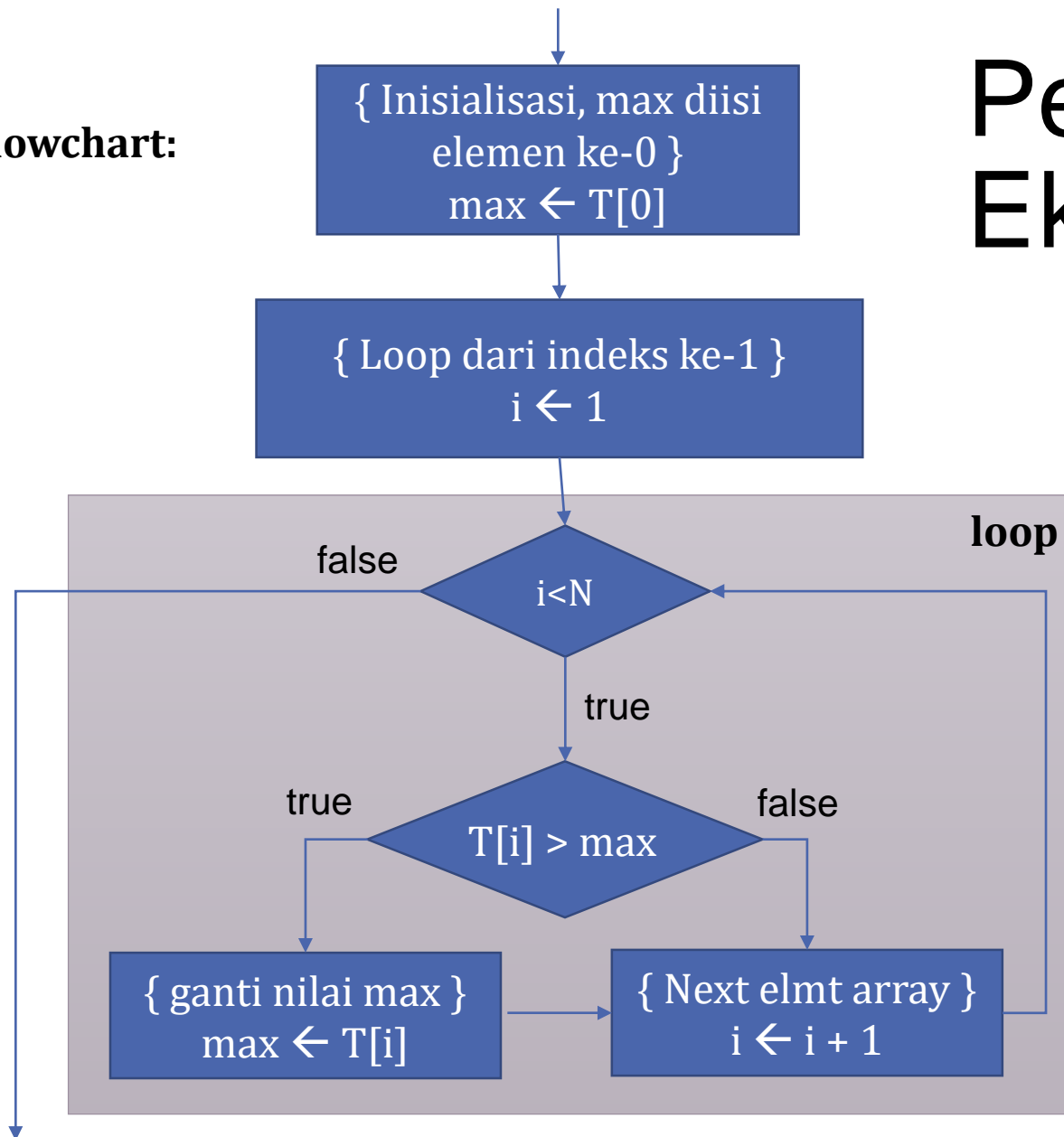
Pencarian Nilai Ekstrem (1)

(Minimum atau Maksimum)

- Mencari nilai **terbesar** atau **terkecil** dari elemen suatu array
- Diketahui:
 - Sebuah array T dengan ukuran N elemen
 - Nilai X (bertipe sama dengan elemen T)
- Buatlah program untuk menuliskan ke layar **nilai terbesar** dari elemen T
- Asumsi: T tidak kosong (minimum 1 elemen, $N > 0$)
- Contoh:
 - $N = 10$; T berisi: [9,12,30,-1,0,4,-1,3,30,14] maka nilai terbesar = 30
 - $N = 8$; T berisi: [1, 3, 5, 8, -12, 90, 3, 5] maka nilai terbesar = 90

Pencarian Nilai Ekstrem (2)

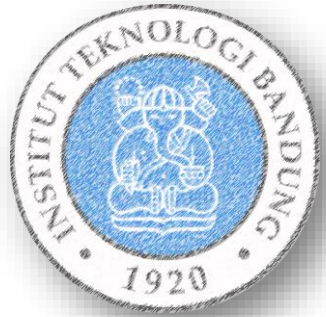
Flowchart:



Pseudocode:

```
...
{ inisialisasi max dgn elemen ke-0 }
max ← T[0]

i traversal [1..N-1]
{ ganti kalau ketemu nilai T[i]>max }
if (T[i]>max) then
    max ← T[i]
...
```

```
# Program MaxArray
# Mencari nilai terbesar pada array

# KAMUS
# N : int
# T : array [0..N-1] of int
# i : int
# max : int

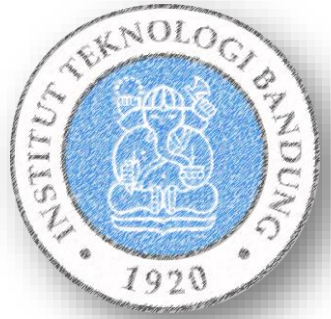
# ALGORITMA
N = 10 # assign N dengan ukuran T
# Asumsi: pengisian array sudah dibuat
# Tetap harus dibuat untuk mengetes program

# Mencari nilai maksimum
max = T[0] # init max dgn elemen pertama

# Pencarian dimulai dari elemen ke-2
for i in range(1,N):
    # jika ada elemen > max, ganti nilai max
    if (T[i] > max):
        max = T[i]

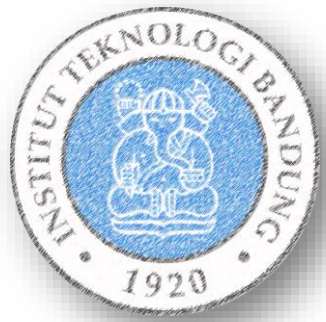
# Cetak nilai terbesar
print ("Nilai terbesar = " + str(max))
```

Pencarian Nilai Ekstrem (3) Program Python



Latihan-3

- Berdasarkan contoh soal sebelumnya, buatlah program untuk mencari nilai **terkecil** dari elemen T.



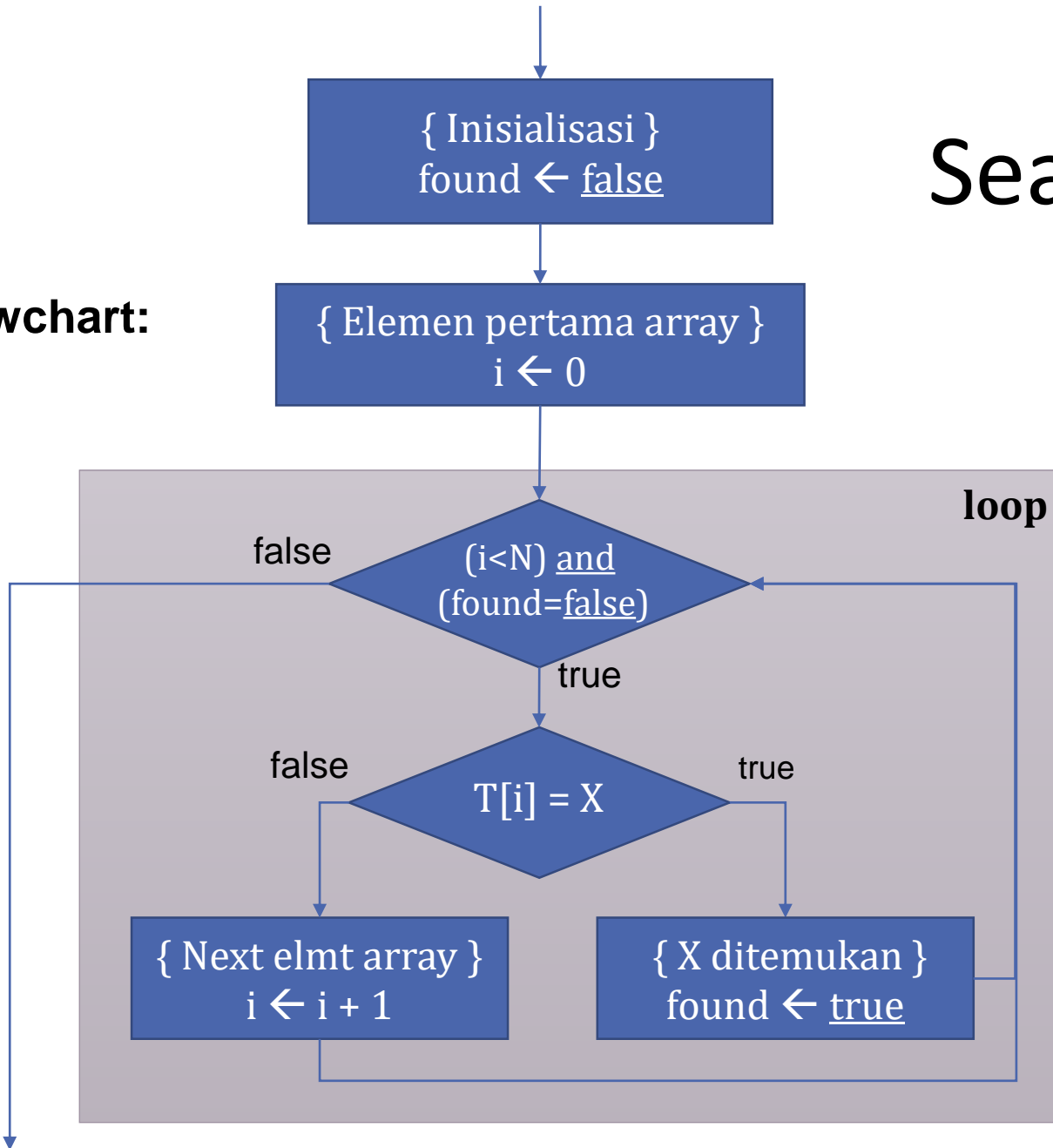
Searching (1)

Mencari Indeks Pertama Kemunculan Nilai

- **Searching** adalah proses yang penting dalam pemrosesan tabel karena sering dilakukan terhadap sekumpulan data yang disimpan dalam tabel
- Diketahui:
 - Sebuah array T dengan ukuran N elemen
 - Nilai X (bertipe sama dengan elemen T)
- Buatlah program untuk menuliskan ke layar **indeks pertama** di T di mana X ditemukan
- Asumsikan: Array tidak kosong (minimum 1 elemen, $N > 0$)
- Contoh:
 - $N = 10$; T berisi: [9,12,30,-1,0,4,-1,3,30,14]; $X = -1$ maka X ditemukan pertama kali di indeks 3
 - $N = 8$; T berisi: [1, 3, 5, 8, -12, 90, 3, 5]; $X = 0$ maka X tidak ditemukan di T

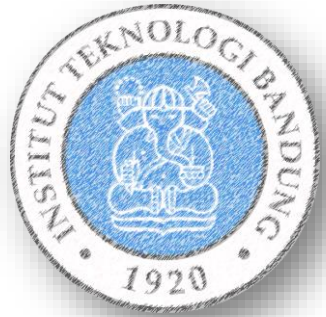
Searching (2)

Flowchart:



Pseudocode:

```
...  
i <- 0  
found <- false  
while (i < N) and (found = false) do  
    if (T[i] = X) then  
        found <- true  
    else { T[i] ≠ X }  
        i <- i + 1  
{ i ≥ N or found = true }  
...
```



```
# Program SearchArray
# Mencari indeks di mana X ditemukan pertama kali di T
# KAMUS
# N : int; ukuran T
# T : array [0..N-1] of int
# i, X : int
# found : bool; menentukan X sdh ditemukan/belum

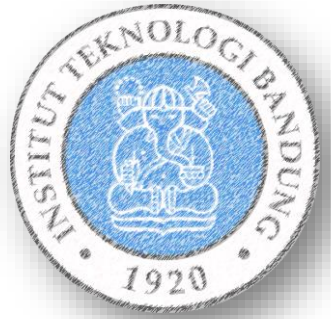
# ALGORITMA
# Asumsi: input array sudah dibuat; N terdefinisi

# Membaca nilai yang dicari, yaitu X
X = int(input())

# Pencarian dimulai dari elemen ke-2
i = 0
found = False # found = False; X belum ditemukan
while (i < N and found == False):
    if (T[i] == X):
        found = True # found = True; X sudah ditemukan
    else:
        i = i + 1 # hanya increment jika X belum
ditemukan
# i = N atau found = True

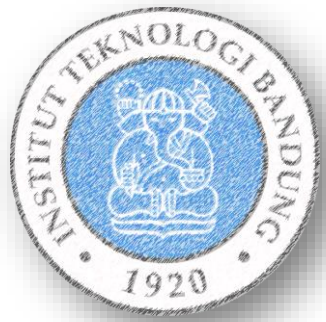
# Cetak Hasil
if (found == True): # X ditemukan di T
    print (str(X) + " ditemukan di indeks ke-" + str(i))
else: # found = False; X tidak ditemukan di T
    print (str(X) + " tidak ditemukan")
```

Searching (3) Program Python



Latihan-4

- Berdasarkan contoh sebelumnya, buatlah program untuk mencari indeks **terakhir** di mana X ditemukan di T.
- Petunjuk: proses pencarian dilakukan “mundur”, yaitu dari indeks elemen terakhir ke elemen pertama.
 - Modifikasi apa yang harus dilakukan terhadap algoritma sebelumnya?

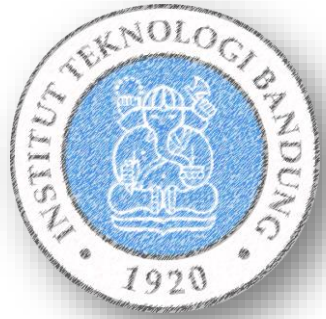


Latihan-5

- Sebuah vektor $v = (v_0, v_1, v_2, v_3, v_4)$ direpresentasikan sebagai suatu array of integer dengan 5 buah elemen.
- Diketahui dua buah vektor, masing-masing terdiri atas 5 elemen, misalnya V dan U .
- Tuliskan hasil penjumlahan kedua vektor.
- Penjumlahan dua vektor menghasilkan vektor lain, W , dengan elemen ke- i adalah: $W_i = V_i + U_i$

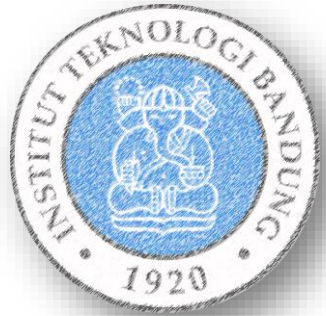
$$W = U + V = (v_0, v_1, v_2, v_3, v_4) + (u_0, u_1, u_2, u_3, u_4)$$

$$W = (v_0+u_0, v_1+u_1, v_2+u_2, v_3+u_3, v_4+u_4)$$



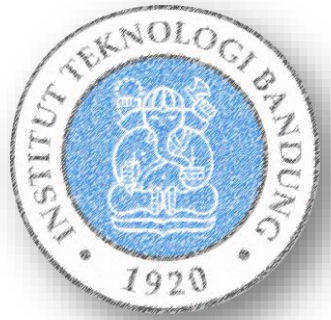
Latihan-6 (1)

- BMKG Kota Bandung setiap hari mencatat suhu harian kota Bandung (dalam derajat Celsius) berdasarkan data dari berbagai sensor temperatur. Data suhu harian ini dibutuhkan untuk berbagai analisis iklim dan cuaca.
- Sebuah program digunakan untuk mencatat suhu kota Bandung selama bulan September 2018 (30 hari).
- Data suhu dalam bentuk bilangan riil.

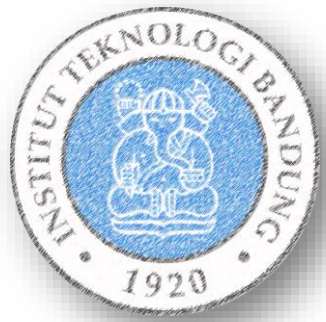


Latihan-6 (2)

- Tuliskan:
 - **Rata-rata** suhu kota Bandung di bulan Sept. 2018
 - Suhu **terendah** di bulan Sept. 2018.
 - Pada **tanggal berapa saja** di bulan Sept. 2018, suhu harian kota Bandung ≥ 30 **derajat Celsius**.
 - Pada tanggal berapa **pertama kali** di bulan Sept. 2018, kota Bandung mengalami suhu **di bawah 15 derajat Celcius** (jika terjadi). Jika tidak pernah terjadi, tuliskan: “Suhu tidak pernah di bawah 15 derajat Celcius”.
- Perhatian:
 - Tanggal dalam bulan September 2018 adalah dari tanggal 1 s.d. 30. Jika tanggal direpresentasikan sebagai indeks array, perhatikan bahwa indeks array di Python dimulai dari 0 (apa yang harus dilakukan?).



Matriks



Definisi

- **Matriks:**

- Sekumpulan informasi yang setiap individu elemennya terdefinisi berdasarkan dua buah **indeks** (yang biasanya dikonotasikan dengan **baris** dan **kolom**)
 - Setiap elemen matriks dapat diakses secara langsung jika kedua indeks diketahui.
 - Setiap elemen matriks mempunyai type yang homogen
 - Indeks baris dan kolom harus bertipe yang mempunyai keterurutan (suksesor/predesesor), misalnya integer.
- Matriks adalah struktur data dengan memori internal. Struktur ini praktis untuk dipakai tetapi memakan memori!
 - Matriks integer 100 x 100 memakan 10000 x tempat penyimpanan integer.

Contoh-1

- MatUkur
 - Indeks (i, j) merepresentasikan suatu titik koordinat
 - Elemen matriks merepresentasikan hasil pengukuran pada suatu titik koordinat tertentu
 - Indeks baris : 1 s.d. 5, indeks kolom : 1 s.d. 5
 - Elemen matriks ber-type real

	1	2	3	4	5
1	12.1	7.0	8.9	0.7	6.6
2	0.0	1.6	2.1	45.9	55.0
3	6.1	8.0	0.0	3.1	21.9
4	9.0	1.0	2.7	22.1	6.2
5	5.0	0.8	0.8	2.0	8.1

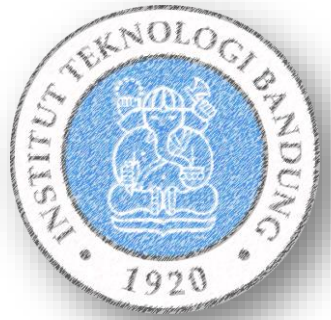
Contoh-2

- MatSat
 - Merupakan matriks dengan elemen bernilai hanya 0 atau 1
 - Indeks baris : 1 s.d. 4; indeks kolom 1 s.d. 4
 - Elemen matriks ber-type integer

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Contoh Pemakaian

- Operasi “biasa” matriks dalam matematika : penjumlahan, perkalian, menentukan determinan, menginvers sebuah matriks, transpose, dll.
 - Semua "perhitungan" itu menjadi tidak primitif, harus diprogram
- Sistem persamaan linier dan *operational research*
- Persoalan algoritmik: untuk menyimpan informasi yang cirinya ditentukan oleh 2 dimensi (diterjemahkan dalam baris dan kolom).
Contoh: *cell* pada sebuah *spreadsheet*, ruangan gedung bertingkat



Implementasi Matriks di Python (1)

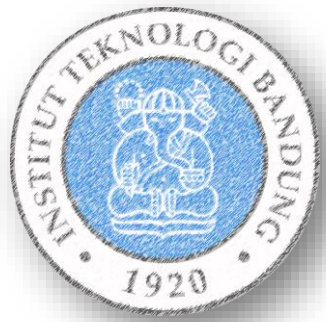
- Memori matriks diimplementasikan sebagai **array 2 dimensi**
- Suatu elemen matriks diakses dengan indeks baris dan kolom bertipe integer
- Elemen matriks dideklarasikan memiliki type yang sama (homogen)

Cara Deklarasi (1)

- Cara deklarasi sekaligus inisialisasi nilai matriks ukuran $n \times m$:

```
<nama-var> = [ [<val-11>, <val-12>, ..., <val-1m>] ,  
                [<val-11>, <val-12>, ..., <val-1m>] ,  
                ...  
                [<val-n1>, <val-n2>, ..., <val-nm>] ]
```

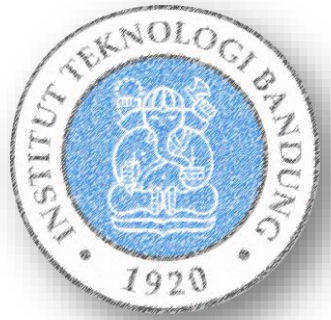
- Akan dideklarasikan array dengan ukuran sebesar $n \times m$
- Type elemen tergantung pada nilai yang diberikan



Contoh Matriks-1

```
MatSatuan = [[0,1,0,1,1,1,0],  
              [1,1,1,1,0,1,1],  
              [0,0,0,1,1,0,1]]
```

Matriks bernama `MatSatuan` dengan setiap elemen bertipe `integer`, dengan `ukuran baris = 3` dan `ukuran kolom = 7`; dengan alamat setiap elemen diakses melalui `indeks baris 0 s.d. 2` dan `indeks kolom 0 s.d. 6`.

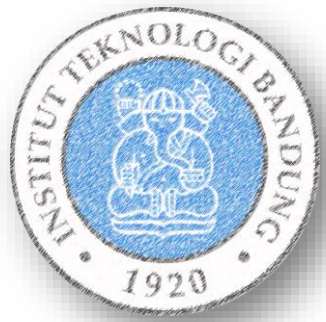


Cara Deklarasi (2)

- Cara deklarasi:

<nama-var> = [[<default-val> for j in range (<m>)] for i in range(<n>)]

- Akan dideklarasikan array dengan ukuran n x m.
- Setiap elemen diberikan nilai 0.
- Type elemen integer



Contoh Matriks 2

```
MatValue = [[0 for j in range (4)] for i in range(3)]
```

Matriks bernama **MatValue** dengan setiap elemen bertipe **integer**, dengan **ukuran baris = 3** dan **ukuran kolom = 4**; dengan alamat setiap elemen diakses melalui **indeks baris 0 s.d. 2** dan **indeks kolom 0 s.d. 3** dan setiap elemen diberi nilai **0**.

Implementasi Matriks di Python (3)

- Cara akses elemen matriks:

`<namamatriks>[<nbrs>][<nkol>]`

- Contoh: `M1` dengan data sbb:

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

```
print (M1[4][0])
```

akan tercetak: 21

```
x = M1[0][0] * M1[3][3]
```

x bernilai 19

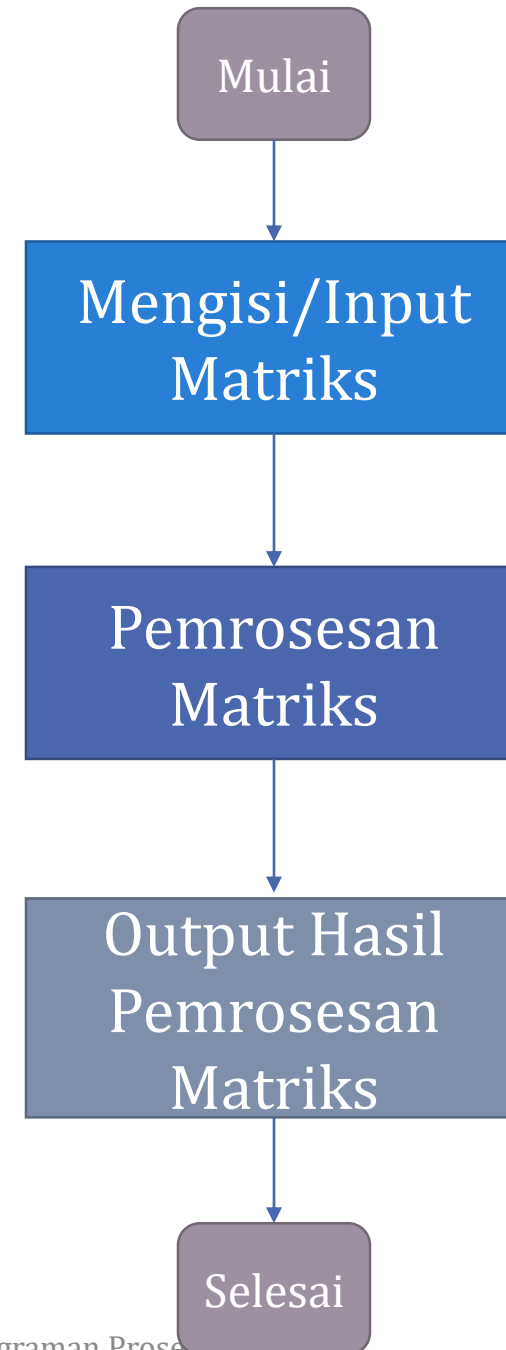
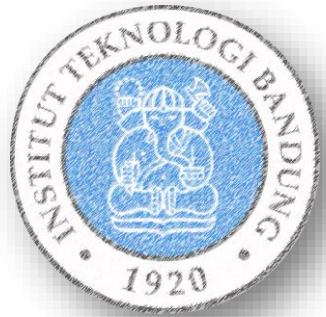
```
M1[1][1] = 8
```

Elemen brs. 1, kol. 1 menjadi 8

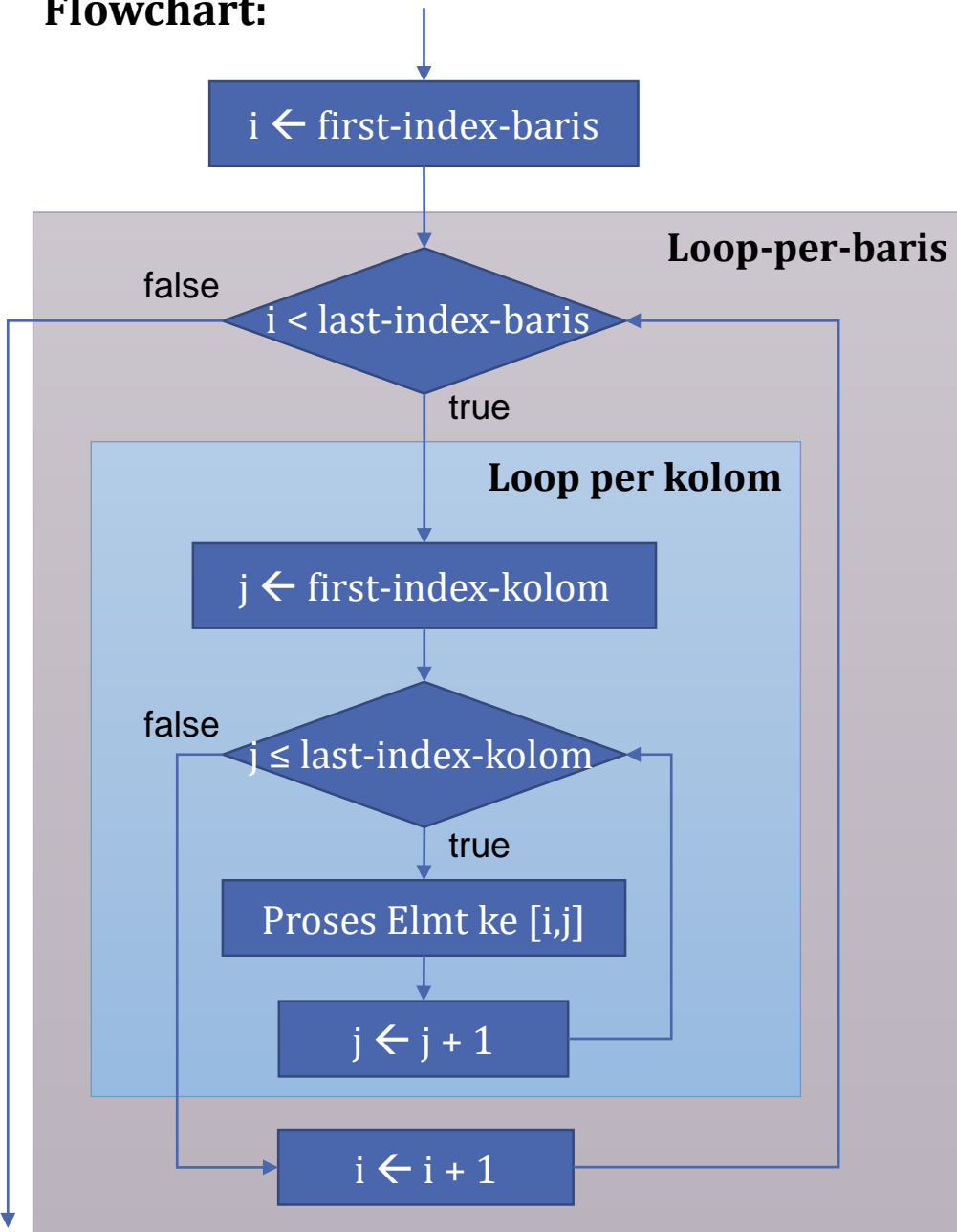
```
M1[0][5] ??
```

tidak terdefinisi! Tidak ada kolom dengan indeks = 5

Pemrosesan Matriks



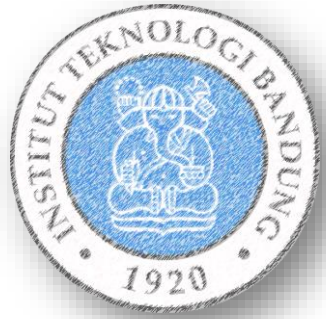
Flowchart:



Pemrosesan Dasar Matriks

Pseudocode:

```
i traversal [first-index-baris..last-index-baris]
    j traversal [first-index-kolom..last-index-kolom]
        { Proses Element ke [i,j] }
        ...
```



Operasi-Operasi pada Matriks

- Mendeklarasikan dan mendefinisikan isi matriks
- Menuliskan isi matriks ke layar
- Operasi 1 matriks:
 - Menghitung total semua elemen matriks
 - Mengalikan isi matriks dengan sebuah konstanta
 - Transpose matriks
- Operasi 2 matriks:
 - Menambahkan dua matriks
 - Mengalikan dua matriks

Isi dan Tulis Matriks (1)

- Deklarasi matriks dan inisialisasi ukuran baris dan kolom
- Membaca isi matriks dari hasil kalkulasi
- Menampilkan ke layar

```
# Program IsiMatriks;
# Isi matriks dan menulis ke layar

# KAMUS
#   M : matriks of integer
#   NBrS, NKol : int (ukuran brs & kol)
#   i, j : int (indeks)

# ALGORITMA

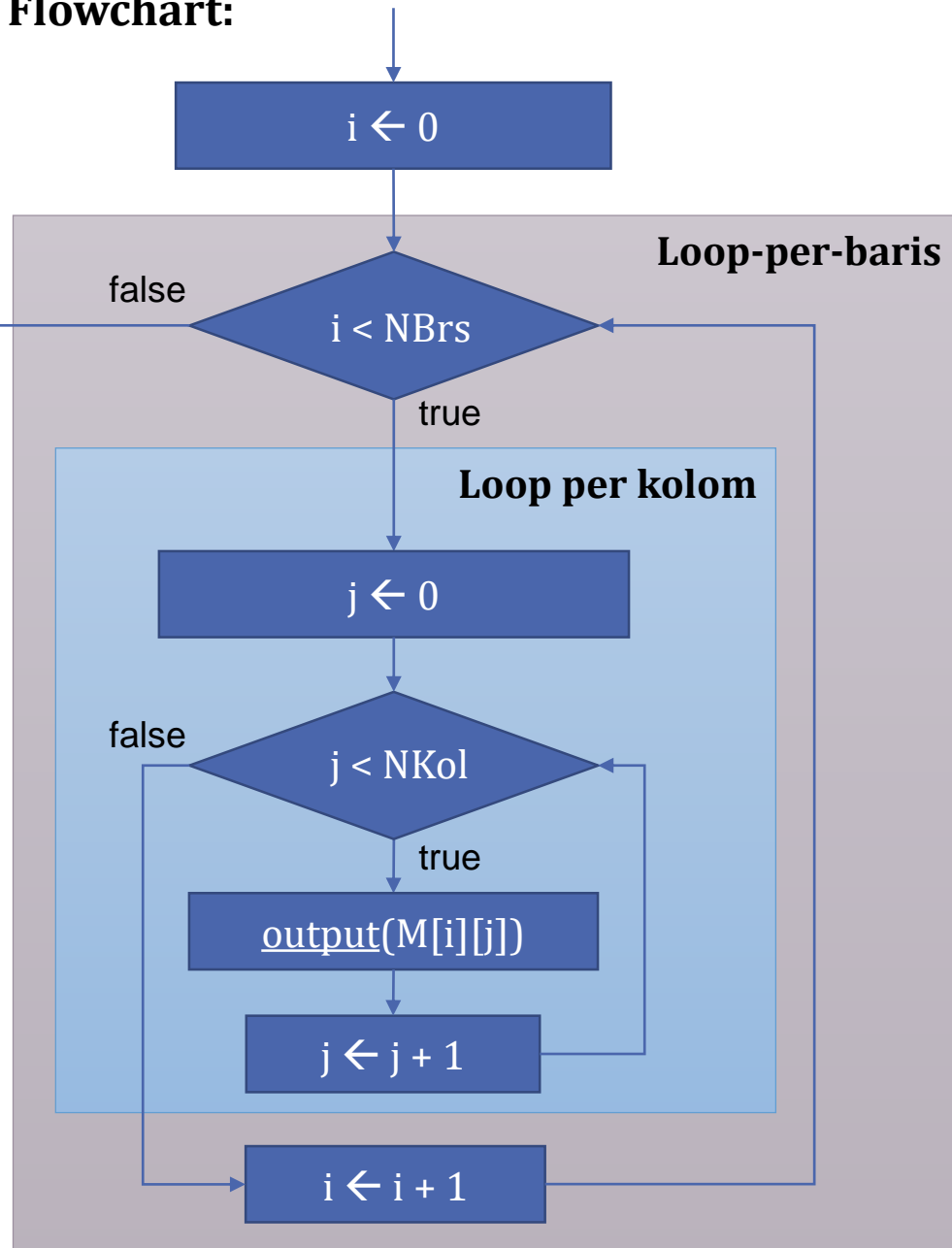
# deklarasi matriks
NBrS = 5; NKol = 5;
M = [[0 for j in range(NKol)] for i in range(NBrS)]

# Mengisi matriks ukuran NBrSxNKol
for i in range (NBrS):
    for j in range (NKol):
        M[i][j] = i * j

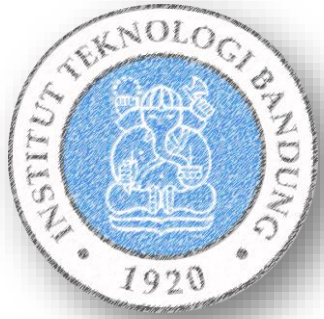
# Menuliskan isi matriks ke layar
for i in range (NBrS):
    for j in range (NKol):
        print(str(M[i][j])+" ", end='')
    print() # print hanya enter
```

Print tanpa
enter

Flowchart:



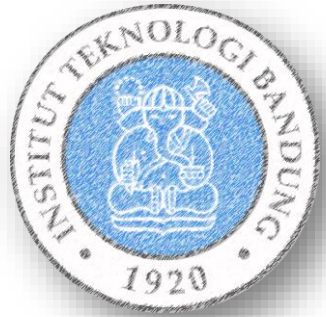
Isi dan Tulis Matriks (2) Bagian Tulis Matriks



Pseudocode:

```
...  
{ Menuliskan isi matriks ke layar }  
i traversal [0..NBrS-1]  
  j traversal [0..NKol-1]  
    { Tulis Element ke [i,j] }  
    output(M[i][j])  
  ...
```

Bagian kode yang lain silakan dibuat sebagai latihan



Note

- Untuk contoh-contoh selanjutnya, hanya disediakan kode program Python
- Silakan membuat *flowchart* dan *pseudocode* yang bersesuaian silakan dibuat sebagai latihan.

Baca dan Tulis Matriks

- Deklarasi matriks
- Inisialisasi ukuran baris dan kolom dari masukan user
- Membaca isi matriks dari user
- Menampilkan isi matriks ke layar

```
# Program BacaMatriks;
# Baca isi matriks dari pengguna dan menulis ke layar
# KAMUS
#   M : matriks of integer
#   NBrS, NKol : int (ukuran brs & kol)
#   i, j : int (indeks)

# ALGORITMA

# deklarasi matriks
NBrS = 5; NKol = 5;
M = [[0 for j in range(NKol)] for i in range(NBrS)]

# Mengisi matriks ukuran NBrSxNKol
for i in range (NBrS):
    for j in range (NKol):
        M[i][j] = int(input("Elemen ke-["+str(i)+", "+str(j)+"] = "))

# Menuliskan isi matriks ke layar
for i in range (NBrS):
    for j in range (NKol):
        print(str(M[i][j])+" ", end='') # print tanpa enter
    print() # print hanya enter
```

Sum Element Matriks

- Menjumlahkan seluruh elemen yang ada di matriks dan menampilkan hasilnya ke layar

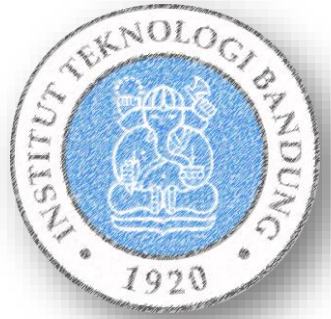
```
# Program SumElmt
# Menjumlahkan isi elemen matriks ke layar
# KAMUS
#   M : matriks of integer
#   NBrS, NKol : int (ukuran brs & kol)
#   i, j : int (indeks)
#   sum : int (jumlah elemen)

# ALGORITMA

# deklarasi matriks
# Mengisi matriks ukuran NBrSxNKol
# Buat sebagai latihan

# Menjumlahkan elemen M
sum = 0
for i in range (NBrS):
    for j in range (NKol):
        sum = sum + M[i][j]

# Cetak nilai sum
print(sum)
```



Latihan-1

- Buatlah program yang membaca sebuah matriks dengan elemen integer, misalnya M, dan masukan sebuah nilai integer, misalnya X dan selanjutnya mengalikan setiap elemen matriks M dengan X.
- Buatlah kode Python dan flowchart/pseudocode (sesuai arahan dosen kelas).

Latihan-1

- Program Python

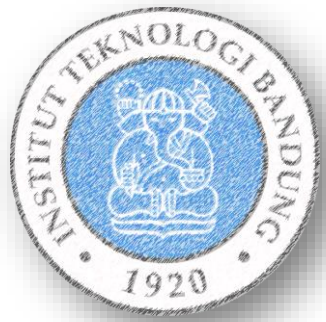
```
# Program KaliKons
# Mengalikan setiap elemen matriks dengan faktor pengali
# KAMUS
#   M : matriks of integer
#   NBrS, NKol : int (ukuran brs & kol)
#   i, j : int (indeks)
#   x : int (faktor elemen)

# ALGORITMA
# deklarasi matriks
# Mengisi matriks ukuran NBrSxNKol
# Buat sebagai latihan

# Input x
x = int(input("Faktor pengali = "))

# Menjumlahkan elemen M
for i in range (NBrS):
    for j in range (NKol):
        M[i][j] = M[i][j] * x

# Mencetak matriks baru ke layar
# Buat sebagai latihan
```



Transpose Matriks (1)

- Dideklarasikan 2 buah matriks, misalnya M dan MTranspose
- MTranspose menampung hasil transpose dari M
 - Ukuran baris M = ukuran kolom MTranspose
 - Ukuran kolom M = ukuran baris Mtranspose
- $Mtranspose[i][j] = M[j][i]$

Transpose Matriks (2)

- Program Python

```
# Program Transpose
# Transpose Matriks

# KAMUS
#   M : matriks of integer
#   NBrS, NKol : int (ukuran brs & kol)
#   MT : matriks of integer (matriks hasil transpose)
#   NBrST, NKolT : int (ukuran brs & kol hasil transpose)
#   i, j : int (indeks)

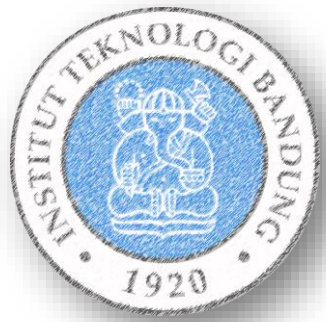
# ALGORITMA

# Deklarasi matriks M
# Mengisi matriks ukuran NBrSxNKol
# Buat sebagai latihan

# Deklarasi MT
NBrST = NKol; NKolT = NBrS;
MT = [[0 for j in range(NKolT)] for i in range(NBrST)]

# Isi MT dengan hasil transpose M
for i in range (NBrST):
    for j in range (NKolT):
        MT[i][j] = M[j][i]

# Mencetak matriks MT ke layar
# Buat sebagai latihan
```

Menjumlahkan 2 Matriks (1)

- Deklarasi 3 matriks, misal M1, M2, dan MHasil
- 2 matriks M1 dan M2 bisa dijumlahkan jika dimensinya sama, yaitu:
 - Ukuran baris M1 = ukuran baris M2
 - Ukuran kolom M1 = ukuran kolom M2
- Hasil ditampung di MHasil: $MHasil[i][j] = M1[i][j] + M2[i][j]$
- Mengurangi M1 dengan M2 secara prinsip sama dengan menjumlahkan M1 dengan M2

Menjumlahkan 2 Matriks (2)

Program Python

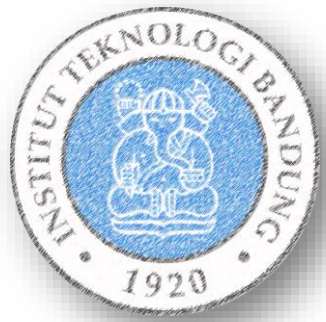
```
# Program Transpose
# Transpose Matriks
# KAMUS
#   M1, M2 : matriks of integer
#   NBrs1, NKol1 : int (ukuran brs & kol M1)
#   NBrs2, NKol2 : int (ukuran brs & kol M2)
#   MH : matriks of integer (matriks hasil penjumlahan)
#   NBrsH NKolH : int (ukuran brs & kol hasil penjumlahan)
#   i, j : int (indeks)

# ALGORITMA
# Deklarasi matriks M1 dan Mengisi matriks M1 ukuran NBrs1xNKol1
# Buat sebagai latihan1
# Deklarasi matriks M2 dan Mengisi matriks M2 ukuran NBrs2xNKol2
# Buat sebagai latihan

# Deklarasi MH
NBrsH = NBrs1; NKolH = NKol1;
MH = [[0 for j in range(NKolH)] for i in range(NBrsH)]

# Isi MH dengan hasil penjumlahan M1+M2
for i in range (NBrsH):
    for j in range (NKolH):
        MH[i][j] = M1[i][j] + M2[i][j]

# Mencetak MH ke layar
# Buat sebagai latihan
```



Mengalikan 2 Matriks (1)

- Deklarasi 3 matriks, misal M1, M2, dan MHasil
- 2 matriks M1 dan M2 bisa dikalikan jika:
 - Ukuran kolom M1 = ukuran baris M2
- MHasil menampung hasil perkalian antara M1 dan M2
 - Ukuran baris MHasil = ukuran baris M1
 - Ukuran kolom MHasil = ukuran kolom M2

Mengalikan 2 Matriks (2)

- Beberapa contoh:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} a\alpha + b\gamma & a\beta + b\delta \\ c\alpha + d\gamma & c\beta + d\delta \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} a & d \\ b & e \\ c & f \end{pmatrix} = \begin{pmatrix} 1a + 2b + 3c & 1d + 2e + 3f \\ 4a + 5b + 6c & 4d + 5e + 6f \\ 7a + 8b + 9c & 7d + 8e + 9f \end{pmatrix}$$

Mengalikan 2 Matriks (3) Program Python

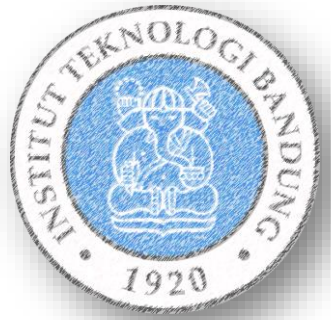
```
# Program KaliMatriks
# Perkalian matriks M1 dengan M2
# KAMUS
#   M1, M2 : matriks of integer
#   NBrs1, NKol1 : int (ukuran brs & kol M1)
#   NBrs2, NKol2 : int (ukuran brs & kol M2)
#   MH : matriks of integer (matriks hasil perkalian)
#   NBrsH NKolH : int (ukuran brs & kol hasil perkalian)
#   i, j, k : int (indeks)

# ALGORITMA
# Deklarasi matriks M1 dan Mengisi matriks M1 ukuran NBrs1xNKol1
# Buat sebagai latihan
# Deklarasi matriks M2 dan Mengisi matriks M2 ukuran NBrs2xNKol2
# Buat sebagai latihan

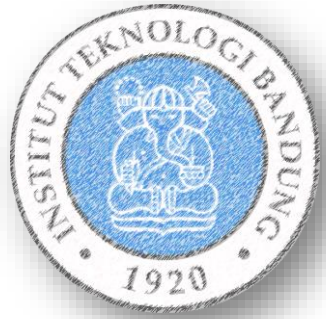
# Deklarasi MH
NBrsH = NBrs1; NKolH = NKol2;
MH = [[0 for j in range(NKolH)] for i in range(NBrsH)]

# Isi MH dengan hasil perkalian M1 dan M2
for i in range (NBrsH):
    for j in range (NKolH):
        MH[i][j] = 0
        for k in range (NKol1):
            MH[i][j] = MH[i][j] + (M1[i][k]*M2[k][j])

# Mencetak MH ke layar
# Buat sebagai latihan
```

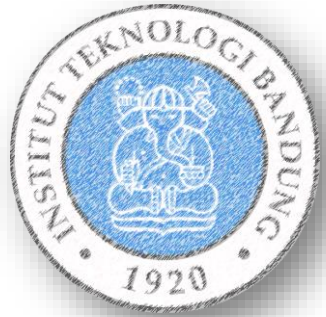


Subprogram



Kode yang berulang

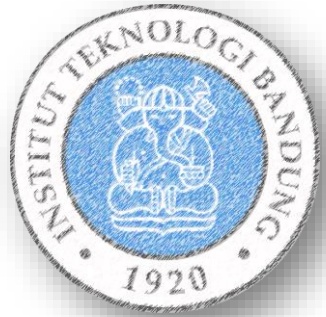
- Semakin besar program, semakin banyak **bagian kode yang berulang**
- Sangat tidak efisien jika bagian kode yang sama/serupa **diketik berulang-ulang**, (bahkan kalau di-*copy-paste*)
- Di samping itu, dalam banyak persoalan, ada berbagai **rumus/formula** yang berulang-ulang dipakai dalam satu program
- Bagaimana jika ada cara supaya bagian kode tersebut tidak perlu diketik berulang-ulang, tapi tetap dapat digunakan berkali-kali dalam program yang sama



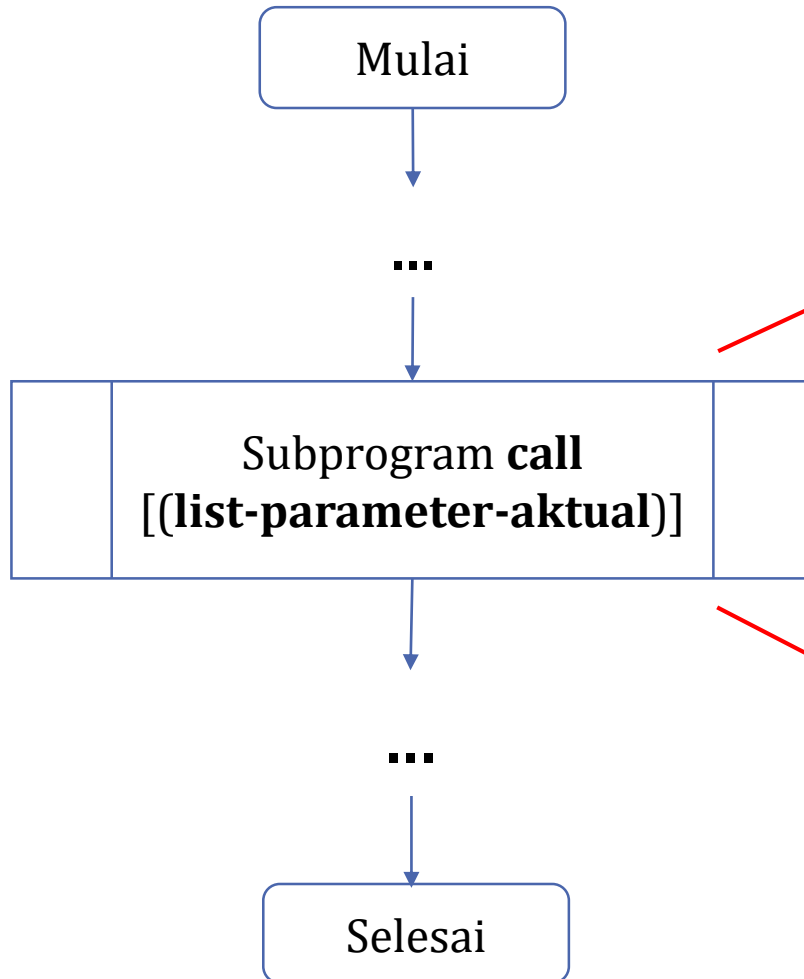
Subprogram

- *A set of instructions designed to perform a frequently used operation within a program*
- 2 (dua) jenis subprogram:
 - **Fungsi:** pemetaan suatu nilai domain (input) ke range (output)
 - Hasil dari fungsi dinyatakan dalam sebuah type data yang eksplisit
 - **Prosedur:** deretan instruksi yang jelas initial state dan final state-nya → mirip seperti program secara umum, namun dalam scope yang lebih kecil

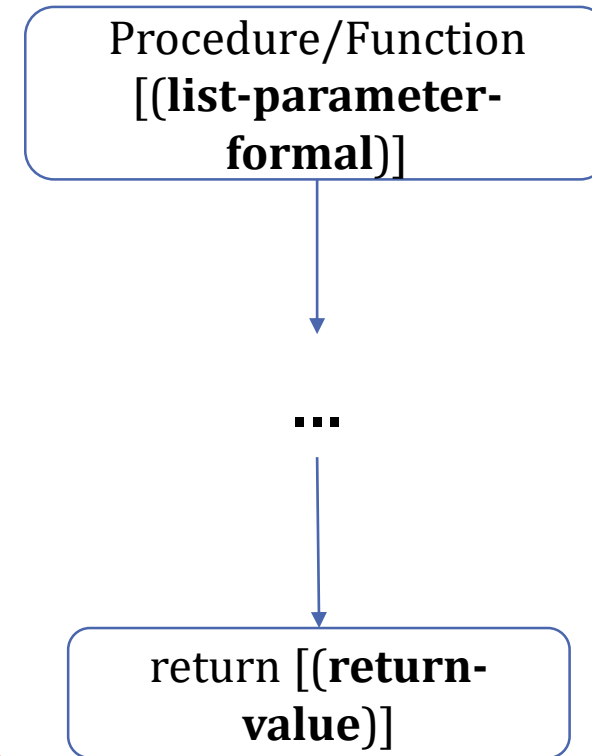
Flowchart Symbol (Umum)

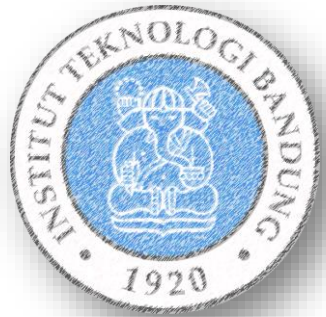


Pemanggilan subprogram dalam algoritma program utama



Flowchart terpisah untuk **pendefinisian** dan **realisasi** subprogram

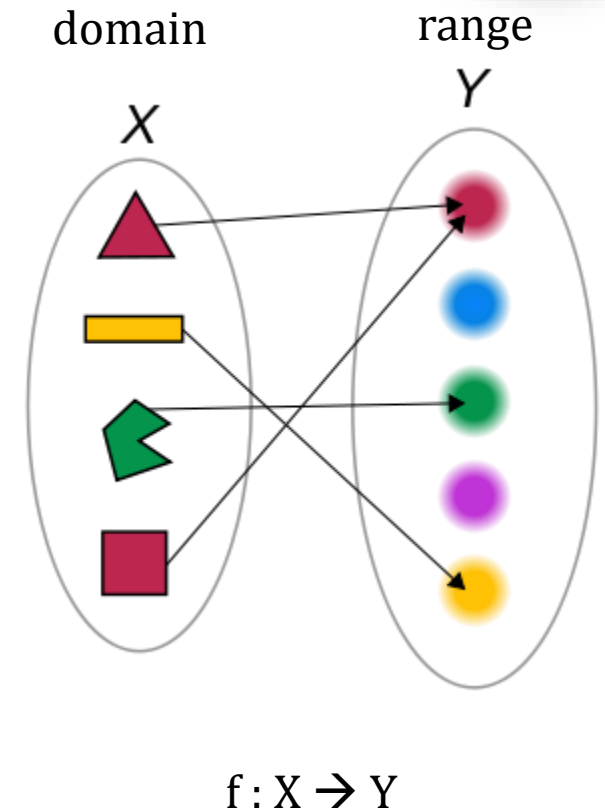




Fungsi

Fungsi

- Konsep fungsi di pemrograman didasari oleh konsep pemetaan dan **fungsi** di matematika
- **Fungsi**: asosiasi (pemetaan) antara 2 himpunan nilai yaitu **domain** dan **range**
 - Setiap elemen pada himpunan domain dipetakan **tepat satu** ke sebuah elemen pada himpunan range
- Contoh: $f(X) = X^2$
 - fungsi untuk menghitung kuadrat dari suatu bilangan
 - Domain: bilangan bulat
 - Range: bilangan bulat (0 atau positif)

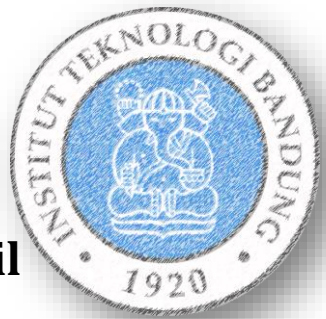


Fungsi dalam Pemrograman



- Memrogram fungsi pada dasarnya adalah: **merakit isi black box**
 - Berangkat dari keadaan awal → himpunan nilai yang terdefinisi sebagai **input** (domain)
 - Menghasilkan nilai-nilai yang mendefinisikan keadaan akhir → himpunan nilai yang terdefinisi sebagai **output** (range)
 - Tugas pemrogram fungsi adalah menentukan langkah-langkah untuk menghasilkan keadaan akhir berdasarkan keadaan awal
- Fungsi didefinisikan sebagai bagian terpisah dari program dan dipanggil dalam **program utama**

Fungsi dalam program



```
# Program Test
# Mengetes fungsi kuadrat
```

```
# KAMUS
```

```
# A : integer
```

```
# B : integer
```

```
# Fungsi Kuadrat
```

```
def Kuadrat ( X ) :
    # menghitung kuadrat X
    hasil = X * X
    return hasil
```

```
# ALGORITMA PROGRAM UTAMA
```

```
A = 5
B = Kuadrat (A) + 10
```

Function flow of control:

- 1) Salah satu baris pada kode program utama **memanggil** fungsi: **B = Kuadrat(A) + 10** # A = 5
- 2) Program beralih ke kode fungsi **Kuadrat** mulai dari baris yang pertama sampai pada baris yang mendefinisikan hasil fungsi (return). Parameter input diasosiasikan dengan daftar parameter input pada fungsi.

```
def Kuadrat ( X ):
    # menghitung kuadrat X
    hasil = X * X
    return hasil
```

Asosiasi
input

```
def Kuadrat ( 5 ):
    # menghitung kuadrat 5
    hasil = 5 * 5
    return hasil    #hasil=25
```

- 3) Program meninggalkan fungsi dengan menyimpan hasil perhitungan dan kembali pada baris terakhir program utama yang ditinggalkannya dan menggantikan hasil perhitungan berdasarkan hasil fungsi: **B = 25 + 10** # B = 35
- 4) Program melanjutkan ke instruksi berikutnya.

Contoh (1)

- Buatlah program yang menerima masukan buah nilai jari-jari lingkaran (bilangan riil), misalnya r , dan menuliskan luas lingkaran ke layar
- Perhitungan luas lingkaran → dibuat menjadi fungsi

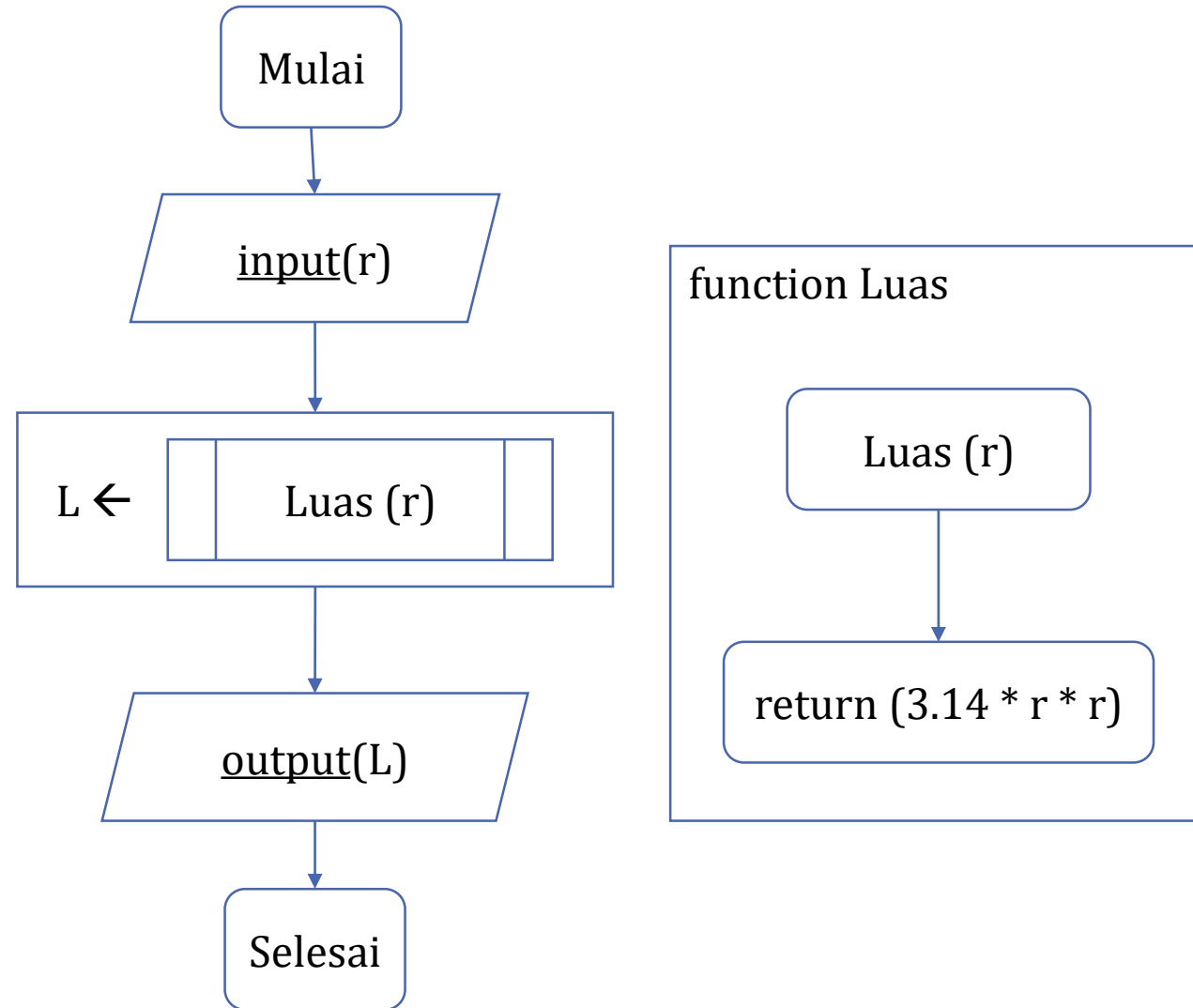
```
# Program LuasLingkaran
# Menghitung luas lingkaran berdasarkan jari-jari

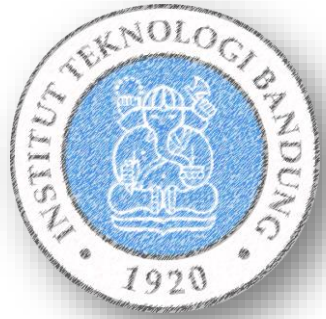
# KAMUS
# r, L : float
# Definisi dan Realisasi Fungsi Luas
def Luas ( r ):
    # menghasilkan luas lingkaran berdasarkan jari-jari r
    Luas := 3.14 * r * r
    return (Luas)

# PROGRAM UTAMA
r = float(input())
L = Luas(r) # pemanggilan fungsi Luas
print(L)
```

Contoh (2)

- Flowchart





Kegunaan Fungsi

- Program dapat didekomposisi menjadi sub-sub bagian
 - Tiap sub bagian dapat didefinisikan sebagai fungsi yang tinggal dipanggil sebagai 1 baris atau ekspresi dalam program utama
- *Code reuse instead of code rewriting*
 - Jika task yang harus dikerjakan fungsi banyak dipakai di program, memrogram menjadi jauh lebih sederhana jika task tersebut dibuat dalam bentuk fungsi
 - Contoh: fungsi untuk menghitung akar kuadrat (**sqrt**) sangat berguna untuk berbagai jenis persoalan → bayangkan kalau setiap kali Anda harus menulis programnya 😊
- Setiap fungsi dapat dites secara mandiri dan tidak tergantung pada bagian program yang lain
 - Di Python: fungsi dapat dites dulu dalam interpreter (tidak harus membuat program utuh terlebih dahulu)
 - Jika program besar dan harus dikerjakan oleh lebih dari 1 programmer, hal ini memudahkan pembagian kerja



Tahap Penggunaan Fungsi

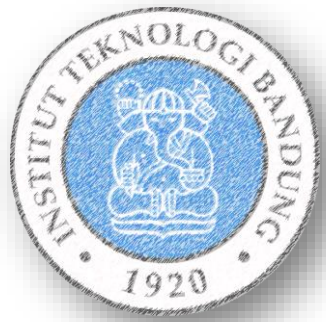
1. Mendefinisikan fungsi
 - Mendefinisikan nama, spesifikasi, domain (input), range (output)
2. Merealisasikan fungsi
 - Membuat program untuk menghasilkan output berdasarkan input
3. Menggunakan fungsi
 - Memanggil fungsi di program utama atau dalam fungsi lain

```
# Program Test
# Mengetes fungsi kuadrat
```

```
# KAMUS
#   A : integer
#   B : integer
```

```
# Fungsi Kuadrat
def Kuadrat ( X ) :
    # menghitung kuadrat X
    hasil = X * X
    return hasil
```

```
# ALGORITMA PROGRAM UTAMA
A = 5
B = Kuadrat (A) + 10
```



Mendefinisikan Fungsi

- Mendefinisikan fungsi dalam program berarti mendefinisikan di bagian blok **KAMUS**:
 - **Nama** dan **spesifikasi** fungsi
 - Himpunan nilai **domain**: type data **parameter input**
 - Himpunan nilai **range**: type data **output**
- Contoh: Fungsi kuadrat: $f(x) = x^2$
 - Nama fungsi: **kuadrat** → nama dibuat oleh programmer
 - Spesifikasi fungsi: “menghasilkan kuadrat dari x”
 - Type data input: **int**
 - Type data output: **int**

Mendefinisikan Fungsi dalam Python

- **Nama fungsi** didefinisikan setelah keyword **def**
- **Spesifikasi fungsi** dituliskan dalam bentuk komentar di bawah nama fungsi
- Type data **input** didefinisikan **implisit** berdasarkan type data **parameter_input**
 - Jika lebih dari 1, tiap parameter dipisahkan dengan koma (,)
- Type data **output** didefinisikan secara **implisit** berdasarkan type nilai_output yang dituliskan setelah perintah **return**

```
# definisi fungsi
def <nama_fungsi> ( [<parameter_input>] ):
    # spesifikasi_fungsi
    ...
    return [nilai_output]
```

Contoh fungsi Kuadrat:

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X
    ...
    return hasil
```

Merealisasikan Fungsi

- Merakit program untuk menghasilkan nilai output berdasarkan nilai input
 - Pada dasarnya dapat menggunakan segala jenis instruksi yang mungkin dalam program
- **KAMUS LOKAL**: dimungkinkan ada nama-nama variabel yang hanya terdefinisi lokal di fungsi (tidak bisa dipakai di program utama atau di fungsi yang lain)
- **ALGORITMA**: bagian program yang berisi kode program fungsi dan minimum mengandung 1 buah perintah **return**
 - **return**: perintah untuk menuliskan hasil fungsi

```
# definisi fungsi
def <nama_fungsi> ( [<parameter_input>] ):
    # spesifikasi_fungsi

    # KAMUS LOKAL
    # nama-nama variabel lokal

    # ALGORITMA
    ... # deretan instruksi untuk
        # menghasilkan nilai_output
        # berdasarkan nilai parameter_input
    return [<nilai_output>]
```

Contoh fungsi Kuadrat:

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL
    # hasil : int

    # ALGORITMA
    hasil = X * X
    return hasil
```

Contoh-1: Fungsi Kuadrat (1)

- Definisi matematika: $f(x) = x^2$
- Bagaimana memindahkannya dalam program?
 - **Nama fungsi:** Kuadrat \rightarrow ditentukan oleh programmer
 - **Spesifikasi fungsi:** menghasilkan kuadrat dari input
 - **Type domain/input:** integer, didefinisikan oleh parameter input x
 - **Type range/output:** integer \rightarrow berdasarkan type hasil x^2
 - **Realisasi fungsi:** $x * x$ atau $x ** 2$ (dalam Python)

```
# Fungsi Kuadrat

def Kuadrat ( X ):

    # Menghasilkan kuadrat dari X

    # KAMUS LOKAL
    # hasil : int

    # ALGORITMA

    hasil = X * X

    return hasil
```

Contoh-1: Fungsi Kuadrat (2)

- Alternatif: tidak perlu variabel kamus lokal → langsung ekspresi di bagian return
 - Untuk program-program yang sangat pendek, ini lebih baik

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL
    # hasil : int

    # ALGORITMA
    hasil = X * X
    return hasil
```



```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL

    # ALGORITMA

    return X * X
```

Contoh-2: Nilai maksimum 2 integer

- Buatlah fungsi yang menghasilkan nilai terbesar dari 2 buah bilangan integer.
- **Nama fungsi:** Max2
- **Spesifikasi fungsi:** Bila diketahui A dan B bilangan integer, dihasilkan bilangan terbesar antara A dan B
- **Type input:** 2 bilangan integer: A dan B
- **Type output:** bilangan integer (maksimum dari A dan B pasti juga integer)
- **Realisasi fungsi:** menggunakan if-else

```
# Fungsi Max2

def      Max2      (      A , B      ):

    # Menghasilkan bilangan terbesar
    # antara A dan B

    # KAMUS LOKAL
    # maks : int

    # ALGORITMA
    if (A >= B):
        maks = A
    else: # A < B
        maks = B

    return maks
```

Contoh-2: Nilai maksimum 2 integer (2)

- Alternatif: tidak perlu variabel kamus lokal

```
# Fungsi Max2
def Max2 ( A, B ):
    # menghasilkan nilai terbesar
    # antara A dan B

    # KAMUS LOKAL
    # maks : int

    # ALGORITMA
    if (A >= B):
        maks = A
    else: # A < B
        maks = B

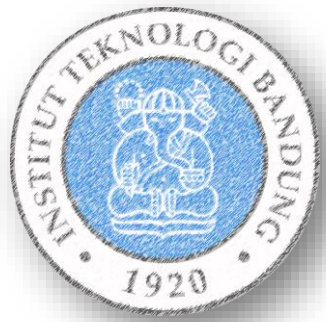
    return maks
```



```
# Fungsi Max2
def Max2 ( A, B ):
    # menghasilkan nilai terbesar
    # antara A dan B

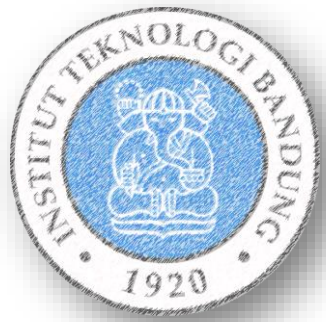
    # KAMUS LOKAL

    # ALGORITMA
    if (A >= B):
        return A
    else: # A < B
        return B
```

Latihan-1:

- Buatlah definisi dan realisasi fungsi **Max3** untuk menghitung nilai maksimum dari 3 bilangan, misalnya A, B, C.
- Contoh: $A = 1, B = -10, C = 5 \rightarrow \text{maksimum} = 5$
- Algoritma:
 - $A \geq B$ and $A \geq C$: maksimum = A
 - $B \geq A$ and $B \geq C$: maksimum = B
 - $C \geq A$ and $C \geq B$: maksimum = C



Menggunakan Fungsi (1)

- Fungsi dipanggil dalam instruksi program utama atau dalam instruksi di fungsi lain sebagai bagian dari **ekspresi**
- Syarat memanggil fungsi:
 - Nama fungsi harus sama
 - Banyaknya parameter input sama dan type data bersesuaian
 - Dalam proses pemanggilan fungsi akan terjadi asosiasi satu ke satu setiap parameter input dengan nilai masukan
 - Hasil dari pemanggilan fungsi harus dalam type yang sama dengan type output fungsi
 - Pemanggilan fungsi sebagai bagian dari ekspresi → bukan sebuah instruksi terpisah

Menggunakan Fungsi (2) - Contoh

- Nama harus sama: **Kuadrat**
- Banyaknya parameter input sama dan type data bersesuaian
 - Ada parameter input di fungsi Kuadrat yaitu x; dan ada 1 input di pemanggilan Kuadrat di program utama, yaitu y. x dan y sama-sama bertipe integer.
- Hasil dari pemanggilan fungsi harus dalam type yang sama dengan type output fungsi.
 - Perintah return di fungsi Kuadrat memberikan data bertipe integer
 - Pada pemanggilan di program utama: Kuadrat(y) akan menghasilkan integer dan tepat dengan type variabel
- Pemanggilan fungsi sebagai bagian dari ekspresi
 - Ya, Kuadrat adalah ekspresi yang ditampung hasilnya di variabel hasil

```
# Program HitungKuadrat
# Menerima masukan sebuah integer dan
# menuliskan pangkat 2 dari nilai tsb
# ke layar

# Kamus
# y, hasil : int

# Definisi Fungsi
def Kuadrat ( x ):
    # Menghasilkan pangkat 2 dari x
    # Algoritma
    return x*x

# Algoritma Program Utama
y = int(input("Masukkan bilangan = "))
hasil = Kuadrat(y)
print("Kuadrat dari "+str(y)+" = "+str(hasil))
```

Contoh-3: 10% dari bilangan terbesar

- Buatlah program yang menerima masukan 2 buah integer, misalnya A dan B. Tuliskan ke layar 10% dari nilai terbesar di antara keduanya.
- Contoh: A = 45 B = 50 terbesar = 50
 tercetak di layar = 5.0
- Untuk mencari nilai terbesar, buat dan gunakan fungsi Max2 yang telah didiskusikan di Contoh-2

Contoh-3: 10% dari bilangan terbesar

```
# Program 10persen_dari_terbesar
# Menerima masukan 2 bilangan integer
# Menuliskan ke layar 10% dari bilangan terbesar di antara keduanya

# KAMUS
# A, B : int
# hasil : float

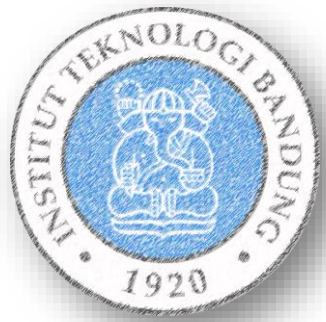
# Definisi Fungsi Max2
def Max2 (A,B):
    # Menghasilkan bilangan terbesar antara A dan B

    # KAMUS LOKAL
    if (A >= B):
        return A
    else: # A < B
        return B

# ALGORITMA PROGRAM UTAMA
A = int(input("Masukkan bilangan pertama = "))
B = int(input("Masukkan bilangan kedua = "))

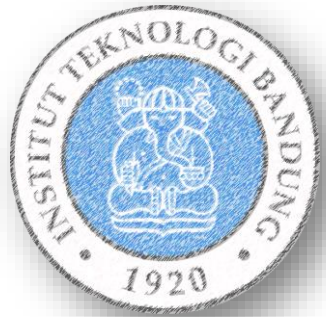
hasil = 0.1 * Max2(A,B)

print ("10% dari bilangan terbesar = " + str(hasil))
```

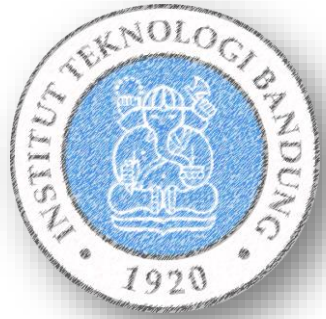


Latihan-2

- Selain dipanggil di program utama, fungsi juga bisa dipanggil di fungsi lain.
- Buatlah definisi dan realisasi **fungsi** bernama **Max3** untuk menghitung nilai maksimum dari 3 bilangan, misalnya A, B, C.
- Contoh: $A = 1, B = -10, C = 5 \rightarrow \text{maksimum} = 5$
- Realisasikan fungsi ini dengan cara membuat dan menggunakan fungsi **Max2** yang telah didiskusikan di Contoh-2.



Prosedur



Prosedur

- **Prosedur:** subprogram mengelompokkan instruksi-instruksi yang sering dipakai di program
 - Tidak harus ada parameter input/output
 - Dapat dipandang sebagai fungsi yang tidak menghasilkan (*return*) nilai
- Dalam Python, didefinisikan dengan **return** tanpa ekspresi/nilai yang dihasilkan di akhir fungsi

```
# definisi prosedur
def <nama_prosedur> ( [<parameter_input>] ):
    # spesifikasi_prosedur

    # KAMUS LOKAL
    # nama-nama variabel lokal

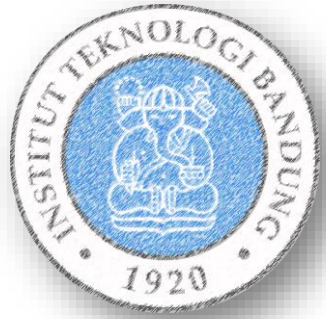
    # ALGORITMA
    ... # deretan instruksi prosedur
    return
```


Contoh-4: Hello Nama

- Buatlah fungsi **CetakNama** yang menerima masukan sebuah string nama dan mencetak “Hello, ” + nama ke layar.
- Tidak ada nilai yang dikeluarkan dari fungsi

```
# Definisi Subprogram
def CetakNama (nama):
    # Mencetak Hello + nama ke layar

    # Algoritma
    print ("Hello, " + nama + "!!")
    return
```



Memanggil Prosedur

- Karena prosedur tidak menghasilkan nilai, pemanggilannya dalam program utama atau fungsi lain juga berbeda.
- Prosedur dipanggil sebagai **1 buah baris instruksi**, bukan sebagai bagian dari ekspresi.
- Asosiasi parameter input dilakukan dengan cara yang sama seperti pada fungsi biasa

Contoh-5:

Hello Nama v2

Buatlah program yang menerima masukan sebuah integer > 0 , misalnya N , dan sebuah string, misalnya **nama** lalu mencetak: “Hello, **nama**!” sebanyak N kali ke layar

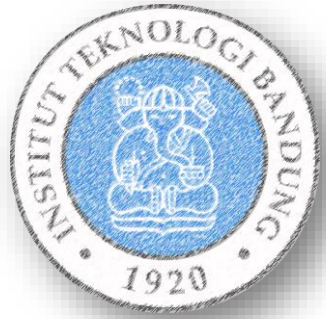
```
# Program HelloHelloNama
# Menerima masukan sebuah integer  $> 0$  N dan string nama
# dan mencetak "Hello" + nama sebanyak N kali

# Kamus
# i, N : int
# nama : string

# Definisi Prosedur CetakNama
def CetakNama (nama):
    # Mencetak Hello + nama ke layar
    # Algoritma
    print ("Hello, " + nama + "!")
    return

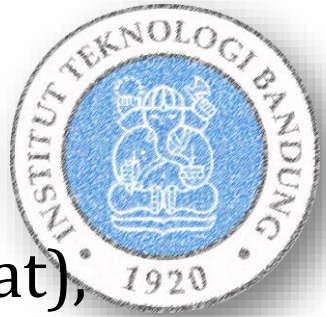
# Algoritma Program Utama
nama = input("Masukkan nama = ")
N = int(input("Berapa kali diulang? "))

for i in range(N):
    CetakNama(nama)
```



Fungsi Standar Python

- Dalam Python didefinisikan sangat banyak fungsi standar yang tersedia dan tinggal digunakan → jadi tidak perlu di-*coding* lagi
- Fungsi-fungsi standar ini didefinisikan dalam *library*
- Contoh library standar yang sering dipakai adalah **math**
- Fungsi-fungsi yang didefinisikan dalam library math:
 - **sqrt** → mencari akar kuadrat suatu bilangan
 - **sin** → mencari sinus
 - **cos** → mencari cosinus
 - **pow** → pangkat suatu bilangan
 - dll.
- Memanggil library **math** dengan menambahkan instruksi pada bagian awal program: **from math import ***
- Informasi lebih lanjut: <https://docs.python.org/3/library/math.html>



Contoh-6: Akar Kuadrat

Buatlah program yang menerima masukan sebuah bilangan riil (float), misalnya N, dan menghasilkan akar kuadrat dari bilangan tersebut

Contoh:

N = 4; $\sqrt{N} = 2.0$

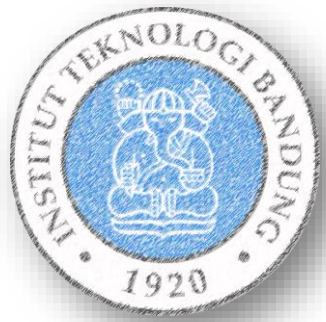
N = 12; $\sqrt{N} = 3.464...$

```
# Program HitungAkar
# Menerima masukan sebuah bilangan riil
# menuliskan akar dari bilangan tersebut

from math import *

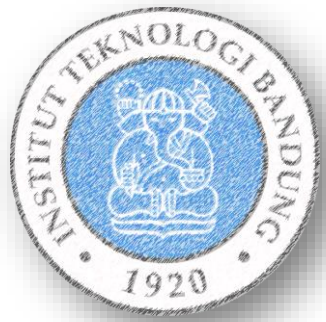
# KAMUS
# N : float

# ALGORITMA
N = float(input("Masukkan bilangan = "))
print ("Akar dari = " + str(N) + " = " + str( $\sqrt{N}$ )))
```



Latihan-3

- Buatlah sebuah fungsi bernama **HitungJarak**, yang menerima masukan: **v** : kecepatan (dalam m/s, bilangan riil) dan **t** : waktu tempuh (dalam s, bilangan riil) dan menghasilkan jarak tempuh **s** dengan rumus: $s = v * t$.
- Asumsikan nilai $t \geq 0$ dan $s \geq 0$.
- Selanjutnya, buatlah program utama yang menggunakan fungsi HitungJarak tersebut (bebas).



Latihan-4

- Masih ingat program untuk mencari nilai maksimum array?
- Buatlah fungsi **MaxArray** yang menerima masukan sebuah array of integer, misalnya T, dan panjang array, misalnya N, dan menghasilkan nilai terbesar yang disimpan dalam array tersebut. Asumsikan $N > 0$.
- Contoh: $T = [5, 4, 3, 2, 1]; N = 5$
maka nilai maksimum = 5



Latihan-5

- Salah satu *task* dalam pemrosesan array adalah mencetak semua elemen array ke layar
- Buatlah prosedur **CetakArray** yang menerima masukan sebuah array of integer, misalnya T, dengan panjang $N \geq 0$, dan mencetak semua elemen array ke layar.
- Cara mencetak: setiap elemen ke-i di cetak per baris dengan cara sbb: $[i] \text{ <elemen>}$
- Contoh: $T = [5, 4, 3, 2, 1]; N = 5$ akan tercetak:

$[0]$	5
$[1]$	4
$[2]$	3
$[3]$	2
$[4]$	1
- Jika $N = 0$, cetaklah: 'Array kosong'