

5. Menyusun Pengulangan

Mengenal Pengulangan “for” dan “while”

Seperti pada bahasa pemrograman lainnya, Python memiliki mekanisme pengulangan untuk menjalankan pekerjaan – pekerjaan yang berulang. Pada umumnya pengulangan terdiri dua jenis. Ada pengulangan yang terbatas dan tidak terbatas. Pengulangan terbatas biasanya dilakukan pengulangan dari 1 hingga kesekian kali (n kali). Pengulangan terbatas menggunakan **for**. Sedangkan pengulangan tidak terbatas menggunakan **while**. Dalam pengulangan menggunakan **while** pengulangan akan terus dilakukan selama kondisional dalam pengulangan **while** tetap dalam keadaan **true** jika dalam keadaan **false** maka pengulangan **while** akan berhenti.

Menyusun Pengulangan dengan “for”

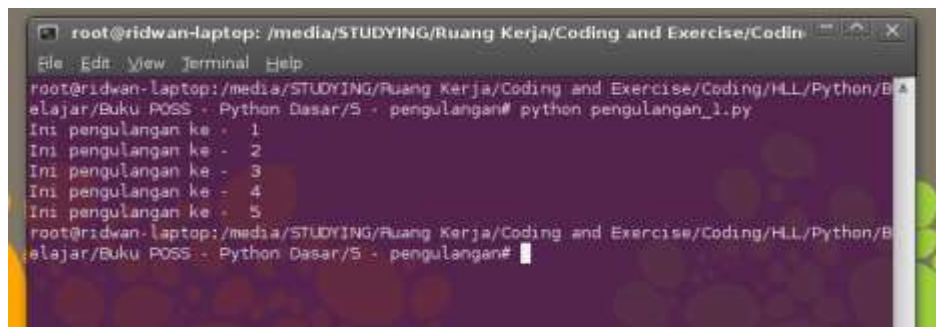
Kita melangkah ke pengulangan yang terbatas dulu yah. Dalam pengulangan **for**, tidak seperti di bahasa pemrograman C atau Java yang menggunakan nilai *incremental* untuk melakukan pengulangan. Di Python, **for** melakukan pengulangan dengan meng-iterasi elemen dari sebuah *list*. *List* ini dapat berisi kumpulan karakter, kumpulan *string*, kumpulan angka, atau kumpulan data jenis lainnya yang disediakan Python. (Untuk lebih lengkapnya di bab berikutnya akan dibahas lebih jauh tentang *List* di Python).

Misal disini ada sebuah *list* yang berisi [1, 2, 3, 4, 5], (sebuah list diawali oleh tanda '[' dan ditutup oleh tanda ']'). Banyaknya elemen pada *list* tersebut menentukan banyaknya pengulangan yang akan dilakukan saat melakukan pengulangan. Mari kita lihat implementasinya pada kode dibawah ini :

listing : kode pengulangan_1.py

```
# pengulangan sebanyak 5 kali for i in [1, 2, 3, 4, 5]:
    print "Ini pengulangan ke - ", i
```

Pada contoh diatas, akan dicetak teks “ini pengulangan ke - “ sebanyak 5 kali. Nilai 'i' pada pengulangan tersebut akan selalu berganti nilainya setiap tahap pengulangan dilakukan. Misal ketika pengulangan pertama, nilai 'i' akan berisi 1, ketika pengulangan kedua, nilai 'i' akan berisi 2, begitu seterusnya sampai elemen terakhir. Jika kode diatas dieksekusi akan menampilkan hasil seperti pada gambar dibawah ini :



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_1.py
Ini pengulangan ke - 1
Ini pengulangan ke - 2
Ini pengulangan ke - 3
Ini pengulangan ke - 4
Ini pengulangan ke - 5
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

<< gambar 5.1 hasil eksekusi pengulangan_1.py >>

Selain menggunakan *list* yang berisi angka, *List* yang berisi *string* dapat digunakan juga untuk melakukan pengulangan **for** di Python. Misal terdapat *list* yang berisi seperti berikut ["Rawon", "Nasi Kuning", "Soto Madura", "Kupat Tahu", "Kerak Telor", "Rendang Batoko", "Pempek Selam", "Ayam Betutu"], dalam *list* tersebut terdapat elemen sebanyak delapan jenis masakan nusantara. Dengan demikian ketika pengulangan **for** menggunakan *list* masakan tadi, pengulangan akan dijalankan sebanyak delapan kali. Mari Anda lihat implementasinya pada kode dibawah ini :

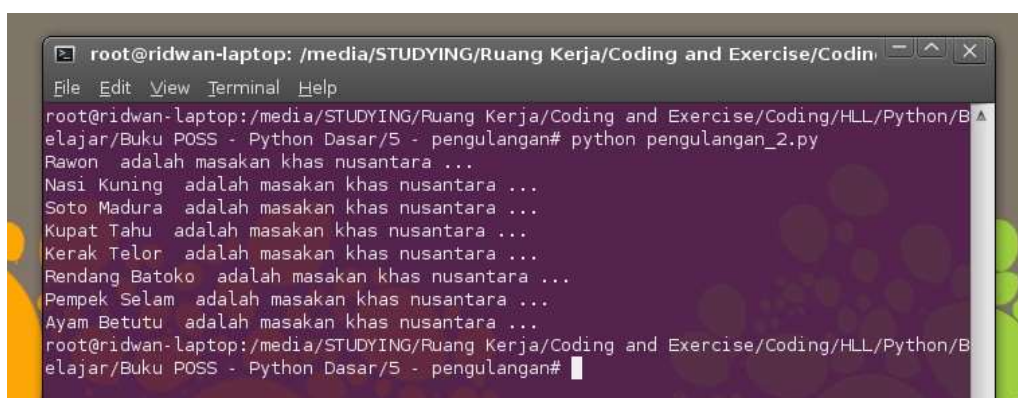
listing : kode pengulangan_2.py

```

# pengulangan sebanyak 8 kali
for i in ["Rawon", "Nasi Kuning", "Soto Madura", "Kupat Tahu", "Kerak Telor", "Rendang Batoko",
"Pempek Selam", "Ayam Betutu"] :
    print i, " adalah masakan khas nusantara ..."

```

Kode diatas jika dieksekusi akan terlihat seperti gambar dibawah ini :



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_2.py
Rawon adalah masakan khas nusantara ...
Nasi Kuning adalah masakan khas nusantara ...
Soto Madura adalah masakan khas nusantara ...
Kupat Tahu adalah masakan khas nusantara ...
Kerak Telor adalah masakan khas nusantara ...
Rendang Batoko adalah masakan khas nusantara ...
Pempek Selam adalah masakan khas nusantara ...
Ayam Betutu adalah masakan khas nusantara ...
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

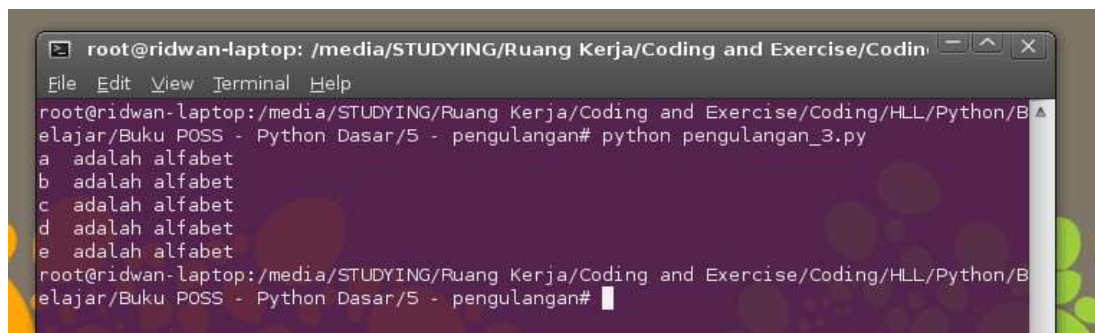
<< gambar 5.2 hasil eksekusi pengulangan_2.py >>

Sekarang Anda coba contoh terakhir dengan menggunakan *string*. *String* pada dasarnya merupakan *list* karakter. Misal terdapat *string* seperti berikut “abcde”. Jika *string* tersebut digunakan pada pengulangan **for**, maka akan terjadi pengulangan sebanyak lima kali. Coba lihat kode dibawah ini :

listing : kode pengulangan_3.py

```
# pengulangan sebanyak 5 kali for i in "abcde":
    print i, " adalah alfabet"
```

Kode diatas jika dieksekusi akan terlihat seperti gambar dibawah ini :



```
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_3.py
a  adalah alfabet
b  adalah alfabet
c  adalah alfabet
d  adalah alfabet
e  adalah alfabet
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#
```

<< gambar 5.3 hasil eksekusi pengulangan_3.py >>

Memahami Function “range”

Kalau teman – teman memperhatikan *list* yang dipakai pada pengulangan_1.py, pengulangan tersebut menggunakan *list* angka yang sudah jadi atau di-*hardcode*. Nah bagaimana nih kalau ingin membentuk suatu pola atau ingin membuat *list incremental* agar pengulangan **for** di Python ini mirip di Java atau C. Di Python terdapat fungsi yang bernama **range**. **Range** ini menghasilkan deret angka dengan parameter (*start*, *stop*, *step*). *Start* adalah batasawal dari *list*, *stop* adalah batas akhir dari *list*, *step* adalah jarak antar angka yang dihasilkan oleh **range**. Ada beberapa kasus penting yang perlu diperhatikan saat menggunakan **range**. Coba perhatikan kode dibawah ini :

listing : kode pengulangan_4.py

```
# pengulangan_4.py

# kasus - 1 : jika step tidak disertakan maka step akan diisi 1 secara default print range(1, 10)

# kasus - 2 : jika step disertakan maka step akan sesuai dengan angka yang diisikan
```

```

print range(1, 10, 2)
print range(1, 10, 3)
print range(1, 10, 4)
print range(1, 10, 5)

# kasus - 3 : jika step melebihi stop maka list hanya akan berisi start print range(1, 10, 11)

# kasus - 4 : jika start lebih besar nilainya daripada stop maka list akan kosong print range(10, 1)

# kasus - 5 : jika start lebih besar nilainya daripada stop dan # jika step melebihi stop maka list akan
kosong
print range(10, 1, 2)
print range(10, 1, 11)

# kasus - 6 : jika start lebih besar daripada stop dan step bernilai minus # dan jika start dikurangi
step menghasilkan angka positif
# maka list akan berisi deret angka menurun print range(10, 1, -1)

# kasus - 7 : jika start lebih besar daripada stop dan step bernilai minus
# dan jika start dikurangi step bernilai minus maka list hanya akan berisi start print range(10, 1, -11)

# kasus - 8 : jika step bernilai 0 maka akan terjadi error print range(1, 10, 0)

# kasus - 9 : jika start lebih besar daripada stop dan step bernilai 0 maka akan terjadi error print
range(10, 1, 0)

```

Kode diatas jika dieksekusi akan terlihat seperti gambar dibawah ini :

```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_4.py
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 3, 5, 7, 9]
[1, 4, 7]
[1, 5, 9]
[1, 6]
[1]
[]
[]
[]
[10, 9, 8, 7, 6, 5, 4, 3, 2]
[10]
Traceback (most recent call last):
  File "pengulangan_4.py", line 33, in <module>
    print range(1, 10, 0)
ValueError: range() step argument must not be zero
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

<< gambar 5.4 hasil eksekusi pengulangan_4.py >>

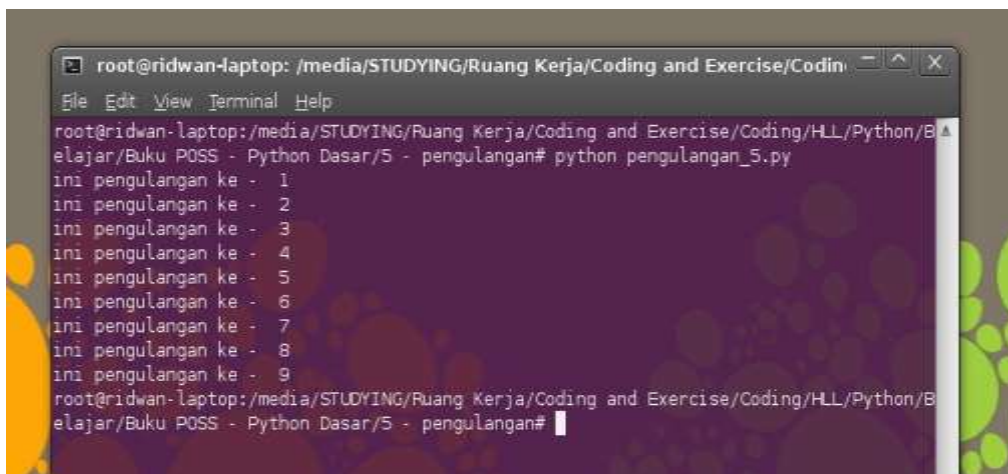
Menggunakan Function “range” pada Pengulangan “for”

Bagaimana dengan pengenalan **range** diatas ? Mudah kan ? Nah sekarang Anda akan coba menggunakan range dalam pengulangan **for**. Coba perhatikan contoh berikut :

listing : kode pengulangan_5.py

```
for i in range(1, 10):
    print "ini pengulangan ke - ", i
```

Pada contoh diatas akan terjadi pengulangan sebanyak 10 kali terhadap *statement* dibawah **for**. Dengan menggunakan **range**, Anda tidak perlu repot untuk membuat **list** terlebih dahulu untuk menentukan banyaknya pengulangan yang akan Anda lakukan terhadap *statement*. Jika kode diatas dieksekusi akan terlihat seperti gambar dibawah ini :



```
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_5.py
ini pengulangan ke - 1
ini pengulangan ke - 2
ini pengulangan ke - 3
ini pengulangan ke - 4
ini pengulangan ke - 5
ini pengulangan ke - 6
ini pengulangan ke - 7
ini pengulangan ke - 8
ini pengulangan ke - 9
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#
```

<< gambar 5.5 hasil eksekusi pengulangan_5.py >>

Agar lebih memahami lagi pengulangan **for**, kita coba lagi pelajari dua contoh berikut. Berikut ada kasus membuat sebuah segitiga yang dibuat dari kumpulan bintang dan membuat baris bilangan prima. Untuk lebih jelasnya coba perhatikan dua kasus berikut :

listing : kode pengulangan_6.py

```
# pengulangan_6.py for i in range(0, 10):
    for j in range(0, i+1): if j == i:
        print "x"
```



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_7.py
1 bukanlah bilangan prima
2 adalah bilangan prima
3 adalah bilangan prima
4 bukanlah bilangan prima
5 adalah bilangan prima
6 bukanlah bilangan prima
7 adalah bilangan prima
8 bukanlah bilangan prima
9 bukanlah bilangan prima
10 bukanlah bilangan prima
11 adalah bilangan prima
12 bukanlah bilangan prima
13 adalah bilangan prima
14 bukanlah bilangan prima
15 bukanlah bilangan prima
16 bukanlah bilangan prima
17 adalah bilangan prima
18 bukanlah bilangan prima
19 adalah bilangan prima
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

<< gambar 5.7 hasil eksekusi pengulangan_7.py >>

Menyusun Pengulangan dengan “while”

Sekarang kita akan bahas pengulangan yang menggunakan **while**. Pengulangan **while** memiliki cara kerja selama kondisi tertentu bernilai **true** maka pengulangan akan diteruskan sampai kondisi bernilai **false**. Tentunya dalam kondisi yang dipakai untuk eksekusi **while** memerlukan operator logika dan perbandingan seperti yang sudah di jelaskan di bab 3.

sebagai contoh coba lihat kode berikut. Kode berikut dieksekusi apabila variabel “angka” masih dibawah 10.

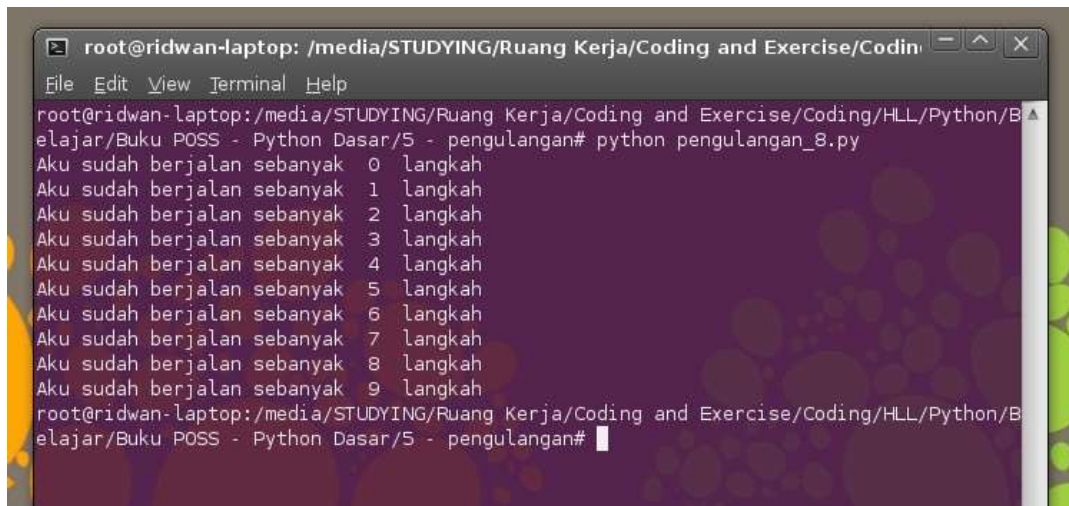
listing : kode pengulangan_8.py

```

angka = 0
while (angka < 10):
    print "Aku sudah berjalan sebanyak ", angka, " langkah "
    angka += 1

```

Kode diatas jika dieksekusi akan tampil seperti pada gambar berikut ini :



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_8.py
Aku sudah berjalan sebanyak 0 langkah
Aku sudah berjalan sebanyak 1 langkah
Aku sudah berjalan sebanyak 2 langkah
Aku sudah berjalan sebanyak 3 langkah
Aku sudah berjalan sebanyak 4 langkah
Aku sudah berjalan sebanyak 5 langkah
Aku sudah berjalan sebanyak 6 langkah
Aku sudah berjalan sebanyak 7 langkah
Aku sudah berjalan sebanyak 8 langkah
Aku sudah berjalan sebanyak 9 langkah
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

<< gambar 5.8 hasil eksekusi pengulangan_8.py >>

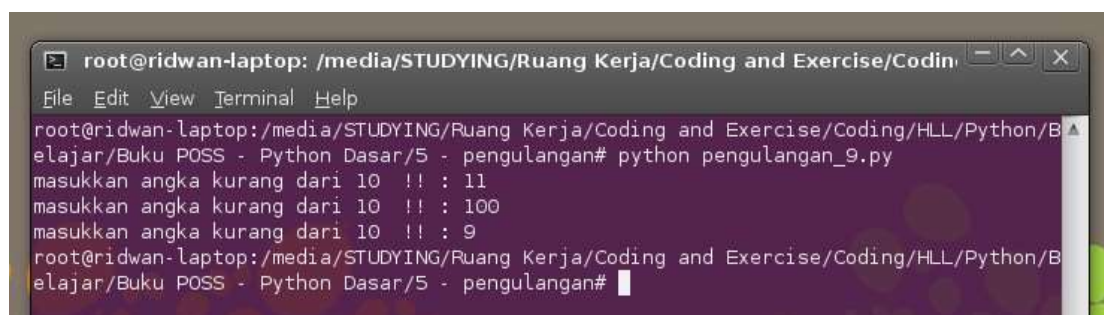
Pada contoh diatas kondisi untuk melakukan pengulangan ditaruh di **while**. Sekarang Anda coba taruh kondisi pengulangan di dalam pengulangannya. Coba lihat contoh berikut :

listing : kode pengulangan_9.py

```

terus_tanya = True while terus_tanya :
    temp = raw_input('masukkan angka kurang dari 10 !! : ') angka = int(temp)
    if angka < 10: terus_tanya = False
    else:
        terus_tanya = True

```



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_9.py
masukkan angka kurang dari 10 !! : 11
masukkan angka kurang dari 10 !! : 100
masukkan angka kurang dari 10 !! : 9
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#

```

<< gambar 5.9 hasil eksekusi pengulangan_9.py >>

Untuk memahami pengulangan **while** lebih lanjut, berikut terdapat contoh penjumlahan angka dari 1 sampai 10. Dalam pengulangan ini terdapat sebuah variabel *jml_angka* yang berfungsi untuk menampung angka – angka yang akan ditambahkan dengan angka berikutnya di setiap

pengulangan. Coba perhatikan kode dibawah ini :

listing : kode pengulangan_10.py

```
i = 1
jml_angka = 0 while i <= 10:
    print 'loop ke - %d : %d + %d' % (i, jml_angka, i) jml_angka = jml_angka + i
    i += 1
print 'total angka yang dijumlahkan : ', jml_angka
```

kode diatas jika dieksekusi akan tampil seperti pada gambar berikut ini :



```
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan# python pengulangan_10.py
loop ke - 1 : 0 + 1
loop ke - 2 : 1 + 2
loop ke - 3 : 3 + 3
loop ke - 4 : 6 + 4
loop ke - 5 : 10 + 5
loop ke - 6 : 15 + 6
loop ke - 7 : 21 + 7
loop ke - 8 : 28 + 8
loop ke - 9 : 36 + 9
loop ke - 10 : 45 + 10
total angka yang dijumlahkan : 55
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/5 - pengulangan#
```

<< gambar 5.10 hasil eksekusi pengulangan_10.py >>

6. Mengenal Data Struktur Python Tingkat Lanjut

Mengenal List, Dictionary dan Tuple

Data yang kompleks biasanya dapat menampung objek atau entitas yang banyak. Misal ada sebuah toples yang berisi banyak permen dengan ragam jenisnya. Dalam sebuah toples dapat saja ada permen rasa asem, permen rasa stroberi, permen rasa nanas, permen rasa asem lagi, dan rasa lainnya. Tapi bukan hanya menyimpan entitas yang banyak saja, struktur data pun ada yang menyimpan jaringan sebuah graf, antrian, tumpukan, dan banyak lagi tipe lainnya.

Di Python sendiri, terdapat beberapa tipe struktur data yang dapat digunakan oleh penggunanya. Tapi tiga tipe data yang umum digunakan, diantaranya **list**, **tuple**, dan **dictionary**. Ketiga tipe data struktur ini sangat membantu sekali bagi programmer Python yang membutuhkan tipe data banyak. Ketiga tipe data ini adalah objek Python yang berarti jika Anda mendefinisikan struktur data tersebut, Anda dapat menggunakan *method – method* yang berhubungan dengan pengolahan struktur data tersebut. Selain itu terdapat pula function untuk mengolah tiga struktur data tadi seperti mencari nilai **max**, **min**, hitung panjang, dan perbandingan isi.

Untuk mendefinisikan sebuah list Anda cukup buat sebuah variabel dengan nama yang diinginkan, kemudian isi variabel tersebut dengan list yang akan dibuat. List diawali dengan tanda '[' dan diakhiri dengan tanda ']'. Isinya dapat beragam, dapat string, number, object, bahkan list lagi. Pemisah antara data yang satu dengan yang lainnya digunakan tanda koma. List dapat ditambah isinya, dirubah data pada elemennya, hapus elemen, dan hapus seluruh list.

Hampir sama dengan list, tuple diawali dengan tanda '(' dan ')'. Elemen pada tuple tidak dapat dihapus dan diubah. Maka dari itu tuple disebut immutable sequence. Lebih jelasnya nanti Anda akan lihat perbedaan tuple dengan list.

Beda halnya antara dengan list dan tuple, pada dictionary setiap data akan memiliki pengenalnya masing – masing. Pengenal tersebut dinamakan dengan key dan datanya dinamakan dengan value. Dictionary diawali dengan tanda '{' dan diakhiri dengan tanda '}'. Khusus untuk key pada dictionary, nilainya harus berupa tipe data yang tidak dapat diganti seperti tuple, string dan number. Tapi umumnya key berisi number dan string. Karena jika Anda mencoba memasukkan tipe data yang mutable, akan keluar peringatan 'unhashable type' saat mendefinisikan dictionary yang key-nya berupa tipe data mutable.

Agar lebih paham mari Anda coba mendefinisikan list, tuple, dan dictionary kedalam sebuah variabel.

listing : strukdat_1.py

```
# cara mendefinisikan list sebuah_list = ['Zorin OS',
```