



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230970
Nama Lengkap	Gregorius Daniel Jodan Perminas
Minggu ke / Materi	05 / Struktur Kontrol Perulangan

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Definisi Perulangan

Dalam pemrograman kita akan mengenal beberapa cara untuk mengatur jalannya program:

1. Sekuensial = Program dieksekusi secara berurutan
2. Percabangan = Program dieksekusi berdasarkan kondisi tertentu
3. **Perulangan = Program atau baris kode dieksekusi beberapa kali sampai kondisi terpenuhi**
4. Kombinasi dari ketiganya

Manfaat perulangan dalam Python:

1. Perulangan, membuat kita dapat melakukan hal yang sama beberapa kali dengan lebih efisien dan menghindari penulisan kode berulang.
2. Melakukan suatu hal secara bertahap, di mana setiap tahap sebenarnya memiliki langkah yang sama.
3. Mengakses sekumpulan data dalam suatu struktur data, misalnya: List, Tuple, Queue, Stack dan beberapa struktur data lainnya.

Bentuk Perulangan For

Jenis perulangan ini dapat digunakan ketika:

1. Jumlah perulangan yang terhitung jelas.
2. Perulangan terjadi karena operasi yang sama pada suatu rentang data atau rentang nilai.

Perulangan for pada rentang tertentu seperti no 2 di atas akan 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 yang ditentukan, sampai stop dengan jumlah peningkatan yang sesuai dengan step yang diminta.

Syntax Perulangan For

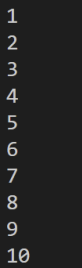
```
1  for nilai in sequence:
2      # blok kode for
```

Gambar 1 - Syntax For Dalam Python

Step Positif

Berikut adalah contoh program yang menggunakan *for* dan *range()*:

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



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Gambar 2 - Perulangan For Dengan Nilai Counter

```
1 for _ in range (1,11):  
2     print("Hello World")
```



```
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World  
Hello World
```


Gambar 3 - Perulangan For Tanpa Nilai Counter

Pada Gambar 1, ada perulangan *for* dengan menggunakan *range()*, dimulai dari 1 sampai 11 (STOP-1) dengan langkah +1, karena langkah defaultnya 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. Gambar 2 adalah contoh kasus dimana counter tidak diperlukan, program tersebut akan menampilkan "Hello World" sebanyak 10 kali.

Step Negatif

Bagaimana caranya ketika kita menemukan kasus untuk menampilkan bilangan genap dari angka 10 sampai 2. Untungnya fungsi *range()* pada Python dapat menerima step negatif.

```
1 for i in range (10,1, -2):  
2     print(i)
```



```
10  
8  
6  
4  
2
```

Gambar 4 - Perulangan For Dengan Step Negatif

Program di atas berfungsi untuk menampilkan angka pada rentang 10-1 (pada step negatif: STOP+1), dengan langkah -2.

Bentuk Perulangan While

Jenis perulangan ini biasanya digunakan ketika jumlah perulangan tidak diketahui (*indefinite*). Perulangan *while* pada Python adalah proses pengulangan suatu blok kode program selama sebuah kondisi terpenuhi.

Syntax Perulangan While

```
1 while <kondisi>:  
2     # blok kode yang akan diulang
```

Gambar 5 - Syntax While Dalam Python

Perulangan Tanpa Batas

Perulangan *while* sangat berkaitan dengan variabel boolean, atau logical statement. Karena penentuan kapan suatu blok kode akan diulang-ulang ditinjau dari True or False dari suatu pernyataan logika. Sehingga jika suatu kondisi itu selalu benar, maka perulangannya pun akan selalu di eksekusi.

```
1 while 3 == 1 + 2:  
2     print("Hello World!")
```

PROBLEMS OUTPUT SEARCH ERROR TERMINAL DEBUG CONSOLE PORTS COMMENTS

```
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!
```

Gambar 6 - Perulangan While Tanpa Batas

Output dari program pada Gambar 6 di atas adalah mencetak "Hello World" terus-menerus tanpa henti. Hal ini karena kondisi $3 = 1 + 2$ bernilai selalu benar. Untuk memaksa program berhenti kita bisa menekan tombol Ctrl + C jika menggunakan CLI.

Perulangan Dengan Batas

```
1 <start>  
2 while <stop>:  
3     # blok kode yang akan diulang  
4     # <step (optional)>
```

Gambar 7 - Syntax Perulangan While Dengan Batas

Di bawah ini merupakan contoh perulangan dengan *while* yang digunakan untuk memastikan input pengguna adalah bilangan genap:

```

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

```

```

Masukkan bilangan genap dong: 57
Masukkan bilangan genap dong: 89
Masukkan bilangan genap dong: 98675
Masukkan bilangan genap dong: 76
76 yang anda masukkan merupakan bilangan genap

```

Gambar 8 - Perulangan While Dengan Batas

Pengguna awalnya memasukkan bilangan ganjil, tetapi program akan terus meminta pengguna memasukkan bilangan genap. Program akan berhenti setelah pengguna memasukkan 88, yang merupakan bilangan genap. Kasus seperti ini sangat sesuai untuk perulangan while, karena kita tidak tahu sampai berapa kali pengguna memasukkan bilangan ganjil yang tidak sesuai dengan permintaan.

Konversi Bentuk For menjadi Bentuk While

Hampir semua bentuk perulangan for dapat dikonversi menjadi bentuk while. Namun tidak berlaku sebaliknya. Perulangan while berkaitan dengan pengecekan kondisi True atau False, sehingga range/batasan tidak selalu diperlukan dalam perulangan jenis ini. Beberapa kesamaan yang ada di bentuk for dan while adalah sebagai berikut:

- Ada nilai awal, untuk memulai perulangan.
- Ada nilai akhir, untuk mengakhiri perulangan.
- Ada langkah, agar iterasi dari nilai awal bisa terus berjalan sampai mencapai nilai akhir.

Misalkan ada perulangan for seperti berikut:

```

1  for i in range(1, 6):
2      print(i)

```

```

1
2
3
4
5

```


Gambar 9 - Perulangan For

Maka dapat diidentifikasi bahwa perulangan tersebut dimulai dari 1, berakhir di 6 (STOP-1 = 5) dengan step adalah +1. Jika dilakukan konversi maka, akan didapat perulangan bentuk while berikut ini dengan hasil output yang sama:

```

1  i = 1          #Start
2  while i <= 5:  #Stop
3      print(i)   #Operation
4      i = i + 1  #Step

```



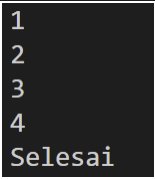
```
1
2
3
4
5
```

Gambar 10 - Konversi Perulangan For Menjadi While

Penggunaan Break dan Continue

Aliran loop dapat diubah dari prosedur standarnya menggunakan break dan continue. Statement break berfungsi untuk keluar dari perulangan secara paksa. Ketika terdapat statement break maka baris selanjutnya tidak akan dijalankan dan perulangan dinyatakan selesai. Berikut adalah contoh program dengan statement break:

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



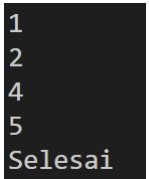
```
1
2
3
4
Selesai
```

Gambar 11 - Statement Break Pada Python

Program tersebut seharusnya menampilkan bilangan 1 sampai 10, namun karena ada statement break ketika `i == 5`, maka program akan dipaksa berhenti ketika `i` sudah mencapai angka 5.

Kemudian untuk statement continue digunakan untuk melompat ke iterasi selanjutnya dan melewati tahap perulangan sekarang. Berikut ini contoh programnya:

```
1  for i in range(1, 6):
2      if i == 3:
3          continue
4      else:
5          print(i)
6  print('Selesai')
```



```
1
2
4
5
Selesai
```

Gambar 12 - Statement Continue Pada Python

Program diatas seharusnya menampilkan bilangan 1-5, namun karena terdapat statement continue ketika `i == 3`, maka iterasi tersebut dilompati dan dilanjutkan ke kode berikutnya hingga program selesai dijalankan.

BAGIAN 2: LATIHAN MANDIRI (60%)

LINK GITHUB: https://github.com/danieljodan/PrakAIPro5_A_71230970.git

SOAL 1

```
1 def perkalian(a,b):
2     total = 0
3     hasilPerkalian = a*b
4     print(f"{a} x {b} =", end=" ") # Digunakan untuk menampilkan hasil dalam satu baris
5     while total < hasilPerkalian: # Perulangan yang berlangsung selama total belum sama dengan hasil perkalian di atas
6         total += b # Perulangan dilakukan dengan menjumlahkan nilai disebelah kanan tanda perkalian
7         if hasilPerkalian == total:
8             print(f"{b} =", end=" ") # Jika hasilPerkalian sama dengan total maka tampilkan tanda "=" di sebelah angka
9         else:
10            print(f"{b} +", end=" ") # Jika belum sama lanjutkan dengan menampilkan tanda "+" di sebelah angka
11    print (f"{total}")
12
13 masukan1 = int(input("Masukkan Bilangan Pertama : "))
14 masukan2 = int(input("Masukkan Bilangan Kedua: "))
15 perkalian(masukan1, masukan2)
```

```
Masukkan Bilangan Pertama : 6
Masukkan Bilangan Kedua: 5
6 x 5 = 5 + 5 + 5 + 5 + 5 + 5 = 30
```

```
Masukkan Bilangan Pertama : 7
Masukkan Bilangan Kedua: 10
7 x 10 = 10 + 10 + 10 + 10 + 10 + 10 + 10 = 70
```

Gambar 13 - Program Latihan Mandiri 5.1

Di atas adalah program perkalian yang akan menerapkan perhitungan perkalian dengan melakukan penjumlahan berulang. Penjelasan kode:

Baris 1: Pendefinisian function dengan nama perkalian()

Baris 2: Inisialisasi variabel "total" dimulai dari 0

Baris 3: Menyimpan hasil perkalian sesungguhnya dalam variabel "hasilPerkalian"

Baris 4: Mencetak "a x b" misal dalam contoh output dibawahnya menjadi "6 x 5" atau "7 x 10"

Baris 5: Melakukan perulangan ketika total belum mencapai hasil perkalian yang sesungguhnya

Baris 6: Menambahkan total dengan variabel "b"

Baris 7-8: Jika hasil perkalian sesungguhnya sudah sama dengan total maka diberi tanda sama di samping variabel "b"

Baris 9-10: Jika hasil perkalian sesungguhnya belum sama dengan total maka diberi tanda "+" di samping variabel "b"

Baris 11: Menampilkan hasil variabel "total"

Baris 13-14: Menerima input dari user

Baris 15: Memanggil function dengan argument dari input baris 13-14

SOAL 2

```
1 def ganjil(bawah,atas):
2     if bawah < atas: # Ketika parameter bawah lebih kecil dari atas maka perulangan i ke atas
3         for i in range (bawah,atas+1): # Batas atas ditambah 1 karena perulangan for akan berhenti sebelumnya
4             if i % 2 != 0:
5                 if i != atas and i != atas-1:
6                     print (i, end=", ") # Angka diberi koma ketika belum sama dengan batas atas (untuk angka ganjil) atau batas atas-1 (untuk angka genap)
7                 else:
8                     print (i, end=".") # Angka diberi titik ketika sudah mencapai batas
9             else: # Ketika parameter bawah lebih besar dari atas maka perulangan i ke bawah
10                for i in range (bawah,atas-1,-1): # Batas bawah dikurangi 1 karena perulangan for akan berhenti sebelumnya
11                    if i % 2 != 0:
12                        if i != atas and i != atas+1: # Angka diberi koma ketika belum sama dengan batas atas (untuk angka ganjil) atau batas atas-1 (untuk angka genap)
13                            print (i, end=", ")
14                        else:
15                            print (i, end=".")
16
17 masukan1 = int(input("Masukkan Batas Bawah: "))
18 masukan2 = int(input("Masukkan Batas Atas: "))
19 ganjil(masukan1,masukan2)
```

Masukkan Batas Bawah: 10
Masukkan Batas Atas: 30
11, 13, 15, 17, 19, 21, 23, 25, 27, 29.

Masukkan Batas Bawah: 97
Masukkan Batas Atas: 82
97, 95, 93, 91, 89, 87, 85, 83.

Gambar 14 - Program Latihan Mandiri 5.2

Di atas merupakan program untuk menampilkan deret angka ganjil secara berurutan dengan batas yang diberi pengguna. Penjelasan kode:

Baris 1: Pendefinisian function dengan nama ganjil()

Baris 2: Jika nilai bawah lebih kecil dari atas maka program dibawahnya dijalankan.

Baris 3: Melakukan perulangan dengan range Batas Bawah sampai Batas Atas+1. Diberi "+1" untuk mencegah perulangan berhenti pada 1 perulangan sebelumnya.

Baris 4: Jika i tidak habis dibagi 2, maka program akan menjalankan kode dibawahnya.

Baris 5-8: Program angka memberi angka koma apabila Batas Atas belum tercapai atau Batas Atas-1. Batas atas ini diberi "-1" untuk mengcover kemungkinan angka genap. Misal Batas Atasnya adalah 30, maka angka 30 ini akan menjadi masalah karena program akan berhenti setelah menampilkan angka 29 (angka ganjil yang terdekat dengan 30) karena program mendeteksi bahwa Batas Atas belum tercapai maka program terus menampilkan koma dan bukan titik sesuai dengan yang diminta soal.

Baris 9-10: Program akan menjalankan perulangan dengan range Batas Bawah sampai Batas Atas-1. Diberi "-1" untuk mencegah perulangan akan berhenti pada 1 perulangan sebelumnya. Kemudian argument step diberi angka "-1" supaya perulangan dilakukan mundur.

Baris 11: Jika i tidak habis dibagi 2, maka program akan menjalankan kode dibawahnya.

Baris 12-15: Program angka memberi angka koma apabila Batas Atas belum tercapai atau Batas Atas +1 (Ini untuk mengcover kemungkinan angka genap sama seperti penjelasan baris 5-8)

Baris 17-18: Menerima input dari user

Baris 19: Memanggil function dengan argument dari input baris 17-18

SOAL 3

```
1 def indeksSemester (jumlah):
2     totalNilai = 0
3     for i in range (1,jumlah+1): # Mengulang input nilai matakuliah sebanyak jumlah yang diinput oleh user
4         masukan1 = input(f"Nilai MK {i}: ")
5         if masukan1 == "A":
6             totalNilai = totalNilai + 4
7         elif masukan1 == "B":
8             totalNilai = totalNilai + 3
9         elif masukan1 == "C":
10            totalNilai = totalNilai + 2
11        else:
12            totalNilai = totalNilai + 1
13    nilaiIPS = totalNilai / jumlah
14    return round(nilaiIPS,2) # Melakukan pembulatan dua angka di belakang koma
15
16 masukan2 = int(input("Berapa jumlah mata kuliah? "))
17 print("Nilai IPS anda semester ini", indeksSemester(masukan2))
```

```
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 15 - Program Latihan Mandiri 5.3

Program di atas digunakan untuk menampilkan nilai IPS dari seorang mahasiswa sesuai dengan input yang dimasukkan. Penjelasan kode:

Baris 1: Pendefinisian function dengan nama indeksSemester

Baris 2: Inisialisasi variabel "totalNilai" dimulai dari angka 0

Baris 3: Mengulang program dibawahnya hingga sesuai dengan jumlah mata kuliah yang dimasukkan user

Baris 4: Menerima input nilai dari user untuk nilai per Mata Kuliah

Baris 5-12: Melakukan klasifikasi nilai, apabila yang dimasukkan "A" maka totalNilai akan ditambah 4, apabila yang dimasukkan "B" totalNilai akan ditambah 3, dan seterusnya.

Baris 13: Menaruh nilai rata-rata di dalam variabel nilai IPS dengan rumus totalNilai / jumlah

Baris 14: Hasil rata-rata kemudian di return dan di round dengan batasan 2 angka di belakang koma

Baris 16: Menerima input dari user berapa jumlah Mata Kuliah yang akan dihitung

Baris 17: Memanggil function dengan argument dari input baris 16