

## 5. Struktur Kontrol Perulangan

### 5.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menggunakan bentuk perulangan for maupun while.
2. Dapat menggunakan break dan continue sesuai dengan permasalahan yang dihadapi.
3. Dapat melakukan konversi dari bentuk for menjadi bentuk while
4. Dapat menerapkan perulangan dalam menyelesaikan permasalahan.

### 5.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

### 5.3 Materi

#### 5.3.1 Definisi Perulangan

Jalannya suatu program dapat diatur secara sekuensial, percabangan, perulangan, maupun kombinasi dari ketiganya. Pengaturan tersebut biasa disebut sebagai struktur kontrol. Perulangan digunakan apabila dalam program diperlukan untuk:

- Melakukan suatu hal yang sama beberapa kali.

- Melakukan suatu hal secara bertahap, di mana setiap tahap sebenarnya memiliki langkah yang sama.
- Mengakses sekumpulan data dalam suatu struktur data, misalnya: List, Tuple, Queue, Stack dan beberapa struktur data lainnya.

Pada Python, perulangan dapat dilakukan dengan menggunakan **for**, **while** maupun dengan cara **rekursif**. Untuk pertemuan ini, dibahas mengenai perulangan for dan while.

### 5.3.2 Bentuk Perulangan for

Pada Python, perulangan dapat dinyatakan dalam bentuk **for** dan **while**. Perulangan **for** biasanya digunakan pada kondisi:

- **Jumlah perulangan sudah diketahui sejak awal.** Misalnya akan dilakukan pembacaan data dari 10 file teks. Walaupun setiap file teks memiliki isi yang berbeda, tetapi membaca file teks secara umum tetap sama. Pembacaan akan dilakukan dari file pertama, kedua, ketiga, dan seterusnya sampai file ke-sepuluh.
- **Perulangan terjadi karena operasi yang sama pada suatu rentang data atau rentang nilai.** Misalnya dalam mencari jumlah dari 100 bilangan pertama, maka secara berturut-turut dilakukan penjumlahan  $1 + 2 + 3 + \dots$  <berulang-ulang>  $+ 100$ . Berarti dilakukan dalam rentang mulai dari 1 sampai 100.

Perulangan for pada rentang tertentu lebih mudah dilakukan dengan menggunakan bantuan fungsi `range()`, yang bentuknya sebagai berikut:

- `range(stop)`. Digunakan untuk menghasilkan rentang dari **0** sampai **stop-1**. Misalnya `range(6)`, berarti menghasilkan rentang 0-5.
- `range(start, stop, [step])`. Digunakan untuk menghasilkan rentang dari **start**, sampai **stop** dengan peningkatan sejumlah **step**.

Berikut ini adalah contoh program untuk menampilkan bilangan dari 1 sampai 100 dengan menggunakan for dan `range()`:

```
for i in range(1, 101):
    print(i)
```

Pada program tersebut, ada perulangan for dengan menggunakan `range()`, dimulai dari 1 (start) sampai 101 (stop-1) dengan langkah 1 (default step adalah 1). Kemudian variabel `i` digunakan sebagai counter, di mana nilai `i` akan naik secara berurutan sesuai dengan nilai yang dihasilkan dari fungsi `range()` tersebut. Untuk kasus di mana tidak diperlukan counter, maka bentuk perulangan bisa seperti berikut ini:

```
for _ in range(1, 101):
    print('Hello World')
```

Program tersebut akan menampilkan tulisan Hello World sebanyak 100 kali, yang tidak membutuhkan nilai dari suatu counter.

#### Step negatif

Perhatikan perulangan for berikut ini yang akan menampilkan seluruh bilangan genap dari 2 sampai 100:

```
for i in range(2, 101, 2):
    print(i)
```

Perulangan dilakukan pada rentang 2-100, dengan langkah 2. Maka rentang yang dipakai adalah 2, 4, 6, 8, 10, 12, ..., 100. Bagaimana jika diperlukan untuk menampilkan bilangan genap dari 100 sampai 2? Fungsi `range()` bisa menerima step negatif, seperti pada contoh berikut ini:

```
for i in range(100, 1, -2):  
    print(i)
```

Program tersebut akan menampilkan bilangan genap mulai dari 100, 98, 96, 94, ..., sampai 2.

### 5.3.3 Bentuk Perulangan While

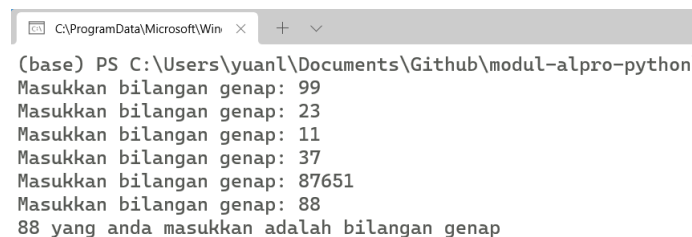
Bentuk while biasanya digunakan pada kondisi di mana jumlah perulangan belum diketahui sebelumnya. Bentuk perulangan while secara umum adalah sebagai berikut:

```
<start>  
while <stop>:  
    operation  
    operation  
    <step (optional)>
```

Berikut ini adalah contoh perulangan dengan menggunakan while yang digunakan untuk memastikan input yang dimasukkan oleh pengguna adalah bilangan genap:

```
1 bilangan = 0  
2 genap = False  
3 while genap == False:  
4     bilangan = int(input('Masukkan bilangan genap: '))  
5     if bilangan % 2 == 0:  
6         genap = True  
7 print(bilangan, 'yang anda masukkan adalah bilangan genap')
```

Output yang dihasilkan oleh program tersebut dapat dilihat pada Gambar 5.1. Pengguna awalnya memasukkan bilangan ganjil, tetapi program terus meminta pengguna memasukkan bilangan genap. Program berhenti setelah pengguna memasukkan 88, yang merupakan bilangan genap. Kasus ini sangat sesuai jika menggunakan perulangan while, karena tidak diketahui sampai berapa kali pengguna memasukkan bilangan ganjil yang tidak sesuai dengan permintaan.



```
C:\ProgramData\Microsoft\Win  
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python  
Masukkan bilangan genap: 99  
Masukkan bilangan genap: 23  
Masukkan bilangan genap: 11  
Masukkan bilangan genap: 37  
Masukkan bilangan genap: 87651  
Masukkan bilangan genap: 88  
88 yang anda masukkan adalah bilangan genap
```

Gambar 5.1: Penggunaan while untuk mengambil input dari pengguna sampai sesuai dengan permintaan.

### 5.3.4 Penggunaan Break dan Continue

Perulangan dapat dikontrol dengan menggunakan **break** dan **continue**. Secara umum break digunakan untuk menghentikan perulangan, sedangkan continue digunakan untuk melanjutkan perulangan ke iterasi berikutnya. Perhatikan program berikut ini yang akan menampilkan bilangan dari 1 sampai 10:

```
for i in range(1, 11):  
    print(i)  
print('Selesai')
```

Program tersebut akan menampilkan bilangan 1 sampai 10, kemudian pada baris terakhir muncul tulisan 'Selesai'. Jika diinginkan perulangan yang seharusnya sampai 10, dihentikan saat mencapai 5, maka diperlukan break dengan kondisi seperti program berikut ini:

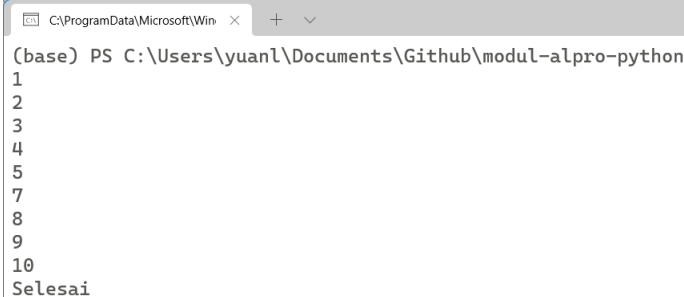
```
for i in range(1, 11):
    if i == 5:
        break
    else:
        print(i)
print('Selesai')
```

Pada saat nilai  $i=5$ , maka boolean expression dari  $i==5$  akan menghasilkan nilai True, program akan menjalankan instruksi break. Maka selanjutnya perulangan akan dihentikan dan program berlanjut pada baris setelah perulangan tersebut, yaitu menampilkan tulisan 'Selesai'.

Apa perbedaan antara break dan continue? Continue digunakan pada saat diperlukan untuk melewati tahap perulangan sekarang, langsung dilanjutkan ke tahap/iterasi perulangan berikutnya. Sebagai contoh, berikut ini adalah program yang menampilkan angka dari 1 sampai 10, tetapi diperlukan untuk tidak menampilkan angka 6:

```
for i in range(1, 11):
    if i == 6:
        continue
    else:
        print(i)
print('Selesai')
```

Output yang dihasilkan dari contoh penggunaan continue dapat dilihat pada Gambar 5.2. Program menampilkan angka dari 1 sampai 10, tetapi melewati angka 6.



```
C:\ProgramData\Microsoft\Win...
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python
1
2
3
4
5
7
8
9
10
Selesai
```

Gambar 5.2: Penggunaan continue untuk melewati iterasi saat counter(i) bernilai 6.

### 5.3.5 Konversi dari Bentuk for Menjadi Bentuk while

Bentuk perulangan for sebagian besar dapat dikonversi menjadi bentuk while. Beberapa hal yang ada di bentuk for dan while adalah sebagai berikut:

- Harus ada nilai awal, untuk memulai perulangan.
  - Harus ada nilai akhir, untuk mengakhiri perulangan.
  - Harus ada langkah, agar iterasi dari nilai awal bisa terus berjalan sampai mencapai nilai akhir.
- Misalkan ada perulangan for seperti berikut:

```
for i in range(1, 11):
    print(i)
```

maka dapat diidentifikasi bahwa perulangan tersebut dimulai dari 1, berakhir di 10, dengan step adalah 1. Konversi ke bentuk while dapat dilakukan dengan mudah, menghasilkan perulangan while berikut ini yang menghasilkan output yang sama:

```
i = 1           //awal
while i <= 10:  //kondisi akhir
    print(i)
    i = i + 1   //step
```

## 5.4 Kegiatan Praktikum

Kegiatan praktikum pada pertemuan ini adalah sebagai berikut:

- Menggunakan perulangan untuk menampilkan deret bilangan.
- Menggunakan break pada perulangan while.

### 5.4.1 Deret Bilangan

■ **Contoh 5.1** Buatlah program untuk menampilkan deret bilangan fibonacci mulai dari 1 sampai batas tertentu yang dimasukkan oleh pengguna!

Deret bilangan fibonacci adalah deret bilangan yang tersusun dari penjumlahan dua suku sebelumnya dari deret bilangan tersebut. Biasanya deret bilangan fibonacci dimulai dari 1, 1, 2, 3, ... dan seterusnya. Ilustrasinya dapat dilihat pada Gambar 5.3.

1   1   2   3   5   8   13   21   34   ...  
                  └─┬─┘  
                  13 + 21 = 34

Gambar 5.3: Deret bilangan fibonacci.

Implementasi dari deret fibonacci lebih baik jika dipisahkan dalam bentuk fungsi fibo() yang menerima parameter batas. Langkah-langkah dari implementasi tersebut adalah:

1. Minta nilai batas dari pengguna.
2. Panggil fungsi fibo() untuk menampilkan deret fibonacci dengan menggunakan perulangan while.

Program untuk menjawab permasalahan tersebut adalah sebagai berikut:

```
def fibo(batas):
    bil1 = 1
    bil2 = 1
    # tampilkan dua suku fibonacci pertama
    if bil1 < batas:
        print(bil1, end='\t')
        print(bil2, end='\t')
    # suku-suku berikutnya dari bil1 + bil2
    suku_baru = bil1 + bil2
    while suku_baru < batas:
        print(suku_baru, end='\t')
        # geser bil1 dan bil2
```

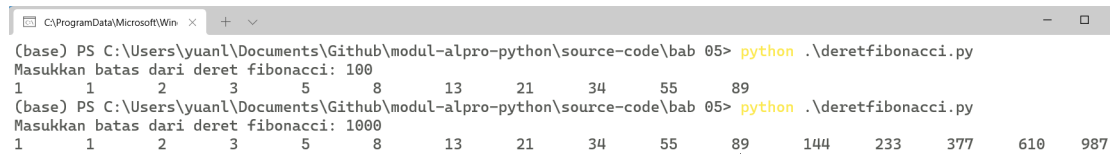
```

bil1 = bil2
bil2 = suku_baru
# hitung lagi suku berikutnya
suku_baru = bil1 + bil2

# program utama
batas = int(input('Masukkan batas dari deret fibonacci: '))
fibonacci(batas)

```

Jika program tersebut dijalankan, hasilnya akan seperti pada Gambar 5.4.



```

(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python\source-code\bab 05> python .\deretfibonacci.py
Masukkan batas dari deret fibonacci: 100
1      1      2      3      5      8      13      21      34      55      89
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python\source-code\bab 05> python .\deretfibonacci.py
Masukkan batas dari deret fibonacci: 1000
1      1      2      3      5      8      13      21      34      55      89      144      233      377      610      987

```

Gambar 5.4: Deret bilangan fibonacci yang kurang dari 100 dan kurang dari 1000.

■ **Contoh 5.2** Buatlah program yang dapat menampilkan deret bilangan konvergen, yang diawali dari input bilangan dari pengguna. Suku-suku dari deret tersebut berikutnya didapatkan dengan cara berikut ini:

- Jika ganjil, maka kalikan dengan tiga, lalu tambah 1.
- Jika genap, bagi dengan 2.

Tampilkan suku-suku dari deret bilangan konvergen tersebut yang berakhir jika mencapai nilai 1. ■

Untuk menyelesaikan permasalahan ini, diperlukan pemahaman terlebih dahulu mengenai deret bilangan konvergen tersebut. Sebagai contoh bila input dari pengguna adalah 5, maka deret bilangan yang dihasilkan adalah: 5, 16, 8, 4, 2, 1. Bagaimana jika pengguna memberi input 12? Deret yang dihasilkan adalah: 12, 6, 3, 10, 5, 16, 8, 4, 2, 1. Dari contoh tersebut, apakah dapat diketahui berapa jumlah suku yang akan ditampilkan? Jika jumlah suku tidak diketahui dari awal, maka permasalahan ini lebih cocok jika diselesaikan menggunakan perulangan bentuk while.

Untuk menyusun perulangan bentuk while, diperlukan beberapa bagian berikut:

- Awal adalah input bilangan dari pengguna.
- Akhir adalah 1. Deret bilangan berakhir jika sudah mencapai nilai 1.
- Step dilakukan sesuai dengan suku sekarang, apakah genap atau ganjil.

Setelah mengetahui awal, akhir dan step, maka program untuk penyelesaian masalah tersebut adalah sebagai berikut:

```

def konvergen(start):
    suku = start
    while suku != 1:
        print(suku)
        if suku % 2 == 0:
            suku = suku // 2
        else:
            suku = suku * 3 + 1

# bagian utama program
start = int(input('Masukkan suku pertama dari deret konvergen: '))
konvergen(start)

```

Jika program tersebut dijalankan, hasilnya dapat dilihat pada Gambar 5.5.



```
C:\ProgramData\Microsoft\Win...
(base) PS C:\Users\yuan\Documents\Github\modul-alpro-python\source-code\bab 05> py
Masukkan suku pertama dari deret konvergen: 12
12
6
3
10
5
16
8
4
2
(base) PS C:\Users\yuan\Documents\Github\modul-alpro-python\source-code\bab 05> py
Masukkan suku pertama dari deret konvergen: 5
5
16
8
4
2
(base) PS C:\Users\yuan\Documents\Github\modul-alpro-python\source-code\bab 05> |
```

Gambar 5.5: Deret bilangan konvergen yang dimulai dari 12 dan 5.

### 5.4.2 Penggunaan Break

■ **Contoh 5.3** Buatlah program yang dapat menghitung rata-rata dari sejumlah input yang diberikan oleh pengguna. Program akan terus meminta input dari pengguna, sampai pengguna memasukkan bilangan negatif atau nol. Program kemudian menampilkan rata-rata dari keseluruhan input (abaikan input negatif atau nol). ■

Untuk menyelesaikan masalah ini, perlu diketahui terlebih dahulu bahwa program akan meminta input dari pengguna terus-menerus sampai pengguna memasukkan nilai negatif atau nol. Apakah dari awal sudah diketahui berapa kali pengguna memasukkan input sampai akhirnya memasukkan nilai negatif atau nol? Jawabannya adalah tidak diketahui. Oleh karena itu, untuk penyelesaian masalah ini akan menggunakan perulangan bentuk while. Untuk mengakhiri permintaan input dari pengguna, dapat dilakukan dengan penggunaan break, dengan kondisi input adalah negatif atau nol. Penyelesaian dari masalah tersebut adalah sebagai berikut:

```
def average():
    total = 0
    count = 0
    while True:
        input_user = int(input('Masukkan nilai (nol atau negatif untuk berhenti): '))
        if input_user < 1: # negatif atau nol
            break
        else:
            total = total + input_user
            count = count + 1
    if count > 0:
        return total / count
    else:
        return 0

# bagian utama program
hasil = average()
print('Rata-rata: ', hasil)
```

Contoh output dari program tersebut dapat dilihat pada Gambar 5.6.

Dari output yang dihasilkan terlihat bahwa program berhenti meminta input pengguna saat mendapatkan input bilangan negatif atau nol.

```

C:\ProgramData\Microsoft\Win
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python\source-code\bab 05>
Masukkan nilai (nol atau negatif untuk berhenti): 18
Masukkan nilai (nol atau negatif untuk berhenti): 12
Masukkan nilai (nol atau negatif untuk berhenti): 30
Masukkan nilai (nol atau negatif untuk berhenti): -3
Rata-rata: 20.0
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python\source-code\bab 05>
Masukkan nilai (nol atau negatif untuk berhenti): 70
Masukkan nilai (nol atau negatif untuk berhenti): 40
Masukkan nilai (nol atau negatif untuk berhenti): 100
Masukkan nilai (nol atau negatif untuk berhenti): 55
Masukkan nilai (nol atau negatif untuk berhenti): 82
Masukkan nilai (nol atau negatif untuk berhenti): 0
Rata-rata: 69.4
(base) PS C:\Users\yuanl\Documents\Github\modul-alpro-python\source-code\bab 05>

```

Gambar 5.6: Penggunaan break untuk menghentikan perulangan while.

## 5.5 Latihan Mandiri

**Latihan 5.1** Buatlah program yang menerapkan perhitungan perkalian dengan menggunakan penjumlahan. Buatlah fungsi perkalian() dalam program tersebut! Berikut ini adalah beberapa contoh perhitungan yang diharapkan:

- $6 \times 5 = 5 + 5 + 5 + 5 + 5 + 5 = 30$ .
- $7 \times 10 = 10 + 10 + 10 + 10 + 10 + 10 + 10 = 70$ .

**Latihan 5.2** Buatlah program yang dapat menampilkan deret bilangan ganjil dari batas bawah dan batas atas yang diberikan oleh pengguna. Jika ternyata batas atas < batas bawah, berarti deret tersebut dimulai dari batas atas, sampai batas bawah (negatif range). Buatlah fungsi ganjil() dalam program tersebut! Berikut ini adalah contoh hasil yang diharapkan:

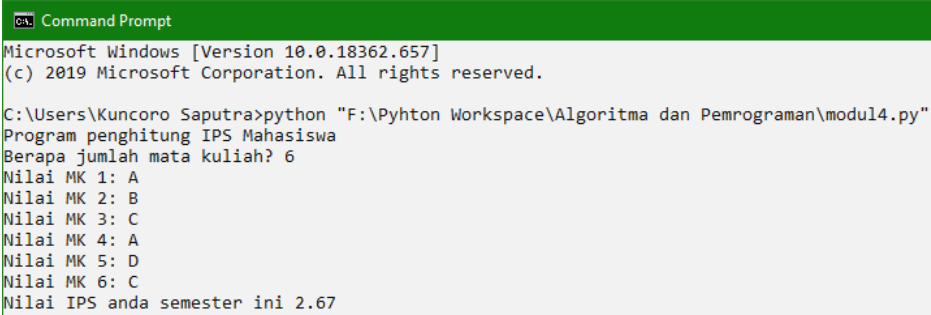
- bawah = 10, atas = 30. Karena bawah < atas, berarti dari kecil ke besar, maka hasilnya adalah: 11, 13, 15, 17, 19, 21, 23, 25, 27, 29.
- bawah = 97, atas = 82. Karena bawah > atas, berarti dari besar ke kecil, maka hasilnya adalah: 97, 95, 93, 91, 89, 87, 85, 83.

**Latihan 5.3** Buatlah sebuah program penghitung nilai Indeks Prestasi Semester (IPS). Input bagi program:

- Jumlah mata kuliah
- Nilai A, B, C, dan D untuk setiap mata kuliah mahasiswa. Diasumsikan sks setiap mata kuliah selalu 3. Kemudian bobot dari masing-masing nilai adalah: A=4, B=3, C=2, D=1.

Output program ialah hasil IPS yang didapatkan. Jalannya program seperti pada Gambar 5.7. Tips: Gunakan kontrol percabangan di dalam perulangan.





```
Command Prompt
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Kuncoro Saputra>python "F:\Pyhton Workspace\Algoritma dan Pemrograman\modul4.py"
Program penghitung IPS Mahasiswa
Berapa jumlah mata kuliah? 6
Nilai MK 1: A
Nilai MK 2: B
Nilai MK 3: C
Nilai MK 4: A
Nilai MK 5: D
Nilai MK 6: C
Nilai IPS anda semester ini 2.67
```

Gambar 5.7: Hasil luaran program IPS mahasiswa