PERULANGAN

Apa Itu Perulangan?

- Perulangan dalam dunia pemrograman adalah baris kode atau instruksi yang dieksekusi oleh komputer secara berulang-ulang sampai suatu kondisi tertentu terpenuhi. Konsep perulangan ini didukung di semua bahasa pemrograman modern, termasuk di antaranya adalah python.

- Dengan perulangan, kita bisa mengeksekusi suatu kode program berkali-kali dengan jumlah tertentu, atau selama sebuah kondisi tertentu terpenuhi.

- Sebenarnya hampir sama seperti percabangan python

Hanya saja bedanya:

- Kalau percabangan, blok kode yang memenuhi kondisi tertentu hanya akan dieksekusi satu kali saja.
- Sedangkan perulangan, ia akan dilakukan seterusnya berulang-ulang dengan jumlah tertentu atau selama kondisi tertentu terpenuhi.

- Pada python, kita bisa melakukan perulangan dengan beberapa cara di antaranya:

- Perulangan for
- Perulangan while
- Fungsi rekursif

- Perulangan for pada python adalah perintah yang digunakan untuk melakukan iterasi dari sebuah nilai sequence atau data koleksi pada python seperti List, Tuple, String dan lain-lain .

- For pada python memiliki perilaku yang berbeda dengan for pada kebanyakan bahasa pemrograman yang lain, karena pada python ia sangat berkaitan dengan data sequence atau data kolektif. Mungkin kalau dibandingkan dengan bahasa lain, for pada python lebih dikenal sebagai foreach.

Syntax For

Berikut ini adalah struktur sintaks metode for:

```
for nilai in sequence:
  # blok kode for
```

- Jadi, ada 3 bagian penting.

- sequence: adalah sebuah nilai yang bersifat iterable alias bisa diulang-ulang.

- Di antara tipe data yang bersifat sequence atau iterable adalah:
- list
- tuple
- string
- dan lain sebagainya

- nilai: adalah setiap item yang diekstrak dari sequence

- Blok kode: yaitu statemen-statemen atau perintah-perintah tertentu yang akan dieksekusi secara berulang

```
listKota = [
  'Jakarta', 'Surabaya', 'Depok', 'Bekasi', 'Solo',
  'Jogjakarta', 'Semarang', 'Makassar'
]

for kota in listKota:
  print(kota)
```

Jika dieksekusi, program di atas akan menghasilkan output:

```
Jakarta
Surabaya
Depok
Bekasi
Solo
Jogjakarta
Semarang
Makassar
```

Mengetahui urutan iterasi for dengan list

- Untuk mengetahui urutan iterasi for dengan list, kita bisa menggunakan fungsi enumerate.
- Fungsi tersebut akan mengekstrak 2 buah nilai:
- yang pertama adalah index: yaitu urutan iterasi yang ke berapa
- dan item yang mana itu adalah nilai dari list itu sendiri.

```
listKota = [
  'Jakarta', 'Surabaya', 'Depok', 'Bekasi', 'Solo',
  'Jogjakarta', 'Semarang', 'Makassar'
]

for i, kota in enumerate(listKota):
  print(i, kota)
```

- Kode program di atas sama saja seperti sebelumnya, kita hanya menambahkan fungsi enumerate() dan mem-passing variabel listKota sebagai parameter. Kita juga mengekstrak dua buah nilai yang kita kasih nama i dan kota.
- Jila dijalankan, berikut adalah output yang kita dapat:

```
0 Jakarta
1 Surabaya
2 Depok
3 Bekasi
4 Solo
5 Jogjakarta
6 Semarang
7 Makassar
```

For dengan fungsi range()

Selain dengan list, kita juga bisa menggunakan for dengan fungsi range().

Perhatikan contoh berikut:

```
## 0 sampai 4
for i in range(5):
  print("Perulangan ke -", i)
```

Output:

```
Perulangan ke - 0
Perulangan ke - 1
Perulangan ke - 2
Perulangan ke - 3
Perulangan ke - 4
```

- Dengan fungsi range, kita bisa melakukan perulangan dari 0, sampai kurang dari nilai range yang kita definisikan (yaitu 5 dalam contoh di atas). Sehingga hasil perulangan yang didapatkan adalah 0 sampai 4.
- Kita bisa memulai range dari selain 0

```
## 10 sampai 15
for i in range(10, 16):
  print('i =', i)
```

Perulangan di atas akan menghasilkan output:

```
i = 10
i = 11
i = 12
i = 13
i = 14
i = 15
```

- Angka genap

Perhatikan contoh berikut:

```
## Bilangan genap kelipatan 2
for i in range(2, 12, 2):
  print('i =', i)
```

Pada contoh di atas, sistem akan melakukan perulangan dimulai dari angka 2, hingga kurang dari 12 dengan interval/kelipatan sebanyak 2.

Hasilnya:

```
i = 2
i = 4
i = 6
i = 8
i = 10
```

- Angka ganjil

Untuk bilangan ganjil, kita mulai saja dari angka 1:

```
## Bilangan ganjil kelipatan 2
for bilangan_ganjil in range(1, 12, 2):
  print(bilangan_ganjil)
```

Output:

```
1
3
5
7
9
11
```

For dengan tuple

Tuple adalah di antara tipe data yang bersifat *iterable*, sehingga kita juga bisa memperlakukannya sebagai objek perulangan mengguankan `for`.

Perhatikan contoh di bawah:

```
tupleBuah = ('Mangga', 'Jeruk', 'Apel', 'Pepaya')

for buah in tupleBuah:
  print(buah)
```

Output:

```
Mangga
Jeruk
Apel
Pepaya
```

String pun demikian, bersifat **iterable**, sehingga bisa kita jadikan objek perulangan.

Perhatikan contoh berikut:

```
for karakter in "IndonesiaID":
  print(karakter)
```

Jika dijalankan, output-nya:

```
I
n
d
o
n
e
s
i
a
I
D
```

Break dan continue

- Pada python, kita bisa menginterupsi dan juga men-skip suatu iterasi pada perulangan.

- Terdapat 2 perintah yang bisa kita gunakan, yaitu:

- break untuk interupsi (memberhentikan paksa) sebuah perulangan
- continue untuk menskip ke iterasi selanjutnya

```
for i in range(10, 20):
  # skip jika i == 15
  if (i == 15):
    continue

  print(i)
```

Output:

```
10
11
12
13
14
16  <-- Habis 14 langsung 16
17
18
19
```

Perhatikan output di atas, pada saat `i == 15`, perintah `print(i)` tidak dieksekusi dan justru di-skip ke iterasi berikutnya.

Kita justru bisa memberhentikan paksa suatu perulangan sekalipun belum sampai ke iterasi yang terakhir.

```
for i in range(10, 20):
  # hentikan jika i == 15
  if (i == 15):
    break

  print(i)
```

Jika dijalankan:

```
10
11
12
13
14 <-- print terakhir sebelum terjadi break pada i == 15
```

Sistem akan memberhentikan perulangan ketika `i == 15` dan belum sempat melakukan perintah `print()`.

For else

- Hampir mirip dengan if ... else.
- Tapi tugasnya berbeda.
- Perulangan for jika ditambahkan blok else, maka perintah yang ada pada blok else hanya akan dieksekusi ketika perulangan selesai secara natural –tanpa interupsi.

```
listKota = [
  'Jakarta', 'Surabaya', 'Depok', 'Bekasi', 'Solo',
  'Jogjakarta', 'Semarang', 'Makassar'
]

for kota in listKota:
  print(kota)
else:
  print('Tidak ada lagi item yang tersisa')
```

Jika dijalankan, program di atas akan menghasilkan output seperti ini:

```
Jakarta
Surabaya
Depok
Bekasi
Solo
Jogjakarta
Semarang
Makassar
Tidak ada lagi item yang tersisa
```

For … Else + Break

- Jika kita gabungkan for ... else dengan break, maka blok else hanya akan dieksekusi jika perintah break tidak dieksekusi.

- Kita bisa memanfaatkan for ... else + break untuk pencarian sebuah item pada list.

```
listKota = [
  'Jakarta', 'Surabaya', 'Depok', 'Bekasi', 'Solo',
  'Jogjakarta', 'Semarang', 'Makassar'
]

kotaYangDicari = input('Ketik nama kota yang kamu cari: ')

for i, kota in enumerate(listKota):
  # kita ubah katanya ke lowercase agar
  # menjadi case insensitive
  if kota.lower() == kotaYangDicari.lower():
    print('Kota yang anda cari berada pada indeks', i)
    break
else:
  print('Maaf, kota yang anda cari tidak ada')
```

- Program di atas akan meminta user untuk menginputkan nama kota yang ingin dicari. Jika kotanya maka akan kita kasih info indeks-nya berapa (dalam listKota), dan jika tidak ada maka perintah print() yang ada di blok else akan dieksekusi.
- Coba jalankan. Kemudian kita input kata solo, ini hasilnya:

```
Ketik nama kota yang kamu cari: solo
Kota yang anda cari berada pada indeks 4
```

Jika kita cari pakai kota yang tidak ada di dalam list, begini hasilnya:

```
Ketik nama kota yang kamu cari: pontianak
Maaf, kota yang anda cari tidak ada
```

- Nah, harusnya sekarang sudah lebih jelas bagaimana cara for ... else bekerja, dan kapan blok kode else akan dieksekusi. Dia hanya akan dieksekusi ketika perulangan mencapai titik akhirnya (alias sudah tidak ada iterasi lagi yang tersisa).

- Ada pun jika sebuah perulangan for dihentikan paksa dengan perintah break, maka perintah yang ada pada blok else tidak akan dieksekusi.