

ANALISIS KASUS & PENGULANGAN - PYTHON

TIM PENYUSUN MATERI WI1102 BERPIKIR KOMPUTASIONAL
INSTITUT TEKNOLOGI BANDUNG © 2024



ANALISIS KASUS

WI1102/BERPIKIR KOMPUTASIONAL

10/13/2024

2

TUJUAN

- Mahasiswa dapat menjelaskan pengertian dan jenis-jenis analisis kasus
- Mahasiswa dapat menggunakan notasi analisis kasus dengan benar
- Mahasiswa dapat memanfaatkan jenis-jenis analisis kasus dalam menyelesaikan persoalan sederhana yang diberikan

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



Mulai

Pilih Mangga

Apakah
kuning?



ya

Matang

tidak

Tidak Matang

Selesai

MEMILIH MANGGA - PSEUDOCODE

10/13/2024

PilihMangga

if (ApakahKuning? = true) then

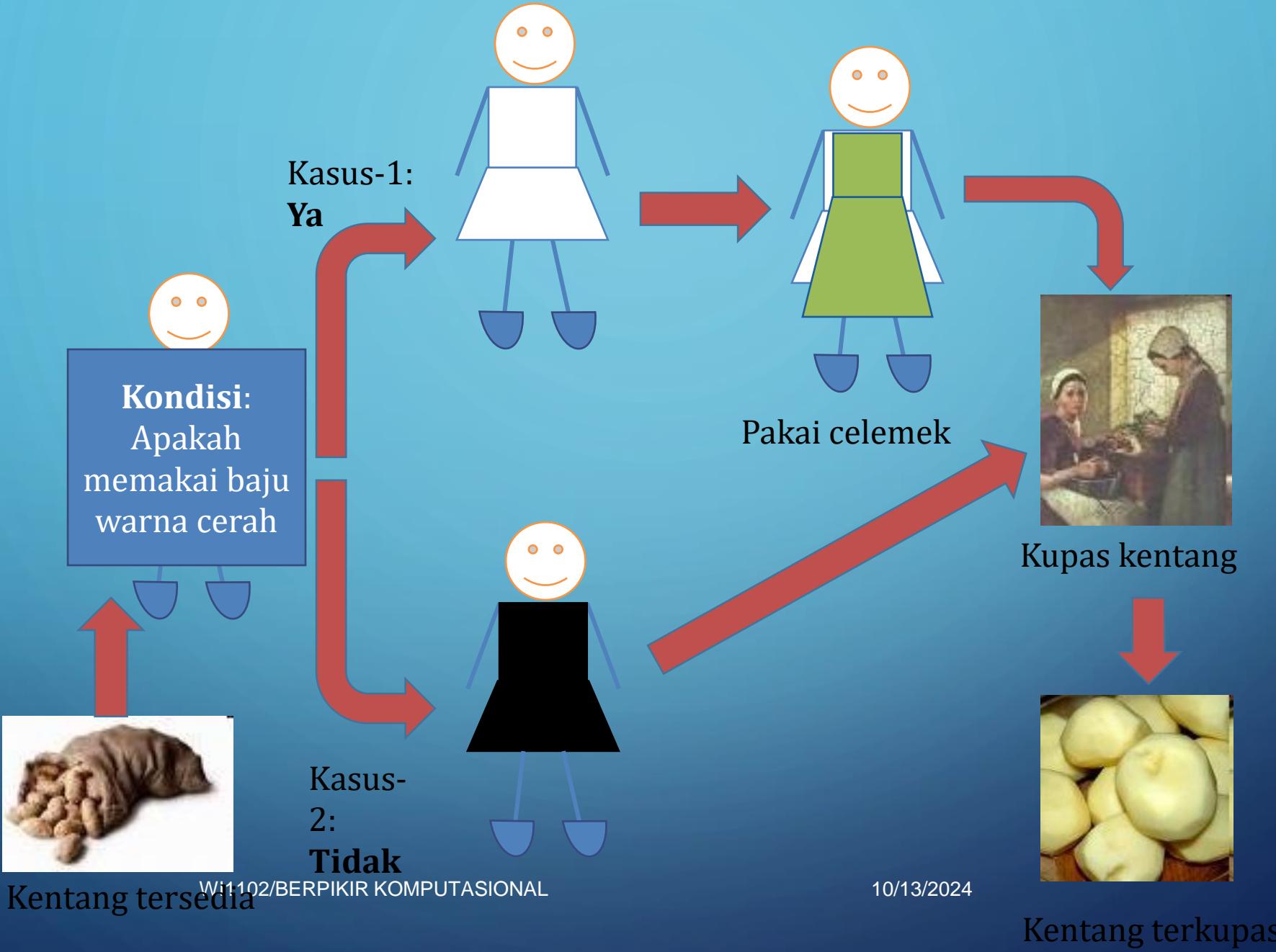
Matang

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

10/13/2024

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 didefinisikan dan ketiga kasus tersebut tidak tumpang tindih

SINTAKS UMUM

Python

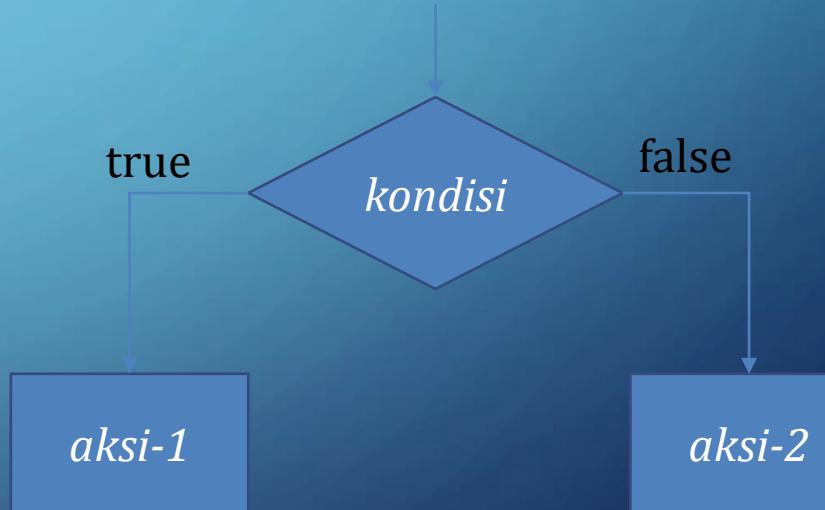
```
if ( kondisi ):  
    aksi-1  
else: # kondisi = false  
    aksi-2
```

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"

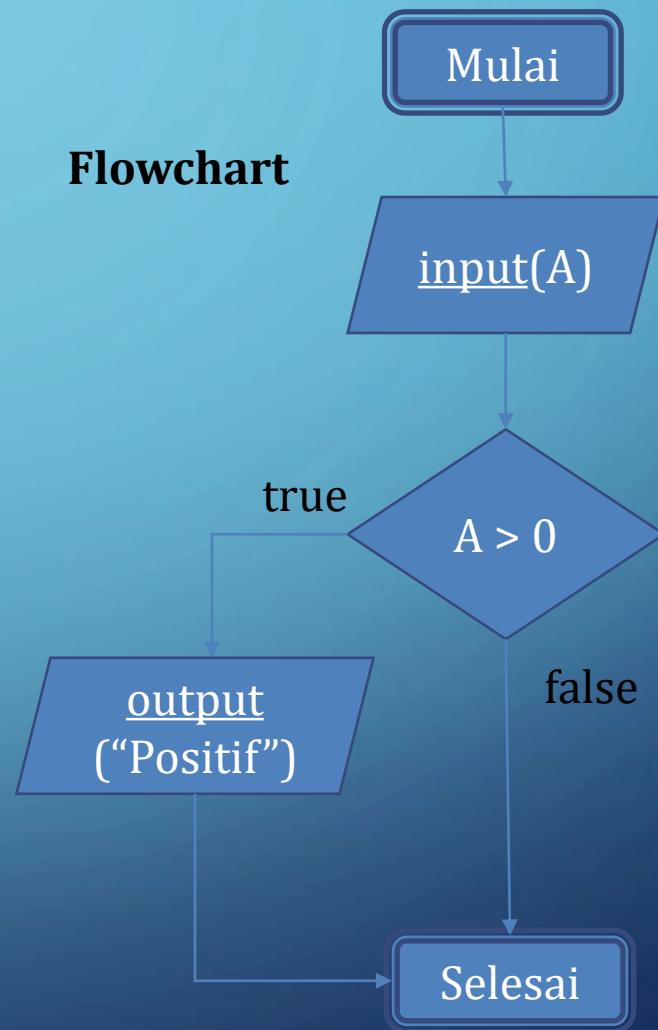
Jika **tidak**, tidak dilakukan apa pun

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”

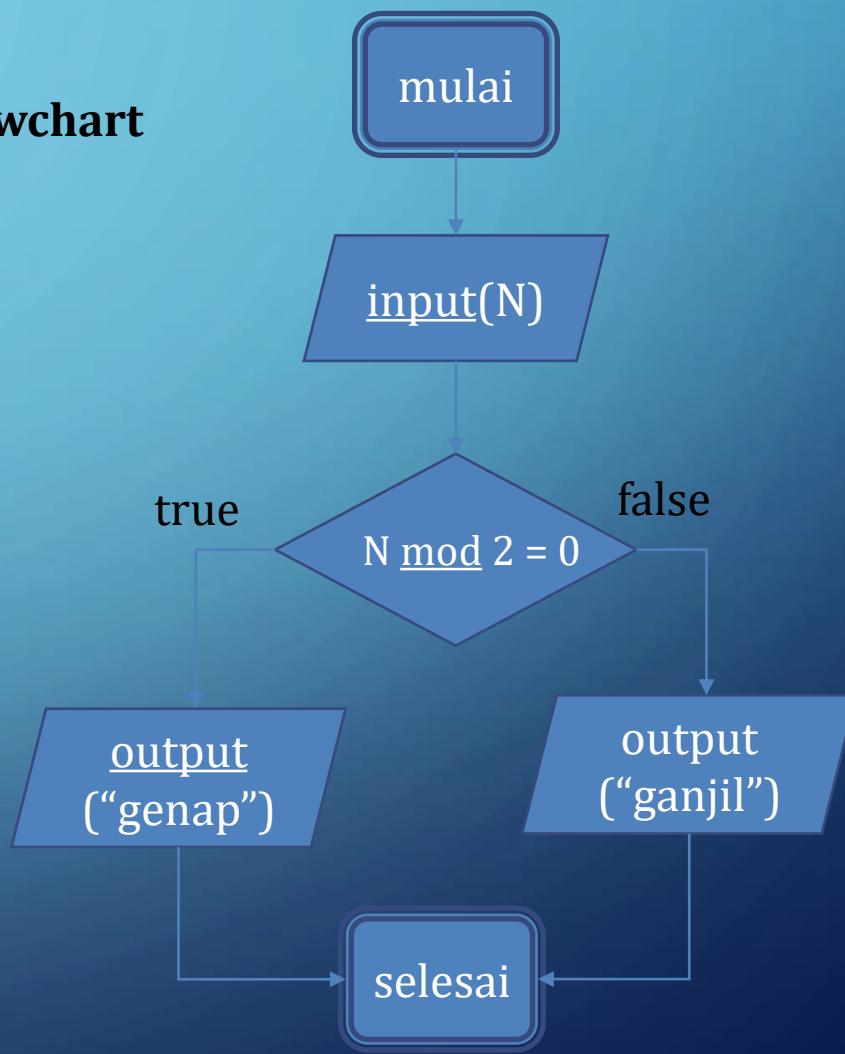
Jika tidak ($N \bmod 2 = 1$), maka cetak “ganjil”

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")
```

- 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”

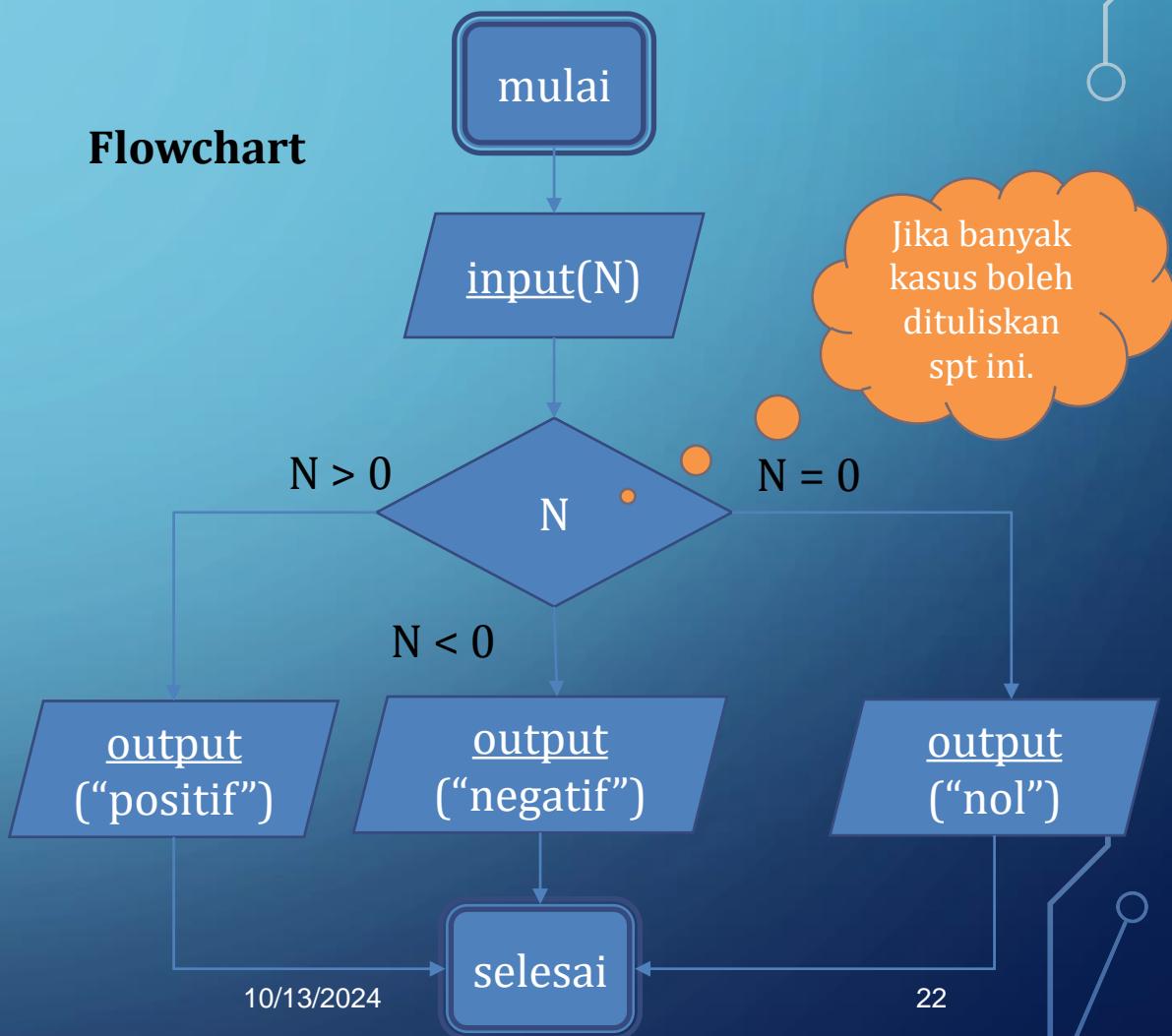
CONTOH-5: POSITIF, NEGATIF, ATAU NOL?

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")
```

PENGULANGAN

WI1102/BERPIKIR KOMPUTASIONAL

10/13/2024

24

TUJUAN

01

Mahasiswa dapat menjelaskan jenis-jenis pengulangan dan penggunaannya serta elemen-elemen dalam pengulangan.

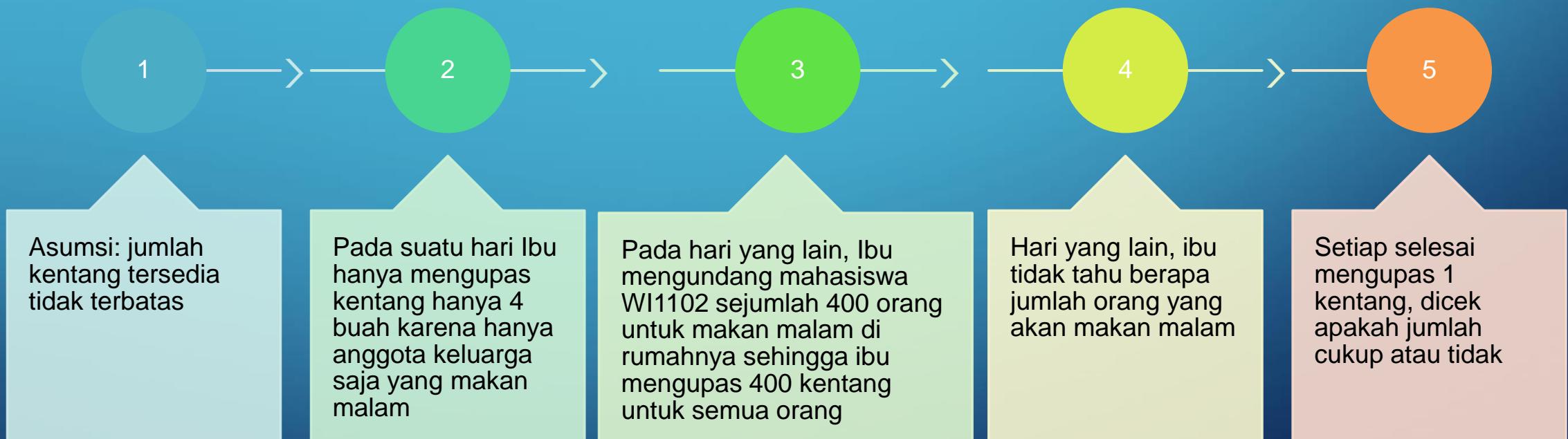
02

Mahasiswa dapat menggunakan notasi pengulangan yang sesuai dengan benar.

03

Mahasiswa dapat memanfaatkan jenis-jenis pengulangan dengan tepat dalam menyelesaikan persoalan sederhana yang diberikan.

MENYIAPKAN KENTANG UNTUK MAKAN MALAM





Jumlah kentang cukup?

Jumlah kentang = 4?

Jumlah kentang = 400?

Jumlah kentang = jumlah orang yang hadir?



Kupas 1 kentang



Jumlah kentang terkupas cukup

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

KOMPUTASIONAL

```
# 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!!
```

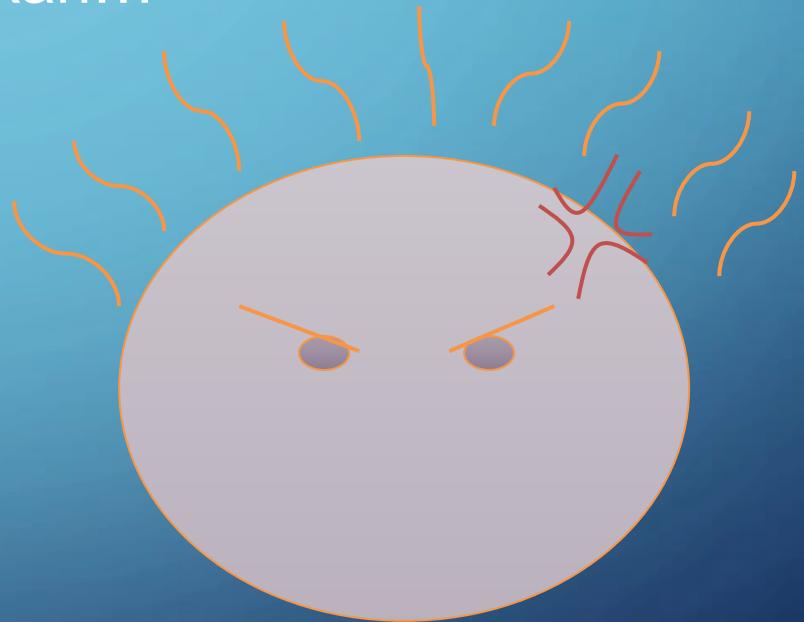
```
# ALGORITMA  
print(1)  
print(2)  
print(3)  
print(4)  
print(5)  
print(6)  
... #lanjutkan sendiri!!  
print(100)  
print(1+2+3+4+5+6+7+8+9+10+ ... #lanjutkan sendiri!!)
```



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 lojik
- **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

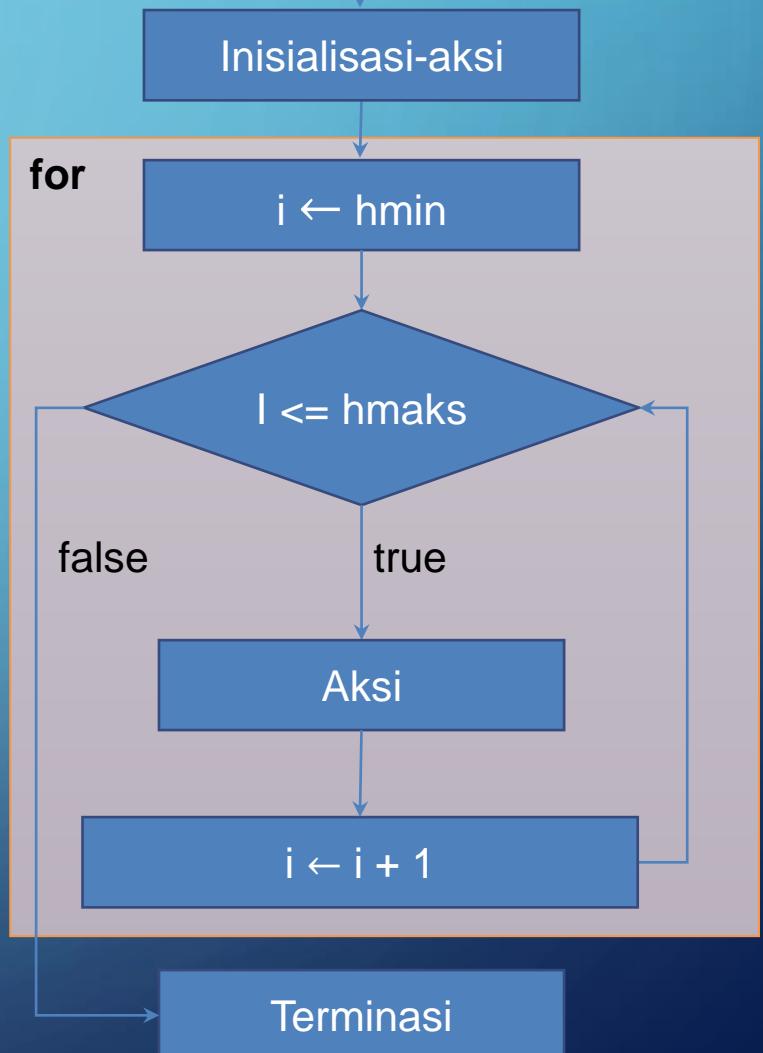
Inisialisasi-aksi

```
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

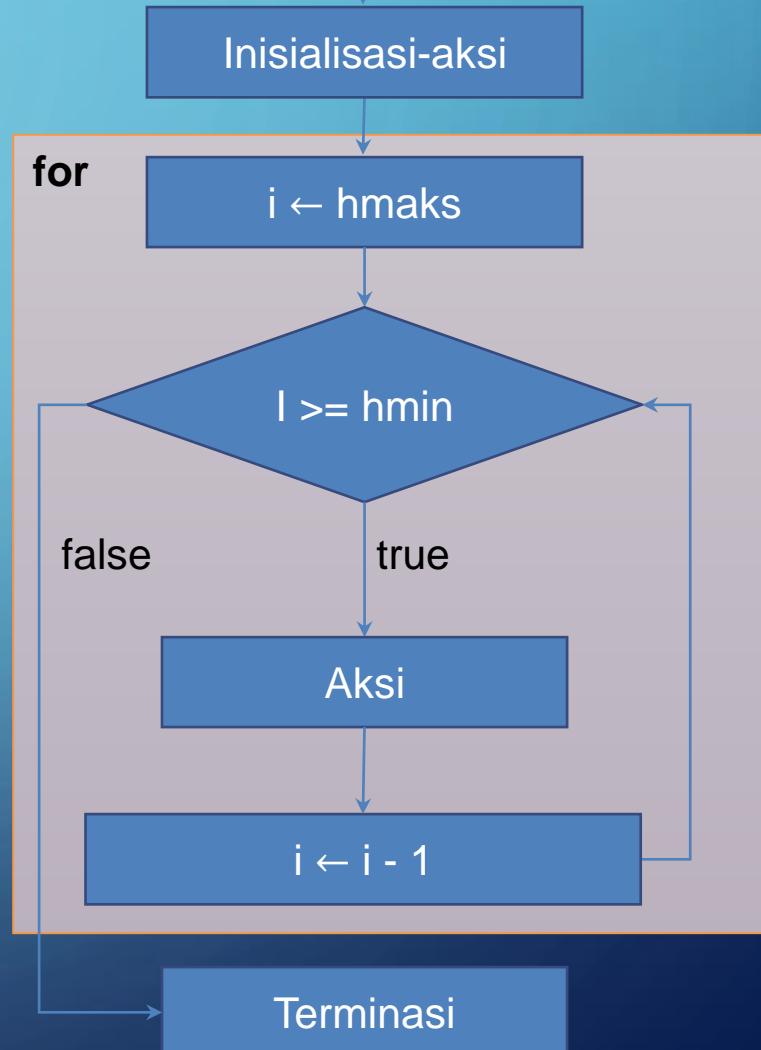
i adalah variabel pencacah (bisa diganti variabel lain)
hmaks = nilai i di awal *loop*; *hmin* = nilai i terakhir yang diproses;
nilai i ketika keluar *loop* adalah *hmin*-1
-1: Setiap berulang, i di-decrement (dikurangi 1)

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)

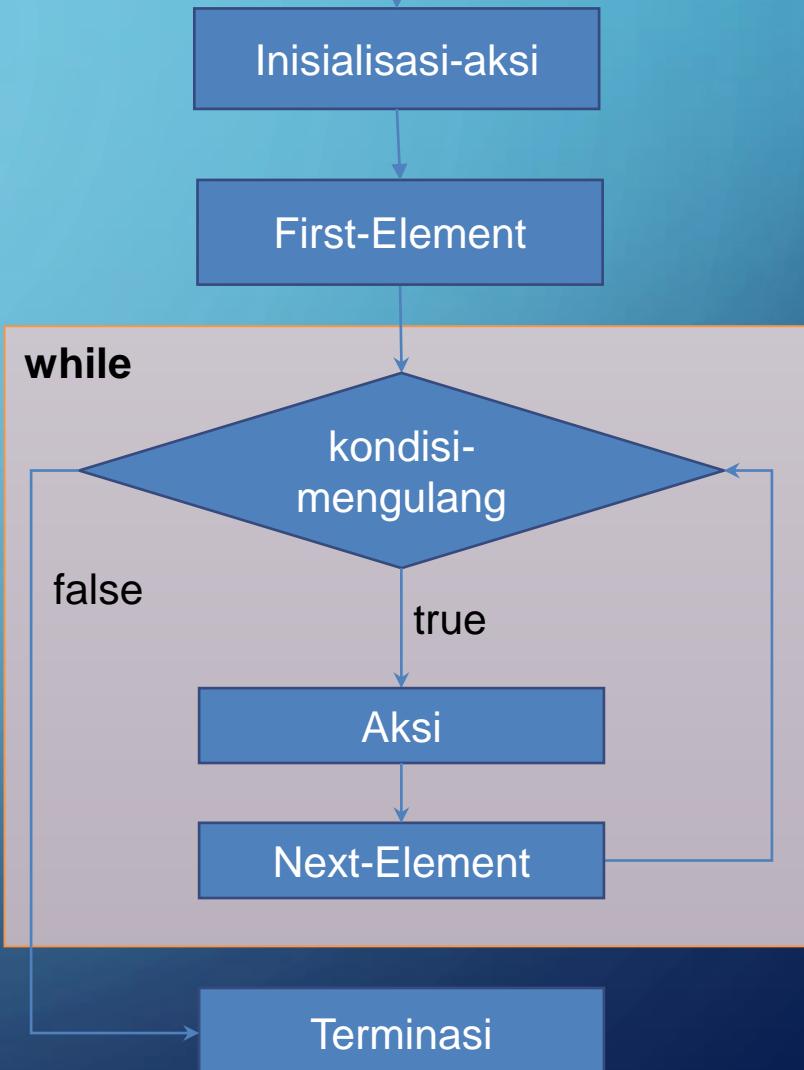
Python

```
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:

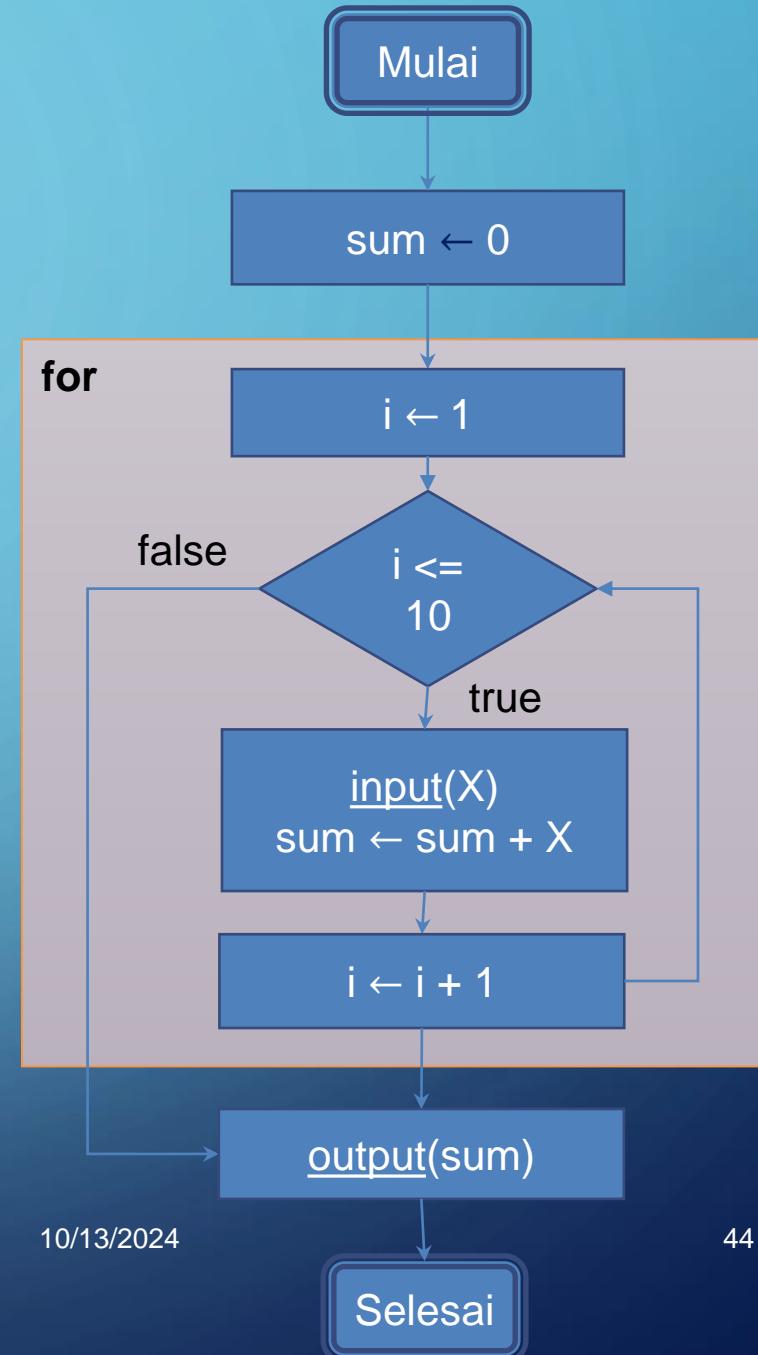
Masukan	Tampilan di Layar
2	18
1	
0	
-9	
7	
13	
2	
2	
1	
-1	

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>

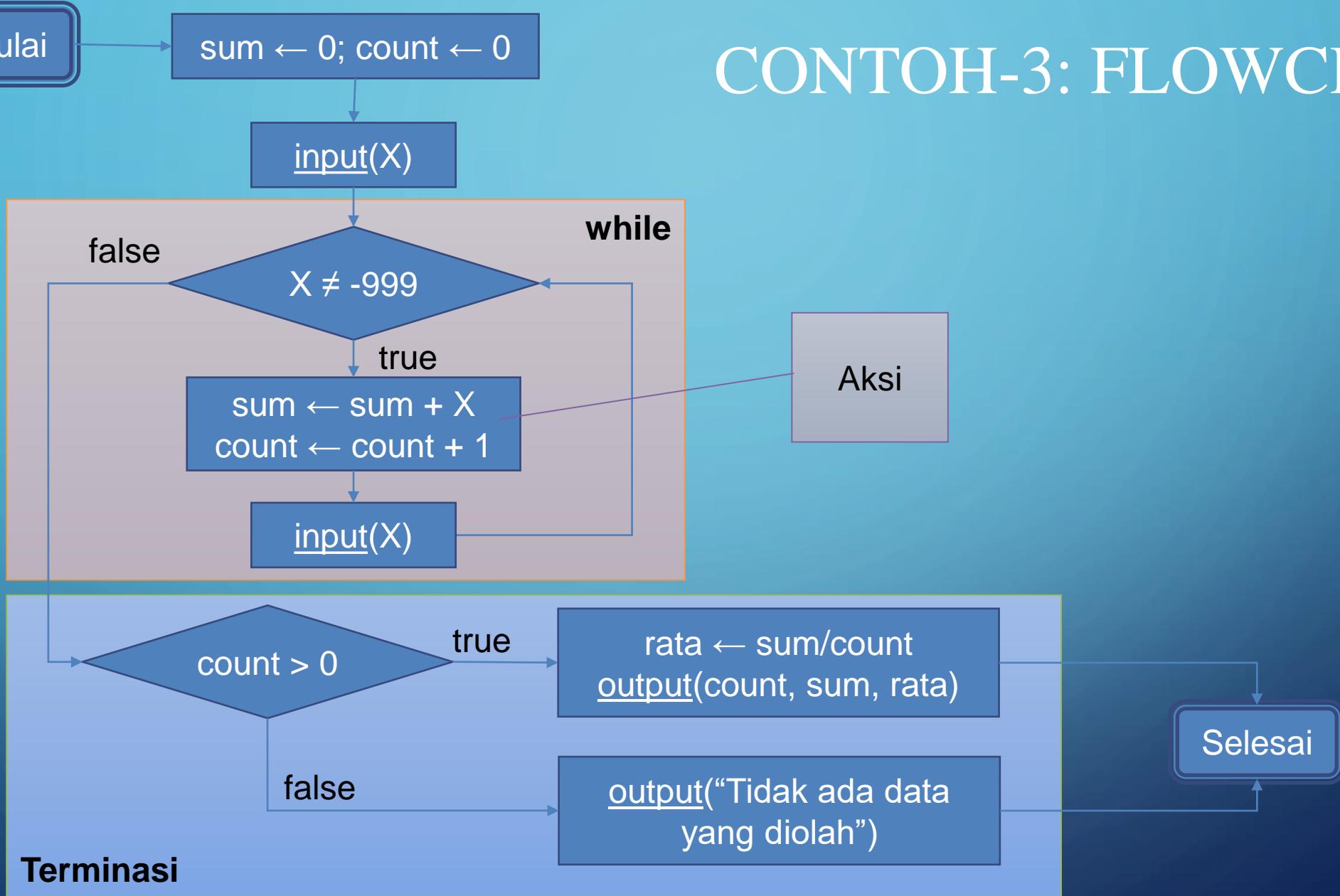
CONTOH-3: PSEUDOCODE

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

input(X) { First-Elmt }
{ Proses pengulangan dengan while-do }
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 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:

For tidak tepat digunakan karena tidak terdefinisi *range* nilainya

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)