

MODUL MATA KULIAH

BAHASA PEMROGRAMAN DASAR

PG168 – 3 SKS



**FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR**

**JAKARTA
SEPTEMBER 2021**

TIM Penyusun

Agus Umar Hamdani, M.Kom
Tri Ika Jaya Kusumawati, M.Kom¹



MODUL PERKULIAHAN #7

KONTROL PERULANGAN

Capaian Pembelajaran	:	Mahasiswa Mampu: <ol style="list-style-type: none">1. Memahami bentuk umum struktur kontrol perulangan <i>For Loop</i> dan <i>While Loop</i>2. Memahami penggunaan <i>control statements</i> (<i>break</i>, <i>continue</i>, <i>pass</i>).
Sub Pokok Bahasan	:	<ol style="list-style-type: none">1. Kontrol perulangan : <i>For Loop</i> dan <i>While Loop</i>2. <i>Control Statements</i> (<i>break</i>, <i>continue</i>, <i>pass</i>).
Daftar Pustaka	:	<ol style="list-style-type: none">1. Zarman, Wendi dan Wicaksono, Mochamad Fajar. "<i>Implementasi Algoritma dalam bahasa Python</i>". Edisi Pertama. Bandung : Penerbit Informatika. 2020.2. Kurniawati, Arik. "Algoritma dan Pemrograman menggunakan Python". Edisi Pertama. Yogyakarta : Depublish. 2016.3. Ismah. "Pemrograman Komputer Dasar-dasar Python". Jakarta : Fakultas Ilmu Pendidikan Universitas Muhammadiyah Jakarta. 2017.4. Irfani, M. Haviz dan Dafid. "Modul Praktikum Dasar Pemrograman dengan bahasa Python". Palembang : Sekolah Tinggi Manajemen Informatika Global Informatika Multidata Palembang. 2016.5. Fikri, Rijalul. "Praktikum Algoritma dan Pemrograman Komputer". Surabaya : Program Studi Teknik Komputer dan Telematika Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember. 2010.6. Wiratmaja, I Gede Harjumawan, et.all. 2021. Program Menghitung Banyak Bata pada Ruangan Menggunakan Bahasa Python. TIERS Information Technology Journal. Vol. 2(1). Undiknas.7. Nuraini, Rini. 2017. Desk Check Table Pada Flowchart Operasi Perkalian Matriks. Jurnal Petir. Vol. 10(1). Sekolah Tinggi Teknik – PLN (STT-PLN).8. Romzi, Muhammad dan Kurniawan, Budi. 2020. Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma. JTIM : Jurnal Teknik Informatika Mahakarya. Vol. 03(2). Hal. 37-44.9. Programiz.com. Python Operators (https://www.programiz.com/python-programming/operators diakses pada 29 September 2021 pukul 21.32 WIB)

PRAKTIKUM 7

STRUKTUR KONTROL PERULANGAN

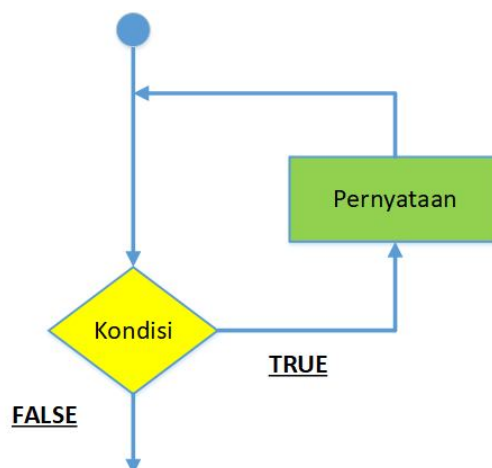
7.1 Teori Singkat

Kontrol Perulangan yang biasa disebut dengan "*looping*" adalah proses melakukan tindakan yang sama secara berulang-ulang atau berkali-kali sampai batas yang telah ditentukan [6]. Struktur kontrol perulangan digunakan untuk mengulangi sejumlah aksi yang sama sebanyak jumlah yang ditentukan atau kondisi yang diinginkan untuk menyelesaikan masalah tertentu [8].

Beberapa karakteristik struktur kontrol perulangan adalah:

- a. Mengerjakan hal yang sama berulang-ulang.
- b. Jumlah perulangan bisa ditetapkan, dapat juga sesuai kondisi.
- c. Ada kondisi awal dan kondisi akhir. Perulangan dimulai dari kondisi awal, naik secara bertahap dan berhenti saat mencapai kondisi akhir.
- d. Menaikkan dan menurunkan kondisi perulangan dilakukan dengan operator *increment* (menambah nilai variabel sebanyak satu angka), *decrement* (mengurang nilai variabel sebanyak satu angka), dan ekspresi matematika.

Struktur perulangan (loop) dapat dilihat pada gambar 7.1.



Gambar 7.1 Struktur Kontrol Perulangan (*loop*)

Adapun komponen-komponen dalam struktur kontrol perulangan terdiri dari :

- a. Kondisi perulangan : setiap perintah atau kumpulan perintah yang dikerjakan, jika memenuhi kondisi tertentu. Selama kondisi terpenuhi, perintah tersebut akan terus dikerjakan.
- b. Badan perulangan : kumpulan perintah yang hendak diulang.
- c. Nilai awal atau inisiasi : pemberian nilai satu atau beberapa variabel sebelum pengulangan dilakukan.
- d. Perubahan variabel kontrol (*updating*) : variabel yang mengontrol berapa kali perintah harus diulang dan mencegah perulangan berlangsung selama tak hingga kali (terus-menerus).

Terdapat 3 (tipe) kontrol perulangan (*loop*) dalam bahasa Python, selengkapnya dapat dilihat pada tabel 7.1.

Tabel 7.1 Kontrol perulangan alam bahasa Python

<i>Control Statements</i>	Deskripsi
<i>While Loop</i>	Mengulangi pernyataan atau sekelompok pernyataan saat kondisi yang diberikan bernilai TRUE. Ini menguji kondisi terlebih dulu sebelum mengeksekusi badan loop.
<i>For Loop</i>	Mengeksekusi urutan pernyataan beberapa kali dan menyingkat kode yang mengelola variabel loop.
<i>Nested Loop</i>	Menggunakan satu atau lebih loop di dalam perulangan lain.

Terdapat tiga perintah untuk mengontrol perulangan (*loop*), selengkapnya dapat dilihat pada tabel 7.2.

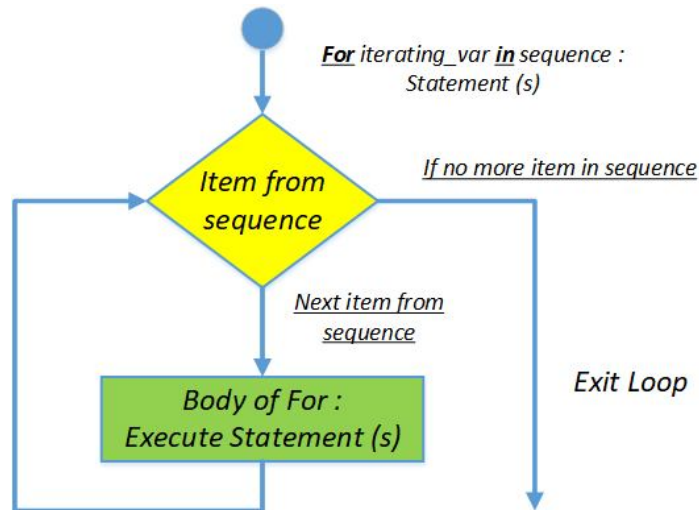
Tabel 7.2 Perintah Mengontrol Perulangan (*loop*)

Perintah	Deskripsi
<i>Break</i>	Mengakhiri pernyataan loop dan mentransfer eksekusi ke pernyataan segera setelah loop.
<i>Continue</i>	Menyebabkan loop melewati semua pernyataan yang terisita dan segera memindahkan kontrol kembali ke atas loop.
<i>Pass</i>	Pernyataan <i>pass</i> dalam Python digunakan ketika pernyataan diperlukan secara penulisan (sintaksis), tetapi tidak ingin perintah atau pernyataan tersebut dijalankan.

Operator logika dan perbandingan dapat digunakan dalam kontrol perulangan For Loop, While Loop dan Nested Loop.

7.2 Kontrol FOR LOOP

Struktur perulangan For Loop digunakan untuk melakukan proses perulangan yang frekuensinya atau jumlah perulangan telah diketahui sebelum proses perulangan dimulai [7]. Perulangan For Loop ini bekerja lebih sebagai metode iterator seperti yang ditemukan pada bahasa pemrograman berorientasi obyek. Dengan perulangan For Loop, kita dapat mengeksekusi satu set pernyataan untuk setiap item dalam tipe List, Tuple, Dictionary, Set maupun String. Struktur kontrol For Loop dalam bahasa Python dapat dilihat pada gambar 7.2.



Gambar 7.2 Struktur Perulangan For Loop

Sintak penulisan struktur kontrol For Loop di bahasa Python dapat dilihat pada gambar 7.3.

```

for var in range(batas_awal, batas_akhir, step) :
    pernyataan yang diulang
  
```

Gambar 7.3 Struktur Kontrol For

Fungsi Range() pada bahasa pemrograman Python digunakan untuk menentukan batas awal perulangan, batas akhir dan step perulangan yang akan dilakukan.

Berikut ini beberapa hal yang perlu diketahui dalam perulangan For Loop:

- Var**, **batas_awal** dan **batas_akhir** harus bertipe bilangan bulat (Integer).
- Pada perulangan For secara naik, maka **batas_awal** \leq **batas_akhir**, sedangkan untuk perulangan For secara turun, maka **batas_awal** \geq **batas_akhir**.
- Pernyataan yang diulang** merupakan badan perulangan.
- Nilai awal perulangan adalah **batas_awal**.
- Banyaknya perulangan dibatasi oleh nilai **batas_akhir**. Program berhenti di nilai **batas_akhir + 1**.

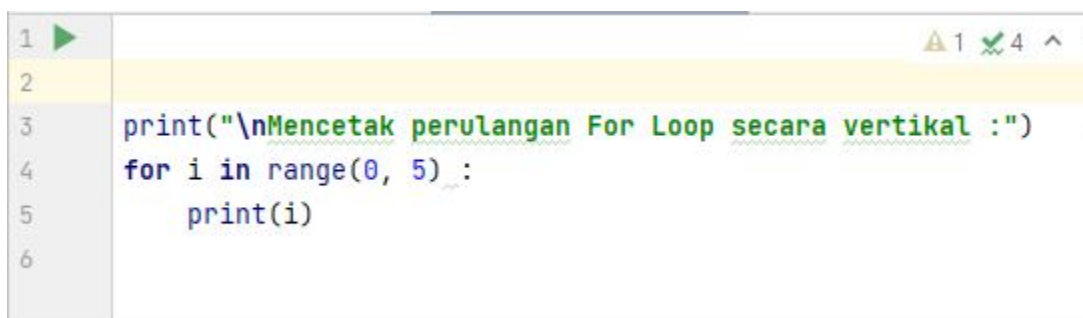
- f. **Variabel pencacah (var)** bertambah atau berkurang sebanyak satu secara otomatis.
- g. Untuk **perulangan For menaik**, berikan **nilai positif untuk step**. Sedangkan untuk **perulangan For menurun**, berikan **nilai negatif untuk step**.
- h. Jika **nilai batas_awal** dan **step tidak diberikan**, maka secara otomatis **perulangan** akan **dimulai dari 0** dengan **step 1**.

Proses Perulangan :

- a. Nilai yang terkandung pada **var** mula-mula sama dengan **batas_awal**.
- b. Kemudian perintah pada badan perulangan dikerjakan.
- c. Kemudian nilai **var** bertambah sebanyak satu **Increment(var)** atau bertambah sesuai nilai step yang diberikan untuk perulangan **For menaik** atau berkurang sebanyak satu **Decrement(var)** atau berkurang sesuai nilai step yang diberikan untuk perulangan **For menurun** secara otomatis.
- d. Proses **perulangan berhenti** secara otomatis di saat **var** mencapai nilai **batas_akhir**.

Studi Kasus 7.1

Program pada gambar 7.4 merupakan contoh penerapan struktur kontrol For Loop untuk mencetak perulangan nilai yang ada pada variabel *i* dengan **batas_awal** dan **batas_akhir** yang diinput melalui keyboard.

A screenshot of a Python code editor interface. On the left, there is a vertical line of numbers 1 through 6. To the right of these numbers is the code:

```
print("\nMencetak perulangan For Loop secara vertikal :")  
for i in range(0, 5) :  
    print(i)
```

 The code is color-coded: `print` is blue, `"\nMencetak perulangan For Loop secara vertikal :")` is green, `for` is blue, `i` is blue, `in` is blue, `range(0, 5)` is green, `:` is blue, and `print(i)` is blue. In the top right corner of the editor, there are icons for a warning (triangle), a checkmark, and a close button, along with the number 1.

Gambar 7.4 Contoh penerapan kontrol For Loop

Perhatikan gambar 7.4, Nilai batas_awal diisi dengan angka 0, sedangkan nilai batas_akhir diisi dengan angka 5. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 7.5.

```
Mencetak perulangan For Loop secara vertikal :  
0  
1  
2  
3  
4
```

Gambar 7.5 Hasil Keluaran Program

Mula-mula variabel i bernilai 1, kemudian perintah print(i) menghasilkan keluaran 1 disertai dengan enter. Proses yang sama kemudian diulang lagi dengan i berikutnya ditambah satu menjadi 1. Kemudian Perintah print(i) menghasilkan keluaran 1. Demikian seterusnya. program akan berhenti saat perulangan mencapai batas_akhir-1.

Studi Kasus 7.2

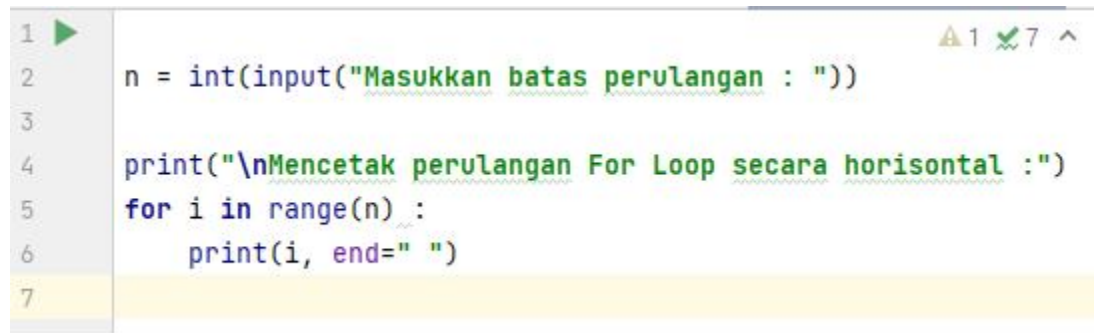
Program pada gambar 7.6 merupakan contoh penerapan struktur kontrol For Loop untuk mencetak perulangan nilai yang ada pada variabel i dengan batas perulangan n yang diinput melalui keyboard.

```
1 ▶  
2 n = int(input("Masukkan batas perulangan : "))  
3  
4 print("\nMencetak perulangan For Loop secara horisontal :")  
5 for i in range(n) :  
6     print(i, end=" ")  
7
```

Gambar 7.6 Contoh penerapan kontrol For Loop

Perhatikan gambar 7.6, program akan menunggu kita untuk memasukkan nilai untuk batas perulangan. Nilai batas perulangan yang diinput akan dikonversi ke dalam tipe Integer dan disimpan ke dalam n. Setelah kode program diatas

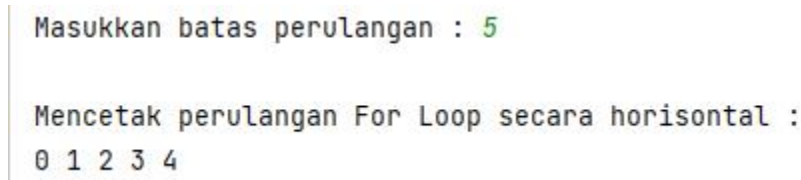
dijalankan, kemudian batas perulangan diberikan dengan nilai 5, maka hasil keluaran yang ditampilkan pada gambar 7.7.



```
1  ▶
2  n = int(input("Masukkan batas perulangan : "))
3
4  print("\nMencetak perulangan For Loop secara horisontal :")
5  for i in range(n) :
6      print(i, end=" ")
7
```

Gambar 7.7 Contoh Penggunaan Kontrol For Loop

Mula-mula variabel i bernilai 1, kemudian perintah print(i) menghasilkan keluaran 1. Proses yang sama kemudian diulang lagi dengan i berikutnya ditambah satu menjadi 2. Kemudian Perintah print(i) menghasilkan keluaran 2. Demikian seterusnya. program akan berhenti saat perulangan mencapai nilai n. Adapun hasil keluaran dari program diatas ditampilkan sebagai berikut :



```
Masukkan batas perulangan : 5

Mencetak perulangan For Loop secara horisontal :
0 1 2 3 4
```

Gambar 7.8 Hasil Keluaran Program

Studi Kasus 7.3

Program pada gambar 7.9 merupakan contoh penerapan struktur kontrol For Loop untuk mencetak perulangan nilai yang ada pada variabel i dengan batas perulangan yang diinput melalui keyboard.

```
1  ▶ print("Program deret aritmatika")
2
3  print("=====")
4
5  sukuAwal = input("Masukkan suku awal : ")
6  beda = input("Masukkan beda : ")
7  banyak = int(input("Masukkan banyak deret : "))
8
9  jumDeret = banyak + 1
10 suku = int(sukuAwal)
11 jum = 0
12
13 for i in range(1, jumDeret, 1) :
14     print("Suku ke-", i, "=", suku)
15     suku = suku + int(beda)
16     jum = jum + 1
17
18 print("Jumlah deret = ", jum)
19
```

Gambar 7.9 Contoh penerapan kontrol For Loop

Perhatikan gambar 7.9, program akan menunggu kita untuk memasukkan nilai suku awal, beda dan banyak deret. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Suku awal : 3
- Beda : 2
- Banyak deret : 5

, maka hasil keluaran yang ditampilkan pada gambar 7.10.

```

Program deret aritmatika
=====
Masukkan suku awal : 3
Masukkan beda : 2
Masukkan banyak deret : 5
Suku ke- 1 = 3
Suku ke- 2 = 5
Suku ke- 3 = 7
Suku ke- 4 = 9
Suku ke- 5 = 11
Jumlah deret = 5

```

Gambar 7.10 Contoh Penggunaan Kontrol For Loop

Mula-mula variabel *i* bernilai 1, kemudian perintah `print()` akan mencetak nilai *i* dan nilai suku sehingga menghasilkan keluaran untuk nilai *i* adalah 1 dan nilai suku adalah nilai 3, dimana nilai awal (1) ditambah dengan beda (2). Proses yang sama kemudian diulang lagi dengan *i* berikutnya ditambah satu menjadi 2. Kemudian Perintah `print()` mencetak nilai *i* adalah 2 dan nilai suku adalah 5, dimana nilai suku (3) ditambah beda (2). Demikian seterusnya. program akan berhenti saat perulangan mencapai nilai `jumlahderet`. setelah itu nilai dari variabel *i*, nilai dari variabel suku dan variabel `jumlah` ditampilkan ke layar.

Studi Kasus 7.4

Program pada gambar 7.11 merupakan contoh penerapan struktur kontrol For Loop untuk mencetak perulangan nilai yang ada pada variabel *i* dengan batas perulangan yang diinput melalui keyboard.

```

1  ▶
2  for i in range(10):
3      if i % 3 == 0 and i % 5 == 0:
4          print("Nilai i : ", i, ': Mendesis dan mendengung')
5      elif i % 3 == 0:
6          print("Nilai i : ", i, ': Mendesis')
7      elif i % 5 == 0:
8          print("Nilai i : ", i, ': Mendengung')
9      else:
10         print(i, '-')
11

```

Gambar 7.11 Contoh Penggunaan Kontrol For Loop dengan Kontrol IF dan Operator AND

Perhatikan gambar 7.11, program akan memeriksa nilai *i* sebanyak batas akhir yang bernilai 10, dimana nilai awal untuk variabel *i* adalah 0. Terdapat tiga kondisi percabangan IF yang dibuat, yaitu `if i%3 == 0 and i%5 == 0`, `elif i%3 == 0` dan `elif i%5 == 0`. Jika kondisi pemeriksaan bernilai benar (TRUE), maka nilai *i* akan dicetak beserta keterangannya. Jika tidak memenuhi ketiga kondisi tersebut, maka akan masuk ke bagian `else`. Setelah kode program diatas dijalankan, maka hasil keluaran yang ditampilkan pada gambar 7.12.

```

Nilai i : 0 : Mendesis dan mendengung
1 -
2 -
Nilai i : 3 : Mendesis
4 -
Nilai i : 5 : Mendengung
Nilai i : 6 : Mendesis
7 -
8 -
Nilai i : 9 : Mendesis

```

Gambar 7.12 Hasil Keluaran Program

Studi Kasus 7.5

Kontrol For Loop pada tipe List

Program pada gambar 7.13 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe List untuk mencetak perulangan nilai yang ada pada .

```
1  ▶ #deklarasi variabel dengan tipe list
2  number_list = []
3  n = int(input("Input ukuran list : "))
4  jum = 0
5  print("\n")
6  for i in range(0, n):
7      print("Input nilai pada indeks ke-: ", i, )
8      list_item = int(input())
9      jum = jum + list_item
10 #menambahkan data ke variabel list
11 number_list.append(list_item)
12 print("Data List : ", number_list)
13 print("Jumlah : ", jum)
14
```

Contoh 7.13 Contoh Penggunaan Kontrol Loop pada tipe List

Perhatikan gambar 7.13, terdapat deklarasi variabel `number_list` yang memiliki tipe List untuk menyimpan nilai yang diinput oleh pengguna. Program akan menunggu untuk memasukkan ukuran list yang bertipe Integer dan disimpan dalam variabel `n`. Kemudian mula-mula variabel `i` akan bernilai 0 (karena batas_bawah diberi nilai 0) dan akan diulang hingga mencapai nilai `n`. Jika kondisi pemeriksaan nilai `i` bernilai TRUE, maka anda akan diminta menginput nilai indeks melalui keyboard. Setelah itu, nilai yang diinput tersebut akan dimasukkan ke dalam variabel `number_list` setelah melalui proses konversi dari tipe string ke tipe Integer. Jika kondisi pemeriksaan nilai `i` bernilai salah (FALSE), maka program akan mencetak nilai `number_list` dan nilai `jum`. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Input ukuran List : 5
- Input nilai pada indeks : 1, 3, 5, 7, 9

, maka hasil keluaran yang ditampilkan pada gambar 7.14.

```
Input ukuran list : 5

Input nilai pada indeks ke-: 0
1
Input nilai pada indeks ke-: 1
3
Input nilai pada indeks ke-: 2
5
Input nilai pada indeks ke-: 3
7
Input nilai pada indeks ke-: 4
9
Data List : [1, 3, 5, 7, 9]
Jumlah : 25
```

Gambar 7.14 Hasil Keluaran Program

Studi Kasus 7.6

Penggunaan Kontrol For Loop menggunakan tipe Tuple

Kita dapat membuat sebuah tipe Tuple melalui tahapan berikut :

1. Membuat sebuah List.
2. Menginput elemen List.
3. Mengkonversi tipe List ke tipe Tuple.

Program pada gambar 7.15 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe Tuple untuk mencetak perulangan nilai pada Tuple tersebut.

```

1  ► #deklarasi variabel dengan tipe list
2    listA = []
3    n = int(input("Input ukuran list : "))
4    jum = 0
5    #print("\n")
6    for i in range(0, n):
7        print("Input nilai pada indeks ke-: ", i, )
8        list_item = int(input())
9        listA.append(list_item)
10
11   #konversi tipe List menjadi tipe Tuple
12   tupleA = tuple(listA)
13   jum = sum(tupleA)
14   print("Data Tuple A : ", tupleA)
15   print("Jumlah item pada Tuple A: ", jum)
16

```

Gambar 7.15 Contoh Penggunaan Kontrol For Loop dengan Tipe Tuple

Perhatikan gambar 7.15, terdapat deklarasi variabel listA yang memiliki tipe List. Program akan menunggu untuk memasukkan jumlah ukuran list yang bertipe Integer dan disimpan dalam variabel n. variabel jum digunakan untuk menghitung total nilai yang diinput. Kemudian mula-mulai variabel i akan bernilai 0 (karena batas_bawah diberi nilai 0) dan akan diulang hingga mencapai nilai n. Jika kondisi pemeriksaan nilai i bernilai TRUE, maka pengguna akan diminta menginput nilai melalui keyboard. Nilai tersebut akan dikonversi ke tipe Integer dan dimasukkan ke dalam variabel list_item. Kemudian semua nilai yang ada di variabel list_item dimasukkan ke variabel listA menggunakan perintah Append(). Jika kondisi pemeriksaan nilai i bernilai salah (FALSE), maka program akan mengkonversi nilai pada ListA menjadi bertipe Tuple dan disimpan ke dalam variabel TupleA. Kemudian menghitung total nilai yang ada di tupleA dan menyimpannya ke variabel jum. Kemudian program akan mencetak nilai tupleA dan nilai jum. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Input ukuran List : 5
- Input nilai pada indeks ke-0 : 1
- Input nilai pada indeks ke-1 : 2

- Input nilai pada indeks ke-2 : 3
- Input nilai pada indeks ke-3 : 4
- Input nilai pada indeks ke-4 : 5,

maka hasil keluaran yang ditampilkan pada gambar 7.16.

```

Input ukuran list : 5
Input nilai pada indeks ke-: 0
1
Input nilai pada indeks ke-: 1
2
Input nilai pada indeks ke-: 2
3
Input nilai pada indeks ke-: 3
4
Input nilai pada indeks ke-: 4
5
Data Tuple A : (1, 2, 3, 4, 5)
Jumlah item pada Tuple A: 15

```

Gambar 7.16 Hasil Keluaran Program

Studi Kasus 7.7

Penggunaan Kontrol For Loop menggunakan tipe Dictionary

Kita dapat membuat sebuah tipe Dictionary melalui cara berikut :

1. Perulangan menggunakan For Loop dan Indexing.
2. Perulangan menggunakan fungsi Key() dan Indexing.
3. Perulangan menggunakan fungsi Items().
4. Menyeleksi pasangan key-value untuk semua transaksi.

Program pada gambar 7.17 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe Dictionary tanpa pengurutan data (*unsorted*) untuk mencetak perulangan nilai pada Dictionary tersebut.


```

1  ▶ #Deklarasi variabel dengan tipe Dictionary
2  transaction_data = {
3      "transaction_id": 1000001,
4      "source_country": "United Kingdom",
5      "target_country": "Italy",
6      "send_currency": "GBP",
7      "send_amount": 1000.00,
8      "target_currency": "EUR",
9      "fx_rate": 1.1648674,
10     "fee_pct": 0.50,
11     "platform": "mobile"
12 }
13
14 for key in transaction_data:
15     print(key, ":", transaction_data[key])
16

```

Gambar 7.17 Contoh Penggunaan Kontrol For Loop dengan Tipe Dictionary.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

transaction_id : 1000001
source_country : United Kingdom
target_country : Italy
send_currency : GBP
send_amount : 1000.0
target_currency : EUR
fx_rate : 1.1648674
fee_pct : 0.5
platform : mobile

```

Gambar 7.18 Hasil Keluaran Program

Studi Kasus 7.8

Program pada gambar 7.19 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe Dictionary dengan pengurutan data (*sorted*) untuk mencetak perulangan nilai pada Dictionary tersebut.

```

1  ▶ #Deklarasi variabel dengan tipe Dictionary
2  transaction_data = {
3      "transaction_id": 1000001,
4      "source_country": "United Kingdom",
5      "target_country": "Italy",
6      "send_currency": "GBP",
7      "send_amount": 1000.00,
8      "target_currency": "EUR",
9      "fx_rate": 1.1648674,
10     "fee_pct": 0.50,
11     "platform": "mobile"
12 }
13
14 for key in sorted(transaction_data):
15     print(key, ":", transaction_data[key])
16

```

Gambar 7.19 Contoh Penggunaan Kontrol For Loop dengan Tipe Dictionary.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

fee_pct : 0.5
fx_rate : 1.1648674
platform : mobile
send_amount : 1000.0
send_currency : GBP
source_country : United Kingdom
target_country : Italy
target_currency : EUR
transaction_id : 1000001

```

Gambar 7.20 Hasil Keluaran Program

Studi Kasus 7.9

Program pada gambar 7.21 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe Dictionary dengan fungsi Item() untuk mencetak perulangan nilai pada Dictionary tersebut.

```

1  ▶ #deklarasi variabel dengan tipe Dictionary
2  transaction_data = {
3      "transaction_id": 1000001,
4      "source_country": "United Kingdom",
5      "target_country": "Italy",
6      "send_currency": "GBP",
7      "send_amount": 1000.00,
8      "target_currency": "EUR",
9      "fx_rate": 1.1648674,
10     "fee_pct": 0.50,
11     "platform": "mobile"
12 }
13 #k : key
14 #v : value
15 for k,v in transaction_data.items():
16     print(k,"->",v)
17

```

Gambar 7.21 Contoh Penggunaan Kontrol For Loop dengan Tipe Dictionary.

Perhatikan bahwa k dan v hanyalah alias standar untuk 'key' dan 'value', Anda dapat memilih penamaan alternatif lainnya.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

transaction_id -> 1000001
source_country -> United Kingdom
target_country -> Italy
send_currency -> GBP
send_amount -> 1000.0
target_currency -> EUR
fx_rate -> 1.1648674
fee_pct -> 0.5
platform -> mobile

```

Gambar 7.22 Hasil Keluaran Program

Studi Kasus 7.10

Program pada gambar 7.23 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe Dictionary dengan fungsi Item() untuk mencetak perulangan nilai pada Dictionary tersebut.

```

1  ▶ #Deklarasi variabel dengan tipe Dictionary
2  transaction_data_n = {
3      "transaction_1":{
4          "transaction_id": 1000001,
5          "source_country": "United Kingdom",
6          "target_country": "Italy",
7          "send_currency": "GBP",
8          "send_amount": 1000.00,
9          "target_currency": "EUR",
10         "fx_rate": 1.1648674,
11         "fee_pct": 0.50,
12         "platform": "mobile"
13     },
14     "transaction_2":{
15         "transaction_id": 1000002,
16         "source_country": "United Kingdom",
17         "target_country": "Germany",
18         "send_currency": "GBP",
19         "send_amount": 3320.00,
20         "target_currency": "EUR",
21         "fx_rate": 1.1648674,
22         "fee_pct": 0.50,
23         "platform": "Web"
24     },
25
26     "transaction_3":{
27         "transaction_id": 1000003,
28         "source_country": "United Kingdom",
29         "target_country": "Belgium",
30         "send_currency": "GBP",
31         "send_amount": 1250.00,
32         "target_currency": "EUR",
33         "fx_rate": 1.1648674,
34         "fee_pct": 0.50,
35         "platform": "Web"
36     }
37 }
38
39 for k, v in transaction_data_n.items():
40     if type(v) is dict and k == 'transaction_2':
41         for sk, sv in v.items():
42             print(sk, '-->', sv)
43

```

Gambar 7.23 Contoh Penggunaan Kontrol For Loop dengan Tipe Dictionary.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

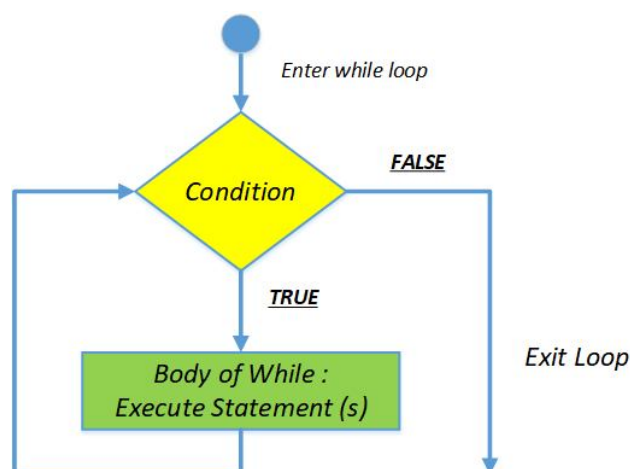
```
transaction_id --> 1000002
source_country --> United Kingdom
target_country --> Germany
send_currency --> GBP
send_amount --> 3320.0
target_currency --> EUR
fx_rate --> 1.1648674
fee_pct --> 0.5
platform --> Web
```

Gambar 7.24 Hasil Keluaran Program

7.3 Kontrol *While Loop*

Struktur perulangan *while* (*While Loop*) digunakan untuk melakukan proses perulangan yang pemeriksaan syarat / kondisinya dilakukan pada awal proses. Struktur *While Loop* umumnya digunakan untuk melakukan proses perulangan dengan frekuensinya belum diketahui pada saat proses perulangan dimulai [7]. Pernyataan *While Loop* dalam bahasa pemrograman Python berulang kali mengeksekusi pernyataan target selama kondisi yang diberikan benar (*TRUE*).

Struktur kontrol *While Loop* dalam bahasa Python dapat dilihat pada gambar 7.25



Gambar 7.25 Struktur Perulangan While Loop

Berikut ini sintak umum penulisan struktur perulangan While Loop di bahasa Python :

```
While condition :  
    execute statement (s)
```

Gambar 7.26 Struktur Kontrol While Loop

Penulisan kondisi perulangan di dalam python **TIDAK** cukup menuliskan perintah `while` disertai dengan kondisi bersyarat dan diakhiri dengan tanda : (titik dua). Setiap pernyataan yang akan dieksekusi harus diberikan Indentasi Tab atau spasi 2x sebagaimana aturan penulisan sintak dalam Python. Setiap Perintah didalam Python pun **TIDAK** diakhiri dengan tanda ; (titik koma). Bagian kondisi adalah sebuah variabel / atau nilai yang bertipe data boolean. Baik berupa nilai True/False secara langsung, atau pun sebuah ekspresi logika. Jika kondisi bernilai True, maka bagian pernyataan (*execute statements*) akan dieksekusi berulang kali oleh komputer.

Studi Kasus 7.11

Program pada gambar 7.27 merupakan contoh penerapan struktur kontrol perulangan *While Loop*. Program tersebut mendemonstrasikan cara menyeleksi kelulusan mahasiswa menurut nilai ujiannya.

```
1  ▶  
2  n = int(input("Masukkan batas perulangan : "))  
3  
4  print("\nMencetak perulangan While Loop secara vertikal :")  
5  i=0  
6  while (i<=n) :  
7      print(i)  
8      i = i + 1  
9
```

Gambar 7.27 Contoh Program dengan Kontrol While Loop

Perhatikan gambar 7.27, variabel *i* diberi nilai awal 0, sedangkan variabel *n* adalah nilai batas akhir yang diinput melalui keyboard. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 7.28.

```
Masukkan batas perulangan : 5

Mencetak perulangan While Loop secara horisontal :
0
1
2
3
4
5
```

Gambar 7.28 Hasil Keluaran Program

Mula-mula variabel *i* bernilai 0, kemudian perintah `print(i)` menghasilkan keluaran 0 disertai dengan enter. Proses yang sama kemudian diulang lagi dengan *i* berikutnya ditambah satu menjadi 1. Kemudian Perintah `print(i)` menghasilkan keluaran 1. Demikian seterusnya. program akan berhenti saat perulangan mencapai `batas_akhir-1`.

Studi Kasus 7.12

Program pada gambar 7.29 merupakan contoh penerapan struktur kontrol While Loop untuk mencetak perulangan nilai yang ada pada variabel *i* dengan batas perulangan *n* yang diinput melalui keyboard.

```
1  ▶ n = int(input("Masukkan batas perulangan : "))
2
3
4  print("\nMencetak perulangan While Loop secara horisontal :")
5  i=0
6  while (i<=n) :
7      print(i, end=" ")
8      i = i + 1
9
```

Gambar 7.29 Contoh penerapan kontrol While Loop

Mula-mula variabel *i* bernilai 0, kemudian perintah `print(i)` menghasilkan keluaran 0. Proses yang sama kemudian diulang lagi dengan *i* berikutnya ditambah satu menjadi 1. Kemudian Perintah `print(i)` menghasilkan keluaran 1. Demikian seterusnya. program akan berhenti saat perulangan mencapai nilai *n*. Adapun hasil keluaran dari program diatas ditampilkan sebagai berikut :

```
Masukkan batas perulangan : 5

Mencetak perulangan While Loop secara horisontal :
0 1 2 3 4 5
```

Gambar 7.30 Hasil Keluaran Program

Studi Kasus 7.13

Program pada gambar 7.31 merupakan contoh penerapan struktur kontrol While Loop untuk mencetak perulangan nilai yang ada pada variabel *i* dengan batas perulangan yang diinput melalui keyboard.

```
1  ▶ print("Program deret aritmatika")
2
3  print("=====")
4
5  sukuAwal = input("Masukkan suku awal : ")
6  beda = input("Masukkan beda : ")
7  banyak = int(input("Masukkan banyak deret : "))
8
9  jumDeret = banyak
10 suku = int(sukuAwal)
11 jum = 0
12 i = 0
13
14 while (i < jumDeret) :
15     print("Suku ke-", i, "=", suku)
16     suku = suku + int(beda)
17     jum = jum + 1
18     i = i + 1
19
20 print("Jumlah deret = ", jum)
21
```

Gambar 7.31 Contoh penerapan kontrol While Loop

Perhatikan gambar 7.31, program akan menunggu kita untuk memasukkan nilai suku awal, beda dan banyak deret. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Suku awal : 3
- Beda : 2
- Banyak deret : 5

, maka hasil keluaran yang ditampilkan pada gambar 7.32.

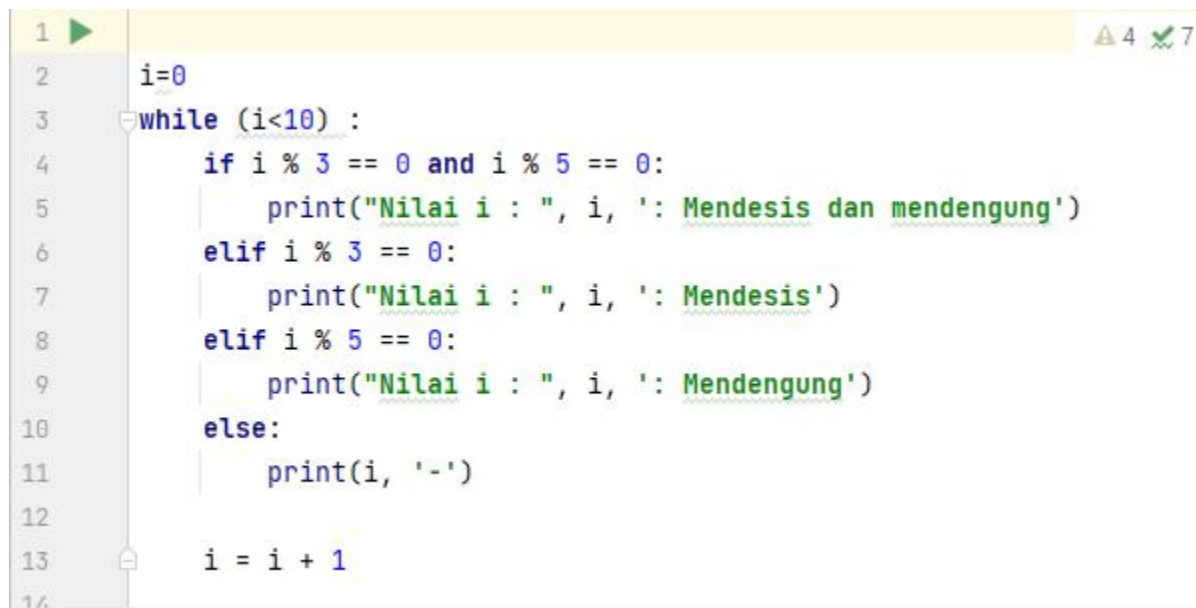
```
Program deret aritmatika
=====
Masukkan suku awal : 3
Masukkan beda : 2
Masukkan banyak deret : 5
Suku ke- 1 = 3
Suku ke- 2 = 5
Suku ke- 3 = 7
Suku ke- 4 = 9
Suku ke- 5 = 11
Jumlah deret = 5
```

Gambar 7.32 Contoh Penggunaan Kontrol While Loop

Mula-mula variabel *i* bernilai 1, kemudian perintah `print()` akan mencetak nilai *i* dan nilai suku sehingga menghasilkan keluaran untuk nilai *i* adalah 1 dan nilai suku adalah nilai 3, dimana nilai awal (1) ditambah dengan beda (2). Proses yang sama kemudian diulang lagi dengan *i* berikutnya ditambah satu menjadi 1. Kemudian Perintah `print()` mencetak nilai *i* adalah 2 dan nilai suku adalah 5, dimana nilai suku (3) ditambah beda (2). Demikian seterusnya. program akan berhenti saat perulangan mencapai nilai *jumderet*. setelah itu nilai dari variabel *i*, nilai dari variabel *suku* dan variabel *jum* ditampilkan ke layar.

Studi Kasus 7.14

Program pada gambar 7.33 merupakan contoh penerapan struktur kontrol While Loop untuk mencetak perulangan nilai yang ada pada variabel *i* dengan batas perulangan yang diinput melalui keyboard.



```

1  ▶
2  i=0
3  while (i<10) :
4      if i % 3 == 0 and i % 5 == 0:
5          print("Nilai i : ", i, ': Mendesisis dan mendengung')
6      elif i % 3 == 0:
7          print("Nilai i : ", i, ': Mendesisis')
8      elif i % 5 == 0:
9          print("Nilai i : ", i, ': Mendengung')
10     else:
11         print(i, '-')
12
13     i = i + 1
14

```

Gambar 7.33 Contoh Penggunaan Kontrol While Loop dengan Kontrol IF dan Operator AND

Perhatikan gambar 7.33, program akan memeriksa nilai *i* sebanyak batas akhir yang bernilai 10, dimana nilai awal untuk variabel *i* adalah 0. Terdapat tiga kondisi percabangan IF yang dibuat, yaitu *if i%3 == 0 and i%5 ==0*, *elif i%3 == 0* dan *elif i%5 == 0*. Jika kondisi pemeriksaan bernilai benar (TRUE), maka nilai *i* akan dicetak beserta keterangannya. Jika tidak memenuhi ketiga kondisi tersebut, maka akan masuk ke bagian *else*. Setelah kode program diatas dijalankan, maka hasil keluaran yang ditampilkan pada gambar 7.34.



```

Nilai i : 0 : Mendesisis dan mendengung
1 -
2 -
Nilai i : 3 : Mendesisis
4 -
Nilai i : 5 : Mendengung
Nilai i : 6 : Mendesisis
7 -
8 -
Nilai i : 9 : Mendesisis

```

Gambar 7.35 Hasil Keluaran Program

Studi Kasus 7.15

Kontrol While Loop pada tipe List

Program pada gambar 7.36 merupakan contoh penerapan struktur kontrol For Loop menggunakan tipe List untuk mencetak perulangan nilai yang ada pada .

```
1  #deklarasi variabel dengan tipe list
2  number_list = []
3  n = int(input("Input ukuran list : "))
4  jum = 0
5  print("\n")
6  i = 0
7  while (i<n) :
8      print("Input nilai pada indeks ke-: ", i, )
9      list_item = int(input())
10     jum = jum + list_item
11     #menambahkan data ke variabel list
12     number_list.append(list_item)
13     i = i + 1
14     print("Data List : ", number_list)
15     print("Jumlah : ", jum)
16
```

Contoh 7.36 Contoh Penggunaan Kontrol While Loop pada tipe List

Perhatikan gambar 7.36, terdapat deklarasi variabel `number_list` yang memiliki tipe List untuk menyimpan nilai yang diinput oleh pengguna. Program akan menunggu untuk memasukkan ukuran list yang bertipe Integer dan disimpan dalam variabel `n`. Kemudian mula-mula variabel `i` akan bernilai 0 (karena batas_bawah diberi nilai 0) dan akan diulang hingga mencapai nilai `n`. Jika kondisi pemeriksaan nilai `i` bernilai TRUE, maka anda akan diminta menginput nilai indeks melalui keyboard. Setelah itu, nilai yang diinput tersebut akan dimasukkan ke dalam variabel `number_list` setelah melalui proses konversi dari tipe string ke tipe Integer. Jika kondisi pemeriksaan nilai `i` bernilai salah (FALSE), maka program akan mencetak nilai `number_list` dan nilai `jum`. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Input ukuran List : 5

- Input nilai pada indeks : 1, 3, 5, 7, 9

, maka hasil keluaran yang ditampilkan pada gambar 7.37.

```
Input ukuran list : 5

Input nilai pada indeks ke-: 0
1
Input nilai pada indeks ke-: 1
3
Input nilai pada indeks ke-: 2
5
Input nilai pada indeks ke-: 3
7
Input nilai pada indeks ke-: 4
9
Data List : [1, 3, 5, 7, 9]
Jumlah : 25
```

Gambar 7.37 Hasil Keluaran Program

Studi Kasus 7.16

Penggunaan Kontrol While Loop menggunakan tipe Tuple

Kita dapat membuat sebuah tipe Tuple melalui tahapan berikut :

1. Membuat sebuah List.
2. Menginput elemen List.
3. Mengkonversi tipe List ke tipe Tuple.

Program pada gambar 7.38 merupakan contoh penerapan struktur kontrol While Loop menggunakan tipe Tuple untuk mencetak perulangan nilai pada Tuple tersebut.

```

1  ▶ #deklarasi variabel dengan tipe list
2  listA = []
3  n = int(input("Input ukuran list : "))
4  jum = 0
5  #print("\n")
6  i = 0
7  while (i<n) :
8      print("Input nilai pada indeks ke-: ", i, )
9      list_item = int(input())
10     listA.append(list_item)
11     i = i + 1
12
13  #konversi tipe List menjadi tipe Tuple
14  tupleA = tuple(listA)
15  jum = sum(tupleA)
16  print("Data Tuple A : ", tupleA)
17  print("Jumlah item pada Tuple A: ", jum)
18

```

Gambar 7.39 Contoh Penggunaan Kontrol While Loop dengan Tipe Tuple

Perhatikan gambar 7.39, terdapat deklarasi variabel listA yang memiliki tipe List. Program akan menunggu untuk memasukkan jumlah ukuran list yang bertipe Integer dan disimpan dalam variabel n. variabel jum digunakan untuk menghitung total nilai yang diinput. Kemudian mula-mula variabel i akan bernilai 0 (karena batas_bawah diberi nilai 0) dan akan diulang hingga mencapai nilai n. Jika kondisi pemeriksaan nilai i bernilai TRUE, maka pengguna akan diminta menginput nilai melalui keyboard. Nilai tersebut akan dikonversi ke tipe Integer dan dimasukkan ke dalam variabel list_item. Kemudian semua nilai yang ada di variabel list_item dimasukkan ke variabel listA menggunakan perintah Append(). Jika kondisi pemeriksaan nilai i bernilai salah (FALSE), maka program akan mengkonversi nilai pada ListA menjadi bertipe Tuple dan disimpan ke dalam variabel TupleA. Kemudian menghitung total nilai yang ada di tupleA dan menyimpannya ke variabel jum. Kemudian program akan mencetak nilai tupleA dan nilai jum. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

- Input ukuran List : 5
- Input nilai pada indeks ke-0 : 1

- Input nilai pada indeks ke-1 : 2
- Input nilai pada indeks ke-2 : 3
- Input nilai pada indeks ke-3 : 4
- Input nilai pada indeks ke-4 : 5,

maka hasil keluaran yang ditampilkan pada gambar 7.40.

```
Input ukuran list : 5
Input nilai pada indeks ke-: 0
1
Input nilai pada indeks ke-: 1
2
Input nilai pada indeks ke-: 2
3
Input nilai pada indeks ke-: 3
4
Input nilai pada indeks ke-: 4
5
Data Tuple A : (1, 2, 3, 4, 5)
Jumlah item pada Tuple A: 15
```

Gambar 7.40 Hasil Keluaran Program

Studi Kasus 7.17

Penggunaan Kontrol While Loop menggunakan tipe Dictionary

Kita dapat membuat sebuah tipe Dictionary menggunakan perulangan menggunakan While Loop. Program pada gambar 7.41 merupakan contoh penerapan struktur kontrol While Loop menggunakan tipe Dictionary dengan tambahan **fungsi Iter()** dan **fungsi Next()** untuk mencetak perulangan nilai pada Dictionary tersebut.

```

1  ▶
2  new_dict = {'Jan':31, 'Feb':29, 'Mar':31, 'Apr':30, 'May':31, 'Jun':30,
3             'Jul':31, 'Aug':31, 'Sep':30, 'Oct':31, 'Nov':30, 'Dec':31}
4  bulan = iter(new_dict.items())
5  default = object()
6
7  print("new_dict berisi : ")
8  while bulan :
9      element_dict = next(bulan, default)
10     #kondisi ini untuk mengetahui apakah iterator sudah habis atau belum.
11     if element_dict is default:
12         break
13     print("{}:{}".format(*element_dict), end=" ")
14

```

Gambar 7.41 Contoh Penggunaan Kontrol While Loop dengan Tipe Dictionary.

Perhatikan gambar 7.41, terdapat deklarasi variabel `new_dict` yang memiliki tipe Dictionary dan sudah berisi data seperti terlihat pada gambar diatas. **Fungsi `iter()`** digunakan untuk mengembalikan iterator dari suatu objek yang diberikan. **Fungsi `next()`** pada dasarnya mengembalikan item selanjutnya dari suatu iterator. Bila item yang diambil sudah habis, maka fungsi `next()` selanjutnya akan mengembalikan nilai default yang sudah ditentukan sebelumnya. Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

new_dict berisi :
Jan:31 Feb:29 Mar:31 Apr:30 May:31 Jun:30 Jul:31 Aug:31 Sep:30 Oct:31 Nov:30 Dec:31
Process finished with exit code 0

```

Gambar 7.42 Hasil Keluaran Program

7.4 Kontrol Nested Loop

Bahasa pemrograman Python memungkinkan untuk menggunakan satu perulangan di dalam perulangan lain atau biasanya disebut perulangan bersarang (*Nested Loop*). Kita dapat meletakkan semua jenis Loop di dalam jenis Loop lainnya. Misalnya, perulangan For bisa berada di dalam perulangan While atau sebaliknya.

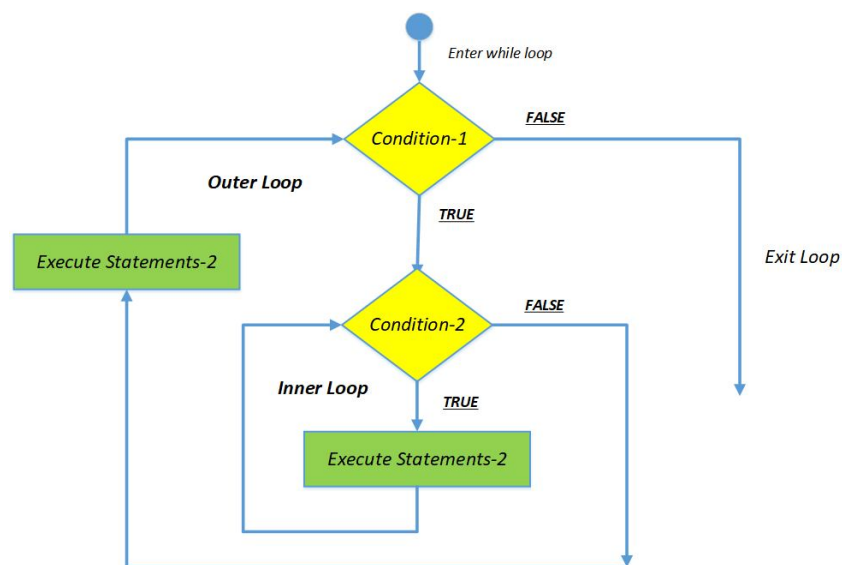
a. Kontrol Nested While Loop

Di dalam perulangan Nested While Loop memiliki dua tipe perintah While, yaitu :

- *Outer for Loop*
- *Inner for Loop*

Cara bekerja Nested While Loop :

- 1) Pertama kali, kondisi bersyarat dari Loop terluar (*Outer for Loop*) dievaluasi terlebih dulu dan ketika menghasilkan nilai benar (TRUE), maka aliran kontrol akan masuk ke Loop terdalam (*Inner for Loop*).
- 2) Perulangan pada Loop terdalam (*Inner for Loop*) dilakukan hingga mencapai kondisi bersyarat memiliki nilai salah (FALSE). Kemudian perulangan akan
- 3) Perulangan pada Loop terluar (*Outer for Loop*) akan berlanjut hingga kondisi bersyarat pada Loop terluar bernilai salah (FALSE).



Gambar 7.43 Struktur Perulangan Nested While Loop

Gambar 7.44 merupakan sintak penulisan perulangan Nested While Loop di bahasa Python.

```
# outer for loop
While condition-1 :
    # inner for loop
    While condition-2 :
        Statements-2

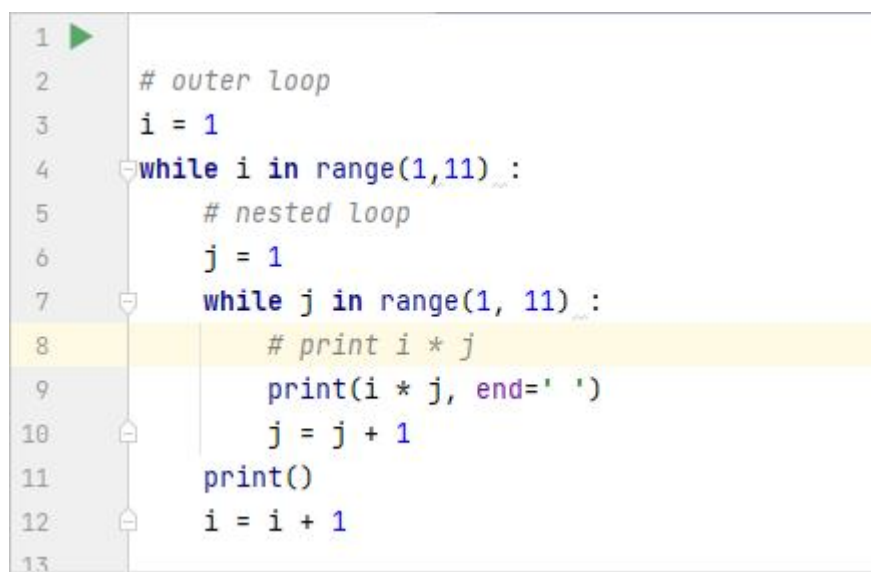
    Statements-1
```

Gambar 7.44 Struktur Kontrol Nested While Loop

Studi Kasus 7.18

Penggunaan Kontrol Nested While Loop

Program pada gambar 7.45 merupakan contoh penerapan struktur kontrol Nested While Loop untuk mencetak perulangan nilai $i * j$.

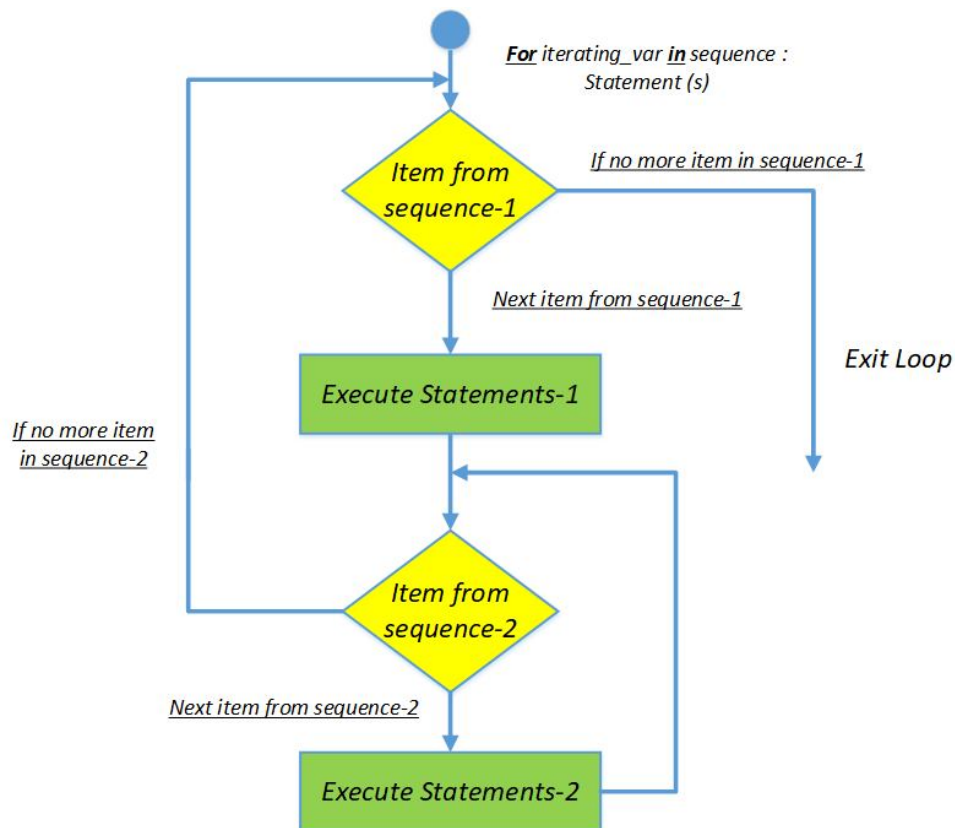


```
1  # outer loop
2  i = 1
3  while i in range(1,11):
4      # nested loop
5      j = 1
6      while j in range(1, 11):
7          # print i * j
8          print(i * j, end=' ')
9          j = j + 1
10     print()
11     i = i + 1
12
13
```

Gambar 7.45 Contoh Penggunaan Kontrol Nested While Loop

b. Kontrol Nested For Loop

Dalam Python, kontrol For Loop digunakan untuk mengulangi urutan seperti: List, String, Tuple, objek perulangan lainnya seperti range.



Gambar 7.46 Struktur Nested For Loop

Berikut ini sintak umum penulisan struktur perulangan Nested For Loop di bahasa Python :

```

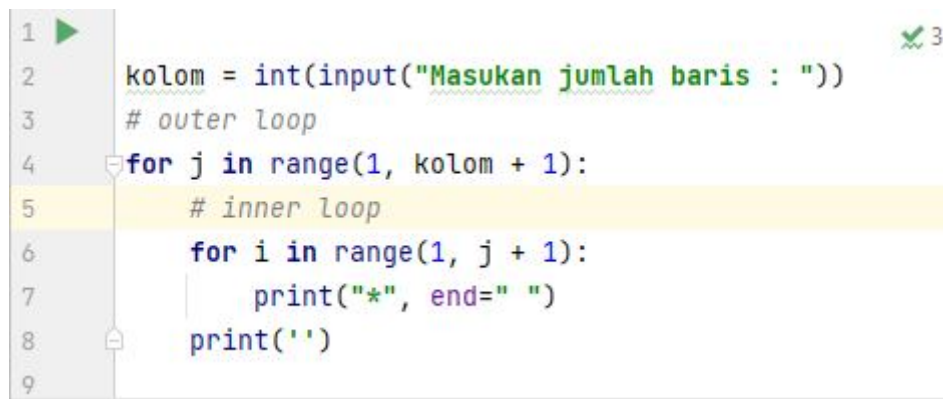
# outer for loop
for element in sequence
# inner for loop
for element in sequence :
    body of inner for loop
    body of outer for loop
  
```

Gambar 7.47 Sintak Penulisan Kontrol Nested For Loop

Studi Kasus 7.19

Penggunaan Kontrol Nested For Loop

Program pada gambar 7.48 merupakan contoh penerapan struktur kontrol Nested For Loop untuk mencetak perulangan tanda * sebanyak jumlah barisa yang diinput.



```
1  ▶
2      kolom = int(input("Masukan jumlah baris : "))
3      # outer loop
4      for j in range(1, kolom + 1):
5          # inner loop
6              for i in range(1, j + 1):
7                  print("*", end=" ")
8              print('')
9
```

Gambar 7.48 Contoh Penggunaan Nested For Loop

Perhatikan gambar 7.48, program akan menunggu inputan jumlah kolom yang disimpan dalam variabel baris dengan tipe Integer. Pada perulangan pertama (*outer loop*), Mula-mula variabel j diberikan nilai awal =1 dan berulang bertambah satu selama kondisi masih dibawah batas akhir, yaitu kolom + 1. Jika nilai j lebih besar dari nilai kolom + 1, maka perulangan akan berakhir. Jika nilai j masih dibawah nilai kolom + 1, maka proses selanjutnya adalah mengisi variabel i dengan nilai awal = 1 dan berulang bertambah satu selama kondisi masih dibawah nilai j+1. Kemudian, program akan mencetak tanda * selama kondisi nilai i masih dibawah nilai j+1.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

Masukan jumlah baris : 10
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

```

Gambar 7.49 Hasil Keluaran Program

7.5 Pernyataan kontrol untuk Perulangan (Loop)

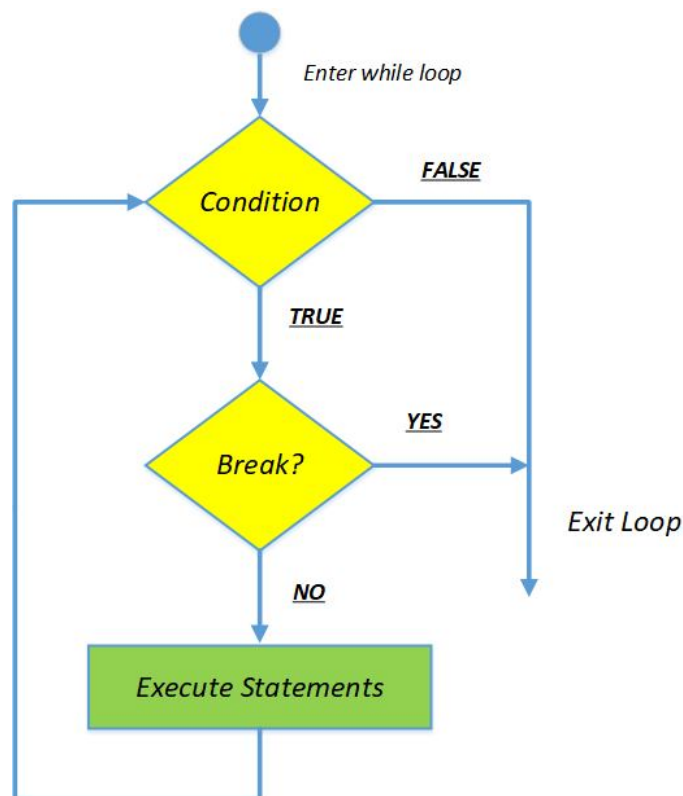
Terdapat tiga pernyataan kontrol untuk perulangan (loop), selengkapnya dapat dilihat pada tabel 7.3.

Pernyataan Kontrol	Deskripsi
Break	Pernyataan break menangani penghentian loop di mana ia digunakan. Jika pernyataan break digunakan di dalam loop bersarang (<i>inner loop</i>), loop saat ini dihentikan, dan aliran akan dilanjutkan dengan kode yang diikuti setelah loop.
Continue	Pernyataan Continue melewati kode yang datang setelahnya, dan kontrol diteruskan kembali ke awal perulangan untuk iterasi berikutnya.
Pass	Pernyataan pass digunakan sebagai pengganti di dalam loop, fungsi, kelas, pernyataan if yang dimaksudkan untuk diimplementasikan nanti.

Studi Kasus 7.20

Penggunaan Pernyataan Break

Pernyataan **break** mengakhiri loop yang memuatnya. Kontrol program akan mengalir ke pernyataan setelah badan perulangan (*exit loop*). Jika pernyataan **break** berada di dalam loop bersarang (loop di dalam loop lain), pernyataan **break** akan mengakhiri loop terdalam. Gambar 7.50 adalah flowchart dari struktur kontrol perulangan **break** :



Gambar 7.50 Struktur Pernyataan Break

Program pada gambar 7.51 merupakan contoh penerapan pernyataan **Break** untuk pada saat mencetak data pada variabel `my_list`.

```

1  ▶
2  my_list = ['Mancing', 'Berkebun', 'Jalan-jalan', 'Shopping']
3
4  for i in range(len(my_list)):
5      print(my_list[i])
6      if my_list[i] == 'Berkebun':
7          print("Ketemu Berkebun")
8          #penggunaan perintah break
9          break
10         print("Setelah perintah break")
11
12     print('Loop dihentikan')
13

```

Gambar 7.50 Contoh Penggunaan Pernyataan Break

Perhatikan gambar 7.50, variabel `my_list` sudah berisi data dan akan dicetak menggunakan perulangan For Loop. Tetapi terdapat kondisi dimana pada saat menemukan variabel `my_list` yang berisi nilai 'Berkebun', maka program akan dihentikan dan hasilnya akan dicetak.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```

Mancing
Berkebun
Ketemu Berkebun
Loop dihentikan

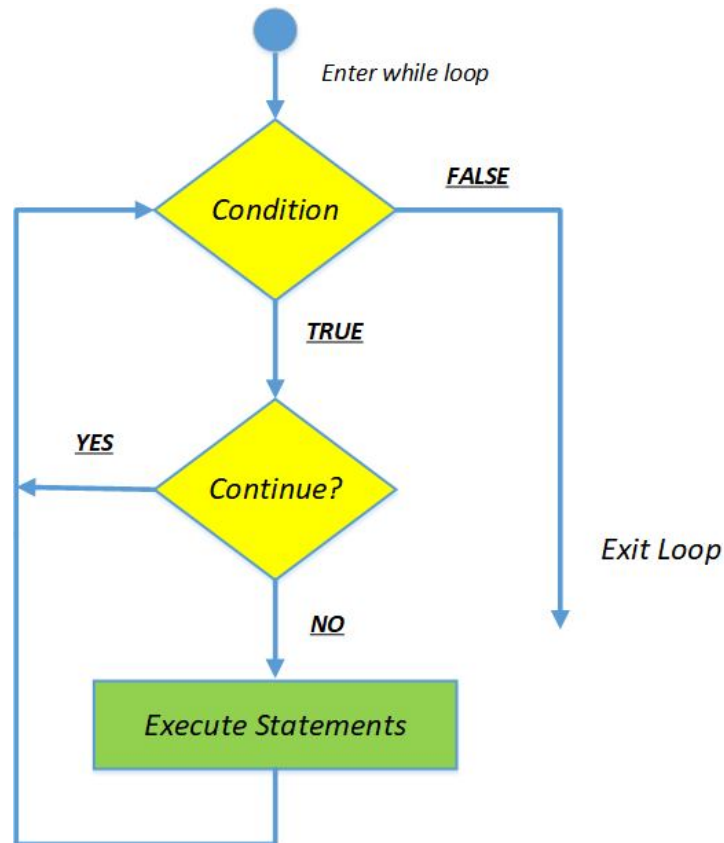
```

Gambar 7.51 Hasil Keluaran Program

Studi Kasus 7.21

Penggunaan Pernyataan Continue

Pernyataan **Continue** digunakan untuk melewati sisa kode di dalam satu loop untuk iterasi saat ini saja. Loop tidak berhenti tetapi berlanjut dengan iterasi berikutnya.



Gambar 7.52 Struktur Pernyataan Continue

Program pada gambar 7.53 merupakan contoh penerapan pernyataan **Continue** untuk pada saat mencetak data pada variabel i dan j.

```

1  ▶
2  for i in range(3):
3      for j in range(3):
4          if j==2:
5              #penggunaan perintah continue
6              continue
7          print("Nilai yang tercetak adalah : ",i,j)
8
  
```

Gambar 7.53 Struktur Pernyataan Continue

Perhatikan gambar 7.53, terdapat variabel i dan j dengan nilai awal masing-masing adalah 0. Perulangan pertama digunakan untuk mengulang nilai i dengan nilai batas akhir adalah 3 (tiga). Jika kondisi i lebih kecil dari nilai batas akhir, maka proses berikutnya adalah mengulang nilai j dengan nilai batas akhir adalah 3. Terdapat kondisi dimana pada saat nilai j sama dengan 2,

maka perintah `continue` dijalankan. Kemudian program akan mencetak nilai `i` dan `j`.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```
Nilai yang tercetak adalah : 0 0
Nilai yang tercetak adalah : 0 1
Nilai yang tercetak adalah : 1 0
Nilai yang tercetak adalah : 1 1
Nilai yang tercetak adalah : 2 0
Nilai yang tercetak adalah : 2 1
```

Gambar 7.54 Hasil Keluaran Program

Studi Kasus 7.21

Penggunaan Pernyataan `Pass`

Pernyataan **`Pass`** tidak akan mengerjakan perintah apapun. Terkadang ada situasi dalam pemrograman, di mana kita perlu mendefinisikan blok yang kosong secara sintaksis. Kita dapat mendefinisikan blok itu dengan perintah **`pass`**. Pernyataan `pass` adalah pernyataan null dalam bahasa Python. Ketika interpreter menemukan perintah **`pass`** dalam program, ia tidak akan mengembalikan operasi. Tidak ada yang terjadi ketika perintah **`pass`** dieksekusi.

Program pada gambar 7.55 merupakan contoh penerapan pernyataan **`Pass`** pada saat mencetak data dari

```
1  ▶
2  kata = 'Python'
3  for huruf in kata:
4      if huruf == 'h':
5          #penggunaan perintah pass
6          pass
7          print ('Disini adalah blok pass')
8          print ('Huruf tercetak :', huruf)
9
10 print ("Good bye!")
11
```

Gambar 7.55 Contoh Penggunaan Pernyataan `Pass`

Perhatikan gambar 7.55, terdapat variabel kata yang berisi 'Python'. Perulangan pertama digunakan untuk mengulang nilai huruf dengan nilai batas akhir adalah sejumlah huruf yang terdapat pada variabel kata. Jika kondisi huruf lebih kecil dari nilai batas akhir, maka proses berikutnya adalah mengulang nilai huruf. Terdapat kondisi dimana pada saat nilai huruf sama dengan 'h', maka perintah pass dijalankan. Kemudian program akan mencetak nilai huruf.

Setelah kode program diatas dijalankan, kemudian diinput dengan nilai sebagai berikut :

```
Huruf tercetak : P
Huruf tercetak : y
Huruf tercetak : t
Disini adalah blok pass
Huruf tercetak : h
Huruf tercetak : o
Huruf tercetak : n
Good bye!
```

Gambar 56. Hasil Keluaran Program

7.6 Praktikum

Langkah-langkah Praktikum

1. Buka Editor Python (IDLE / Pycharm / VSCode).
2. Buatlah file baru dengan membuka menu File > New > Source File atau dengan shortcut Ctrl + N.
3. Tulislah kode program berikut ini :

Program 7.1 : Praktikum71. Py

1. Buatlah program menggunakan bahasa Python untuk mencetak informasi total harga bensin menggunakan tipe List dengan scenario sebagai berikut :

Algoritma / Inisiasi Persoalan :

Variabel input :

N, harga

Proses :

Harga per satuan = $n * \text{harga}$

Output :

Harga per satuan

Berikut ini adalah kode program untuk menjawab permasalahan diatas :

```
1  ▶ #deklarasi variabel dengan tipe list
2  listA = []
3  # Input jumlah elemen list
4  n = int(input("Masukan jumlah N : "))
5  harga = 7650
6  # setiap sublist memiliki dua elemen nilai
7  for i in range(0, n):
8      print("Satuan ke-{} : ".format(i + 1), (i+1) * harga )
9      elist = [i+1, (i+1) * harga]
10     #proses menambahkan data ke variabel list
11     listA.append(elist)
12     print("Elemen Harga Bensin adalah : \n", listA)
13
```

Gambar 6.57 Kode Program Praktikum 71

2. Simpan Program ini dengan nama Praktikum71.py
3. Jalankan program praktikum61 di atas, kemudian tuliskan apa yang tercetak

4. alankan program praktikum71 tersebut, lalu tuliskan apa yang tercetak di layar.

Program 7.2 : Praktikum72. Py

1. Tuliskan kode program 7.2 berikut :

```
1  ▶
2  x=10
3  while x>1:
4      y=10
5      while y>=x:
6          print(x, end=" ")
7          y = y - 1
8      x = x - 1
9      print("\n")
10
```

Gambar 7.58 Kode Program Praktium72

2. Jalankan program praktikum72 di atas, kemudian tuliskan apa yang tercetak.

7.7 Rangkuman

1. Terdapat dua kontrol perulangan di dalam bahasa Python, yaitu For Loop dan While Loop.
2. Kontrol For Loop digunakan Ketika batas awal dan batas akhir dari perulangan diketahui di awal.
3. Kontrol While Loop digunakan ketika batas awal dan batas akhir dari perulangan belum diketahui di awal.
4. Kita dapat menggunakan perintah perulangan bersarang (Nested Loop) menggunakan For Loop dan While Loop.
5. Pernyataan **break** digunakan ketika hendak menghentikan perulangan yang aktif saat ini dan mengembalikan proses untuk keluar dari perulangan.
6. Pernyataan **continue** digunakan ketika hendak ingin melanjutkan perulangan yang aktif saat ini dan mengembalikan proses ke awal perulangan.

7. Pernyataan **pass** tidak mengerjakan operasi apapun, merupakan pernyataan null dalam bahasa Python.

7.8 Latihan

1. Buatlah program untuk mencetak susunan angka sehingga menghasilkan tampilan sebagai berikut :

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

Gambar 7.58 Keluaran program

2. Buatlah program untuk mencetak susunan angka sehingga menghasilkan tampilan sebagai berikut :

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

Gambar 7.59 Tampilan keluaran program

7.9 Tugas Mandiri

Kerjakan soal-soal berikut :

1. Buatlah program Python untuk mencetak susunan angka berikut ini :
10 20 30 40 50 60 70 80 90 100
2. Buatlah program Python untuk mencetak susunan angka berikut ini :
100 95 90 85 80 75 70 65 60 55
3. Buatlah program Python untuk mencetak deret angka berikut ini :
1 2 4 8 16 32 56 128 256 512 1024
4. Seseorang menyimpan uang Rp. 1.000.000 di bank dengan bunga ber-bunga 2% perbulan. Setelah satu bulan uangnya menjadi Rp. 1.020.000. Satu bulan berikutnya uang Rp. 1.020.000 ini mendapat bunga lagi 2%, yaitu Rp.20.400 sehingga setelah 2 bulan uangnya menjadi $\text{Rp. } 1.020.000 + \text{Rp. } 20.400 = \text{Rp. } 1.040.400$. Demikian seterusnya (bunga bulan ini ditambahkan ke saldo uangnya dan mendapatkan bunga lagi pada bulan berikutnya) . Susun program untuk menghitung dan mencetak jumlah uangnya setelah 10 bulan.



FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan

Jakarta Selatan, 12260

Telp: 021-5853753 Fax : 021-5853752

<http://fti.budiluhur.ac.id>