DASAR PYTHON

BAB 6 : WHILE LOOPS & FOR LOOPS

6.1 Tujuan

Mahasiswa mengenal materi While Loops dan For Loops pada bahasa pemrograman python

6.2 Ulasan Materi

A. Python While Loops

Python memiliki dua perintah loop primitif yaitu while loops dan for loops.

1. **While Loop (Perulangan While)**

   Dengan while loop kita dapat mengeksekusi satu set statements selama kondisinya true. Contoh, print i selama i kurang dari 6 :

```
i = 1
while i < 6:
  print(i)
  i += 1
```

   Output :

```
1
2
3
4
5
```

   While Loop membutuhkan variabel yang relevan untuk ready, dalam contoh ini kita perlu mendefinisikan variabel pengindeksan, i, yang kita set ke 1.

2. **The Break Statement (Pernyataan break)**

   Dengan pernyataan break kita dapat menghentikan perulangan meskipun kondisi while true. Contoh, keluar dari loop ketika i adalah 3 :

```
i = 1
while i < 6:
  print(i)
  if (i == 3):
    break
  i += 1
```

Output :

```
1
2
3
```

3. The continue Statement (Pernyataan Lanjutan)

Dengan pernyataan continue kita dapat menghentikan iterasi saat ini, dan melanjutkan dengan yang berikutnya.

Contoh, lanjutkan ke iterasi berikutnya jika i adalah 3 :

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Perhatikan bahwa nomor 3 tidak ada dalam hasil

Output :

```
1
2
4
5
6
```

4. **The else Statement**

Dengan pernyataan else kita dapat menjalankan blok kode satu kali ketika kondisinya tidak lagi true. Contoh, print pesan setelah kondisinya false :

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Output :

```
1
2
3
4
5
i is no longer less than 6
```

B. Python For Loops

1. **Python For Loops**

Perulangan for digunakan untuk mengulangi urutan (yaitu list, tupel, dictionary, set, atau string). Ini kurang seperti kata kunci for dalam bahasa pemrograman lain, dan bekerja lebih seperti method iterator seperti yang ditemukan dalam bahasa pemrograman berorientasi objek lainnya.

Dengan for loop kita dapat mengeksekusi satu set pernyataan, sekali untuk setiap item dalam list, tuple, set dll. Contoh print setiap fruits dalam list fruits :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Output :

```
apple
banana
cherry
```

Perulangan for tidak memerlukan variabel pengindeksan untuk disetel sebelumnya.

2. **Looping Through a String (Looping Melalui String)**

   Bahkan string adalah objek yang dapat diubah, mereka berisi urutan karakter

   Contoh loop huruf-huruf dalam kata "banana".

```
for x in "banana":
  print(x)
```

Output :

```
b
a
n
a
n
a
```

3. **The break Statement**

   Dengan pernyataan break kita dapat menghentikan loop sebelum loop melewati semua item. Contoh, keluar dari loop ketika x “banana” :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Output :

```
apple
banana
```

Contoh, keluar dari loop ketika x “banana”, tetapi kali ini jeda dating sebelum print :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Output :

```
apple
```

4. The Continue Statement (Pernyataan Lanjutan)

Dengan pernyataan continue kita dapat menghentikan iterasi loop saat ini, dan melanjutkan dengan yang berikutnya. Contoh, jangan mencetak banana :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Output :

```
apple
cherry
```

5. The Range() Function (Fungsi range())

Untuk mengulang satu set kode beberapa kali, kita dapat menggunakan fungsi range().Fungsi range() mengembalikan urutan angka, mulai dari 0 secara default, dan bertambah 1 (secara default), dan berakhir pada angka yang ditentukan.

Contoh, menggunakan fungsi range() :

```
for x in range(6):
  print(x)
```

Output :

```
0
1
2
3
4
5
```

Fungsi range() default ke 0 sebagai nilai awal, namun dimungkinkan untuk menentukan nilai awal dengan menambahkan parameter: range(2, 6) , yang berarti nilai dari 2 hingga 6 (tetapi tidak termasuk 6).

Contoh, menggunakan parameter mulai :

```
for x in range(2, 6):
  print(x)
```

Output :

```
2
3
4
5
```

Fungsi range() default untuk menambah urutan dengan 1, namun dimungkinkan untuk menentukan nilai kenaikan dengan menambahkan parameter ketiga: range(2, 30, 3). Contoh, tingkatkan urutan dengan 3 (default adalah 1) :

```
for x in range(2, 30, 3):
  print(x)
```

Output :

```
2
5
8
11
14
17
20
23
26
29
```

6. Else di For Loop

Kata kunci else dalam for loop menentukan blok kode yang akan dieksekusi ketika loop selesai. Contoh, print semua angka dari 0 hingga 5, dan cetak pesan saat loop telah berakhir :

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Output :

```
0
1
2
```

```
3
4
5
Finally finished!
```

7. **Else di For Loop (Loop Bersarang)**

   Loop bersarang adalah loop di dalam loop. "inner loop" akan dieksekusi satu kali untuk setiap iterasi dari "outer loop". Contoh, print setiap kata sifat untuk setiap fruit :

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

   Output :

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

8. **The Pass Statement (Pernyataan Pass)**

   For loop tidak boleh kosong, tetapi jika karena alasan tertentu memiliki for loop tanpa konten, masukkan pernyataan pass untuk menghindari error. Contoh :

```
for x in [0, 1, 2]:
  pass
```

memiliki for loop kosong seperti ini, akan menimbulkan kesalahan tanpa pernyataan pass

Output :

6.3 Praktikum

Contoh : While Loop

While merupakan perulangan dimana mengecek kondisi terlebih dahulu, apabila terpenuhi maka perulangan akan berjalan, sampai selesai. Untuk implementasinya akan diuraikan pada poin dibawah ini.

1. Inisialisasikan variabel I dengan bertipe integer dan memiliki nilai 1

```
i = 1
```

2. Tambahkan perulangan while dengan menambahkan operator perbandingan kurang dari 6. Penjelasan dari sintaks ini mendeskripsikan tetap dijalankan apabila variabel i kurang dari 6.

```
while i < 6:
```

3. Didalam while tambahkan output menggunakan perintah print dengan memiliki nilai i

```
  print(i)
```

4. Tambahkan perintah increment untuk variabel i. Perintah increment digunakan untuk menambahkan nilai secara kontinyu selama perulangan berjalan. Penambahan nilai memiliki pola +1

```
  i += 1
```

5. Untuk full source code dilampirkan sebagai berikut :

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Percobaan 1 : While Loop

Sekarang kerjakan percobaan 1 dengan mengikuti langkah-langkah dibawah ini !

1. Buatlah variabel dengan nama ***i*** yang memiliki nilai integer 6
2. Buatlah perulangan while untuk menampilkan angka 6 sampai 1 secara terbalik, dengan menambahkan perulangan while dan menambahkan operator perbandingan lebih dari sama dengan 1
3. Didalam while tampilkan output variabel I menggunakan fungsi print
4. Tambahkan perintah pengurangan untuk variabel i. Perintah pengurangan digunakan untuk mengurangi nilai secara kontinyu selama perulangan berjalan. Pengurangan nilai memiliki pola -=
5. Jalankan kode program percobaan 1 diatas dengan menekan tombol Check Code Validity