



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230970
Nama Lengkap	Gregorius Daniel Jodan Perminas
Minggu ke / Materi	06 / Percabangan & Perulangan Kompleks

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%)

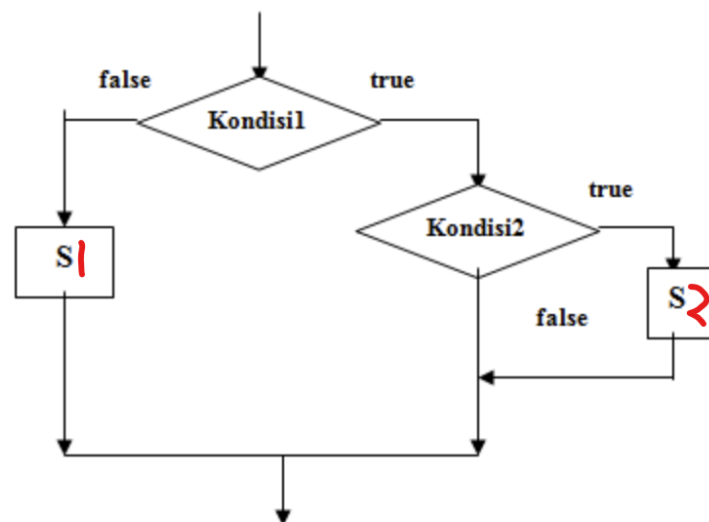
Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6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

Struktur Percabangan Kompleks

Percabangan sendiri merupakan salah satu jenis algoritma yang memungkinkan program komputer untuk memilih tindakan berdasarkan kondisi atau syarat tertentu. Struktur percabangan kompleks sendiri berarti percabangan tersebut memiliki banyak kondisi alternatif. Di bawah ini merupakan beberapa contoh bentuk dari struktur percabangan kompleks yang ditulis dalam bahasa pemrograman Python:

```
1  if kondisi1:
2      if kondisi2:
3          S
4      S
5  else:
6      S
7      S
8
```

Gambar 1 – Contoh Program 1



Gambar 2 - Flowchart Untuk Program 1

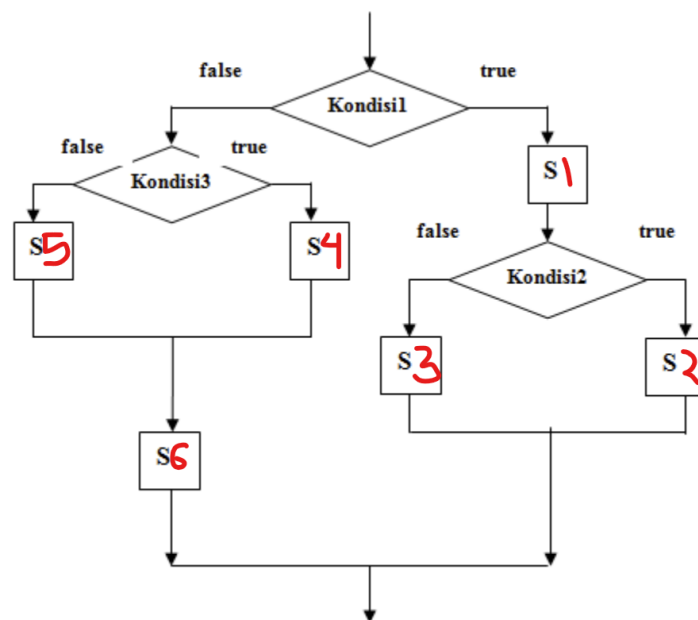
Misalnya saja ada program seperti di atas, program akan menjalankan S2 hanya ketika kondisi 1 DAN kondisi 2 terpenuhi. Jika kondisi 1 saja yang terpenuhi maka tidak ada program yang akan dijalankan karena S2 membutuhkan kondisi 1 dan 2 untuk terpenuhi. Sedangkan bila program mendeteksi dari kondisi 1 saja sudah tidak terpenuhi maka program akan mengambil jalur ELSE.

```

1  if kondisi1:
2      S1
3      if kondisi2:
4          S2
5          S2
6      else:
7          S3
8          S3
9  else:
10     if kondisi3:
11         S4
12         S4
13     else:
14         S5
15         S5
16     S6

```

Gambar 3 - Contoh Program 2



Gambar 4 - Flowchart Untuk Program 2

Di atas ini merupakan contoh program percabangan kompleks yang lain. Di sini terdapat 3 kondisi berbeda. Runtutan program ini yaitu, pertama program akan mengecek kondisi 1, apabila terpenuhi maka S1 akan dijalankan. Setelah itu program lanjut mengecek kondisi 2, apabila terpenuhi maka S2 yang dijalankan dan apabila tidak maka S3 yang dijalankan. Jalur akan berbeda ketika dari kondisi 1 saja sudah tidak terpenuhi, maka program akan langsung loncat ke baris ELSE dan mengecek apakah kondisi 3 terpenuhi. Jika kondisi 3 terpenuhi maka S4 dijalankan, apabila tidak maka S5 yang dijalankan. Untuk S6 sendiri akan selalu dijalankan tidak peduli apakah kondisi 3 terpenuhi atau tidak.

Catatan: Di bawah ini adalah dua jenis percabangan yang berbeda

```
1      # Percabangan 1
2      if (kondisi1):
3          instruksi1
4      elif(kondisi2):
5          instruksi2
6      elif(kondisi3):
7          instruksi3
8      elif(kondisi4):
9          instruksi4
10
11     #Percabangan 2
12     if (kondisi1):
13         instruksi1
14     if(kondisi2):
15         instruksi2
16     if(kondisi3):
17         instruksi3
18     if(kondisi4):
19         instruksi4
```

Gambar 5 - 2 Jenis Percabangan Yang Berbeda

Kedua jenis percabangan di atas tampak mirip, namun ketika di eksekusi Python akan membuat keputusan dengan cara yang berbeda ketika terdapat IF atau ELIF. Pada percabangan yang pertama ketika salah satu kondisi terpenuhi maka program akan mengeksekusi instruksi dibawahnya dan percabangannya lainnya tidak akan dijalankan. Pada percabangan yang kedua setiap kondisi akan dicek secara independen oleh Python, misalnya kondisi 1 terpenuhi maka tentu instruksi 1 akan dijalankan. Pembedanya adalah setelah mengecek kondisi 1 Python akan terus mengecek kondisi berikutnya walaupun kondisi di atasnya sudah bernilai TRUE.

Struktur Perulangan Kompleks

BREAK

```
1      for i in range(1000):
2          print(i)
3          if i==10:
4              break
```

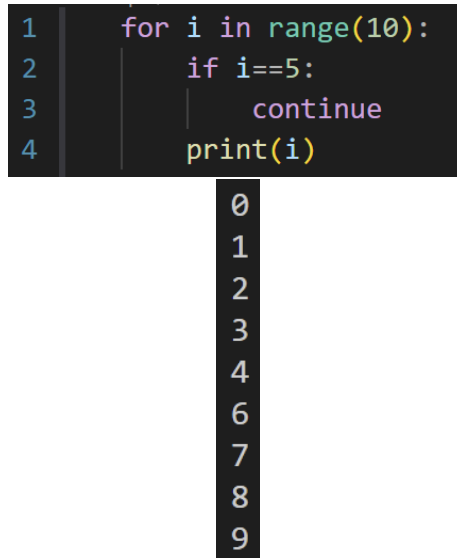


```
0
1
2
3
4
5
6
7
8
9
10
```

Gambar 6 - Perulangan dengan Statement Break

Program di atas seharusnya akan menampilkan angka dari 0 – 999 namun karena ada statement break maka perulangan dapat dihentikan sebelum range terpenuhi. Perulangan akan berhenti ketika kondisi percabangan yang telah ditentukan terpenuhi yaitu $i = 10$.

CONTINUE



```
1 for i in range(10):
2     if i==5:
3         continue
4     print(i)
```

```
0
1
2
3
4
6
7
8
9
```

Gambar 7 - Perulangan Dengan Statement Continue

Perintah continue menyebabkan proses perulangan untuk lompat ke iterasi setelahnya dan mengabaikan statement berikutnya. Biasanya perintah continue juga diimplementasikan menggunakan perintah IF. Program pada gambar 7 di atas dalam kondisi normal seharusnya memberi output angka 0,1,2,3,4,5,6,7,8,9, namun karena ada perintah continue maka angka 5 tidak ditampilkan. Hal ini terjadi karena pada saat angka 5 akan ditampilkan, perintah continue dijalankan, sehingga perintah mencetak di bagian bawahnya tidak akan dikerjakan dan langsung dilanjutkan ke perulangan berikutnya.

Perulangan Bertingkat

Perulangan pada dasarnya adalah sebuah logika pemrograman yang dapat memerintah program untuk mengerjakan pekerjaan yang sama berulang kali sesuai dengan batas yang telah ditentukan. Sedangkan perulangan bertingkat atau perulangan kompleks sendiri adalah bentuk per-ulangan di mana di dalam

suatu perulangan terdapat perulangan lain, sehingga terjadi perulangan bertingkat yang mengakibatkan waktu proses semakin lama.

```
1   for i in range(m):
2       for j in range(n):
3           <lakukan perintah ini di inner>
4           <lakukan perintah itu di inner>
5       <lakukan perintah lain di outer>
6       <lakukan perintah lain lagi di outer>
```

Gambar 8 - Perulangan For Bertingkat

Loop luar for i in range(m) akan berjalan sebanyak m kali. Untuk setiap iterasi dari loop luar: Loop dalam for j in range(n) akan berjalan sebanyak n kali. Setelah loop dalam selesai, program akan menjalankan loop luar lagi. Jadi, <lakukan perintah ini di inner> akan dijalankan sebanyak $m * n$ kali, sementara <lakukan perintah lain di outer> akan dijalankan sebanyak m kali.

```
1   while(i<=m):
2       while(j<=n):
3           <lakukan perintah ini di inner>
4           <lakukan perintah itu di inner>
5       <lakukan perintah lain di outer>
6       <lakukan perintah lain lagi di outer>
```

Gambar 9 - Perulangan While Bertingkat

Loop luar while(i<=m) akan berjalan selama i kurang dari atau sama dengan m. Untuk setiap iterasi dari loop luar: Loop dalam while(j<=n) akan berjalan selama j kurang dari atau sama dengan n. Setelah loop dalam selesai, program akan menjalankan loop luar lagi. Jadi, <lakukan perintah ini di inner> akan dijalankan sebanyak $m * n$ kali (dengan asumsi i dan j dimulai dari 1), sementara <lakukan perintah lain di outer> akan dijalankan sebanyak m kali.

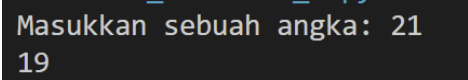
BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

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

SOAL 1

```
1 def cekPrima(angka):
2     if angka <= 2:
3         return f"Tidak ada angka prima yang lebih rendah dari {angka}"
4     else:
5         for i in range(angka-1, 1, -1):
6             prima = True
7             for j in range(2, i):
8                 if i % j == 0:
9                     prima = False
10                    break
11            if prima:
12                return i
13
14 masukan1 = int(input("Masukkan sebuah angka: "))
15 print(cekPrima(masukan1))
```



Gambar 10 - Program Untuk Latihan 6.1

Program di atas merupakan program untuk menampilkan angka prima terdekat yang lebih rendah dari *input* yang dimasukkan *user*. Penjelasan kode:

Baris 1: Pendefinisian function dengan nama cekPrima()

Baris 2-3: Karena tidak ada angka prima yang lebih rendah dari angka 2, maka diberi kondisi ketika angka yang di *input user* kurang dari atau sama dengan 2, maka program akan menampilkan "Tidak ada angka prima yang lebih rendah dari {angka}"

Baris 4: Ketika angka lebih dari 2 program akan melakukan pencarian angka prima

Baris 5-6: Syarat bilangan prima adalah bilangan tersebut hanya bisa dibagi oleh angka 1 dan dirinya sendiri, lalu karena yang diminta angka terdekat dari bawah maka untuk mempercepat perhitungan iterasi dimulai dari angka-1 supaya program tidak membagi angka dengan dirinya sendiri. Asumsi awal prima adalah TRUE

Baris 7-10: Kedua baris ini akan mencoba untuk membagi *i* dengan *j*. Ketika ditemukan salah satu *i* yang habis dibagi *j*, maka program akan langsung menyatakan bahwa prima adalah FALSE dan perulangan *for j in range(2,i)* akan dihentikan dan program akan melanjutkan ke *i* yang berikutnya.

Baris 11-12: Ketika program tidak bisa menemukan bahwa i bisa dibagi j berapapun maka variabel *prima* akan tetap bernilai `TRUE`, sehingga kondisi `if prima` akan dijalankan dan nilai i dikembalikan

Baris 14-15: Digunakan untuk mengambil *input* dari *user* dan memanggil fungsi `cekPrima()`

SOAL 2

```
1  def deretFaktorial(n):
2      faktorial = n
3      total = 1
4      for _ in range(1, n+1):
5          for j in range(1, faktorial+1):
6              total = total * j
7              print(total, end = " ")
8              for k in range(faktorial, 0, -1):
9                  print(k, end = " ")
10             total = 1
11             faktorial = faktorial - 1
12             print()
13
14 masukan1 = int(input("Masukkan n: "))
15 deretFaktorial(masukan1)
```

Masukkan n: 6
720 6 5 4 3 2 1
120 5 4 3 2 1
24 4 3 2 1
6 3 2 1
2 2 1
1 1

Gambar 11 - Program Untuk Latihan 6.2

Program ini digunakan untuk menampilkan deret faktorial dengan hasil faktorial diletakkan pada bagian awal baris. Penjelasan kode:

Baris 1: Pendefinisian fungsi `deretFaktorial()`

Baris 2: Menyiapkan variabel *faktorial* untuk menyimpan *input user* di dalam fungsi

Baris 3: Menginisiasi variabel *total* dimulai dari angka 1

Baris 4: Melakukan perulangan sebanyak n yang diminta (banyak baris yang akan dicetak)

Baris 5-7: Melakukan perulangan perkalian untuk mencari total perkalian faktorial yang dimasukkan *user* dan menaruhnya di awal baris

Baris 8-9: Melakukan perulangan sebanyak angka yang dimasukkan *user*, namun iterasi dilakukan dari atas ke bawah. Lalu hasilnya ditampilkan di sebelah total yang dihitung oleh baris 5-7

Baris 10: Digunakan untuk mereset *total* kembali ke angka 1

Baris 11: Variabel faktorial akan dikurangi 1 supaya baris berikutnya dimulai dari faktorial = n-1 dan begitu seterusnya

Baris 12: Digunakan untuk mencetak baris berikutnya

Baris 14-15: Menerima *input* dari *user* dan memanggil fungsi deretFaktorial()

SOAL 3

```
1 def deretKotak (t,l):
2     luas = t * l
3     for i in range (1,luas+1):
4         print(i, end = " ")
5         if i % l == 0:
6             print()
7
8 masukan1 = int(input("Masukkan Tinggi: "))
9 masukan2 = int(input("Masukkan Lebar: "))
10 deretKotak(masukan1,masukan2)
```

Masukkan Tinggi: 5
Masukkan Lebar: 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20

Gambar 12 - Program Untuk Latihan 6.3

Program di atas digunakan untuk menampilkan deret bilangan berbentuk kotak dengan tinggi dan lebar yang di sesuaikan dengan *input user*. Penjelasan kode:

Baris 1: Pendefinisian fungsi deretKotak()

Baris 2: Menyiapkan variabel luas untuk menyimpan perhitungan total luas

Baris 3: Melakukan perulangan i dari angka 1 sampai luas yang di *input* oleh *user*

Baris 4: Mencetak i dalam satu baris

Baris 5-6: Ketika i bisa dibagi lebar (kelipatan lebar) maka cetak baris ke bawah

Baris 8-10: Menerima *input* dari *user* dan memanggil fungsi deretKotak ()