Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230982**
Nama Lengkap	**Rainie Fanita Chrisabel Hadisantoso**
Minggu ke / Materi	**05 / Percabangan dan Perulangan Kompleks**

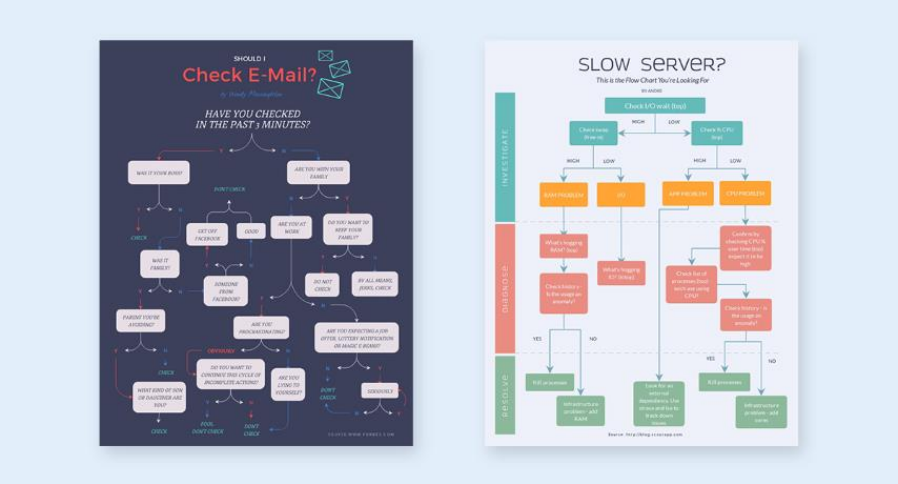
SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

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

MATERI : Percabangan dan Perulangan Kompleks

Pertemuan ini bertujuan untuk memahami dan menggunakan struktur percabangan dan perulangan kompleks dalam Python serta mengaplikasikannya pada program untuk memecahkan masalah-masalah yang diberikan. Dalam membangun struktur program, kita tidak hanya menggunakan percabangan dan perulangan secara individu, namun kita juga menempatkan percabangan dan perulangan di dalam percabangan atau perulangan utama. Hal tersebut dilakukan untuk membangun struktur program yang memerlukan pemenuhan kondisi maupun perulangan yang lebih kompleks untuk memecahkan masalah. Dengan percabangan dan perulangan kompleks kita dapat merepresentasikan proses berpikir yang paling rumit sekalipun.



Gambar 1. Contoh flowchart masalah sehari-hari

Struktur Percabangan Kompleks

Ada berbagai macam bentuk percabangan kompleks yang dapat diaplikasikan dalam algoritma program. Percabangan kompleks biasa digunakan ketika ada kondisi yang perlu dipenuhi dan ketika dipenuhi ada pembagian lagi dengan kondisi-kondisi tertentu. Percabangan kompleks memiliki banyak sekali bentuk struktur. Berikut beberapa contoh bentuk percabangan kompleks beserta contoh program Python.

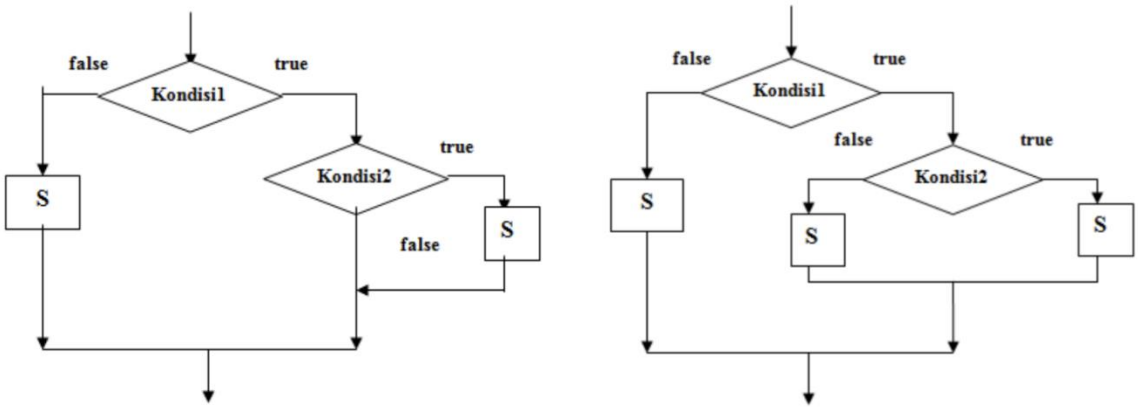
1. Percabangan if else, dan di dalam if terdapat percabangan if. Contoh flowchart di Gambar 2.

```
if bilangan > 0:
    if bilangan > 0:
        print("bilangan bulat positif")
else:
    print("bukan bilangan bulat positif")
```

2. Percabangan if else, dan di dalam if terdapat percabangan if else. Biasanya untuk mengklasifikasikan kondisi di dalam kondisi yang terpenuhi. Contoh flowchart di Gambar 2.

```
if bilangan > 0:
    if bilangan == 0:
        print("nol")
    else:
```

```
        print("bilangan bulat positif")
else:
    print("bukan bilangan bulat positif")
```



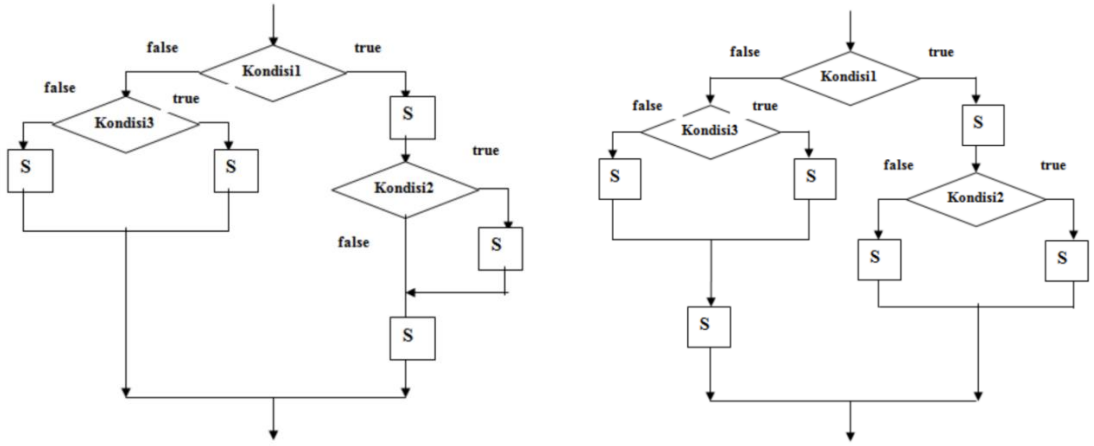
Gambar 2. Flowchart 1 dan 2

3. Percabangan if else, dan if memiliki percabangan if, dan else memiliki percabangan if else di dalamnya. Contoh flowchart di Gambar 3.

```
if bilangan > 0:
    if bilangan == 0:
        print("nol")
     print("bilangan bulat positif")
else:
    if bilangan % 2 == 0:
        print("bilangan genap bulat negatif")
    else:
        print("bilangan ganjil bulat negatif")
```

4. Percabangan if else, masing-masing memiliki percabangan if else di dalamnya dan program diakhiri dengan print akhir. Contoh flowchart di Gambar 3.

```
if bilangan > 0:
    print("bilangan bulat positif")
    if bilangan % 2 == 0:
        print("bilangan genap bulat positif")
    else:
        print("bilangan ganjil bulat positif")
else:
    if bilangan % 2 == 0:
        print("bilangan genap bulat negatif")
    else:
        print("bilangan ganjil bulat negatif")
print("yeyy")
```



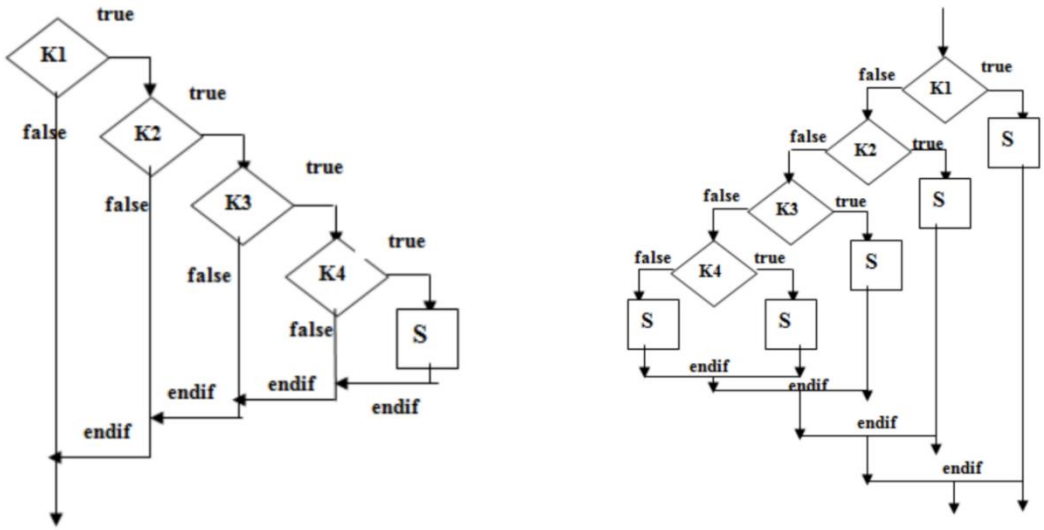
Gambar 3. Flowchar 3 dan 4

5. Percabangan if, dalamnya ada percabangan if, dalamnya ada percabangan if, dalamnya ada perca-. Gambar flowchart ada di Gambar 4.

```
if bilangan > 0:
    if bilangan > 10:
        print("satuan")
        if bilangan > 100:
            print("puluhan")
            if bilangan > 1000:
                print("ratusan")
                if bilangan > 10000:
                    print("ribuan")
```

6. Percabangan if else, dalamnya ada percabangan if else, dalamnya ada if else, dalamnya lagi if else. Gambar flowchart ada di Gambar 4.

```
if bilangan > 0 and bilangan < 10:
    print("satuan")
else:
    if bilangan > 10 and bilangan < 100:
        print("puluhan")
    else:
        if bilangan > 100 and bilangan < 1000:
            print("ratusan")
        else:
            if bilangan > 1000 and bilangan < 10000:
                print("ribuan")
            else:
                print("banyak banget")
```



Gambar 4. Flowchart 5 dan 6

Penggunaan percabangan bertingkat sangat efektif dan efisien untuk digunakan dalam beberapa masalah tertentu. Dengan menggunakan percabangan bertingkat, program tidak mencoba semua kondisi percabangan satu-persatu. Sehingga percabangan bertingkat menggunakan waktu eksekusi yang lebih cepat dibandingkan dengan waktu eksekusi menggunakan percabangan biasa. Dengan percabangan biasan komputer akan mencoba kondisi percabangan satu-persatu, meskipun akhirnya hanya ada satu kondisi percabangan yang terpenuhi. Proses yang runtut menyebabkan percabangan biasa memiliki waktu eksekusi yang lebih lambat.

Percabangan biasa:

```
if bilangan > 0 and bilangan < 10:
    print("satuan")
elif bilangan > 10 and bilangan < 100:
    print("puluhan")
elif bilangan > 100 and bilangan < 1000:
    print("ratusan")
elif bilangan > 1000 and bilangan < 10000:
    print("ribuan")
else:
    print("banyak banget")
```

Percabangan terpisah:

```
if bilangan > 0 and bilangan < 10:
    print("satuan")
if bilangan > 10 and bilangan < 100:
    print("puluhan")
if bilangan > 100 and bilangan < 1000:
    print("ratusan")
```

```
if bilangan > 1000 and bilangan < 10000:
    print("ribuan")
if bilangan > 10000:
    print("banyak banget")
```

Struktur Perulangan Kompleks

Dalam struktur perulangan kompleks digunakan perintah break, continue maupun perlangan dalam perulangan. Berikut penjelasan beserta contoh program masing-masing.

1. Break

   Perintah break digunakan untuk menghentikan perulangan yang berjalan. Peirntah break dapat diaplikasikan dengan menggunakan percabangan if. Misalnya ketika memenuhi kondisi if maka perulangan akan break atau berhenti.

   ```
   for i in range (10000):
       if i == 10:
           print("stop")
           break
       print(i, end=" ")
   ```

   Dari program perulangan menggunakan break diatas, akan ditampilkan hasil:

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

   Meskipun perulangan memiliki rentang i hingga 10000, ketika i mencapai 10, program akan menampilkan “stop” dan kemudian perintah break untuk menghentikan jalannya perulangan. Selain itu program juga tidak menampilkan bilangan 10, karena perintah if dijalankan terlebih dahulu sehingga program terhenti sebelum menampilkan bilangan i ke 10. Coba jika perintah print dan if dibalik:

   ```
   for i in range (10000):
       print(i, end=" ")
       if i == 10:
           print("stop")
           break
   ```

   Program akan menampilkan:

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

   Program lebih dahulu membaca perintah print i dan baru kemudian mencapai perintah if, menampilkan “stop” dan menghentikan program.

2. Continue

   Perintah continue digunakan untuk melompati perulangan yang dilakukan. Continue juga bisa menggunakan percabangan if dengan kondisi yang masuk dalam range perulangan. Berikut contoh program menggunakan continue serta hasil yang ditampilakn.

   ```
   for i in range (10):
       if i == 5:
   ```

```
        print("skip")
        continue
    print(i, end=" ")
```

Hasilnya:

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

Dari hasil kita dapat melihat ketika i bernilai 5 sesuai kondisi if, program menampilkan “skip” kemudian tidak lanjut menampilkan i. Hal tersebut karena perintah continue melompati perulangan saat itu dan melanjutkan proses perulangan berikutnya.

3. Perulangan Bertingkat

Perulangan bertingkat atau nested loop adalah struktur perulangan komplkes dimana di dalam perulangan ada perulangan lain. Struktur ini cenderung memerlukan waktu proses yang lebih lama, karena perulangan mengulang-ulang perintah dan di dalam perulangan ada perulangan lagi yang juga mengulang-ulang perintah. Nested loop dapat merangkai proses yang cukup rumit sehingga dalam menggunakan perlu ketelitian. Berikut contoh nested loop, perulangan i dengan rentangg 5 berisi perulangan j dalam rentang 10:

```
for i in range (5):
    for j in range (10):
        print("O", end= " ")
    print()
```

Hasilnya:

```
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
```

Hasil program menampilkan barisan sebanyak 5 baris yang berisi “O” sebanyak 10 kali setiap barisnya. Perulangan i bertugas mengulang perulangan inner tiap barisan dan perulangan j bertugas menampilkan “O” sebanyak 10 kali. Berikut penggunaan perintah perulangan lain dengan hasil yang sama:

```
i = 0
j = 0
while i <= 5:
    while j <= 10:
        print("O", end=" ")
        j += 1
    i += 1
    print()
```

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.

SOAL 5.1

Membuat program untuk mencari bilangan prima ganjil lebih kecil terdekat dari bilangan yang diinputkan oleh pengguna (n). Berikut contoh hasil test case:

- Contoh: input n=12, maka prima terdekat < 12 adalah 11
- Contoh: input n=21, maka prima terdekat < 21 adalah 19

1. Source Code

```
def prima(n):
  if n > 1:
    for i in range (2,n-1):
      if n % i == 0:
        return False
        break
    else:
        return True
  else:
    return False

def terdekat(n):
  turun = n
  naik = n

  if prima(n) == True:
    print(f"input n = {n}, maka prima terdekat < {n} adalah {n}")
  else:
    for i in range (n-1):
      turun -= 1
      naik += 1

      turun_prima = prima(turun)
      naik_prima = prima(naik)

      if turun_prima and naik_prima:
        print(f"input n = {n}, maka prima terdekat < {n} adalah {turun} dan {naik}")
        break
      elif turun_prima and not naik_prima:
        print(f"input n = {n}, maka prima terdekat < {n} adalah {turun}")
        break
      elif naik_prima and not turun_prima:
        print(f"input n = {n}, maka prima terdekat < {n} adalah  {naik}")
        break

n= int(input("Masukkan nilai n = "))
terdekat(n)
```

2. Output

```
Masukkan nilai n = 12
input n = 12, maka prima terdekat < 12 adalah 11 dan 13
```

3. Penjelasan

Dalam program dibuat 2 fungsi, fungsi pertama untuk mencari bilangan prima dan fungsi kedua untuk mencari bilangan prima terdekat.

a. Mencari bilangan prima= def prima(n)

Untuk mencari prima bilangan > =1 dieliminasi menjadi False

Jika bilangan < 1

- Dicari bilangan bukan prima dengan perulangan dengan rentang 2 sampai bilangan – 1
- Karena bilangan prima hanya bisa dibagi oleh dirinya sendiri, sehingga jika bilangan dapat dibagi dengan bilangan dibawahnya(sisa bagi = 0), maka bilangan tersebut False.
- Maka selain itu, bilangan adalah prima.

b. Mencari prima terdekat= def terdekat(n)

Untuk dicek apakah bilangan n adalah prima, maka n dimasukkan fungsi prima, jika true maka terdekat adalah n sendiri.

Jika tidak, maka dilakukan perulangan

- Untuk mencari prima terdekat, di tambahkan variabel tambahan untuk mencari prima dibawah n (-1) dan diatas n (+1). Variabel sudah didefinisi di awal fungsi bernilai n.
- Kemudian masing-masing variabel naik dan turun dicek dengan fungsi prima.
- Jika keduanya prima maka terdekat adalah keduanya
- Jika hanya naik prima, maka terdekat adalah variabel diatas n
- Jika hanya turun prima, maka terdekat adalah variabel dibawah n

SOAL 5.2

Program menghitung dan menampilkan fibonacci perkalian berdasarkan nilai yang dimasukkan pengguna. Hasil perkalian fibonacci ditampilkan diikuti bilangan-bilangan pengalinya. Kemudian ditampilkan hasil perkalian fibonacci dengan nilai-1 diikuti bilangan-bilangan pengalinya. Begitu terus hingga hasil perkalian dan bilangan pengali bernilai 1.

1. Source Code

```
[ ]  def fibo_kali(n):
       i = n
       while i > 0:
         hasil = 1
         for j in range (i, 0, -1):
           hasil *= j
         print(hasil, end=" ")
         for k in range (i, 0, -1):
           print(k, end=" ")
         i -= 1
         print()

     n = int(input("n = "))
     fibo_kali(n)
```

2. Output

```
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
```

3. Penjelasan

   Dalam fungsi fibo_kali, di tuliskan program untuk menghitung sekaligus menampilkan hasil perkalian dan deret angka. Pertama-tama didefinisikan variabel i adalah bilangan n yang dimasukkan pengguna. Variabel tidak dimasukkan ke dalam perulangan while agar nilai i tidak ikut kembali ke nilai awal setiap perulangannya. Dengan begitu tiap terjadi perulangan nilai i merupakan hasil nilai i perulangan sebelumnya. Kemudian perulangan selama i > 0

   - Dalam perulangan kita menghitung hasil dengan nilai awal 1. Hasil dimasukkan dalam perulangan karena tiap perulangan memiliki hasil yang beda.
   - Perulangan for pertama untuk mengkalikan hasil dengan bilangan j dari i sampai 1.
   - Kemudian hasil perkalian ditampilkan
   - Perulangan for kedua untuk menampilkan bilangan pengali secara urut dari besar ke kecil setelah hasil perkalian. Pengulangan menampilkan bilangan pengali dalam 1 barisdimulai dari i hingga 1.
   - Terakhir i dikurangi 1, sehingga setiap perulangan I terus menerus berkurang dan menampilkan hasil fibonacci dengan urut hingga 1.

   Diluar fungsi diberikan variabel n untuk input nilai dari pengguna dan memanggil fungsi fibo_kali. Fungsi fibo_kali tidak perlu di print karena didalam fungsi sudah ada perintah print untuk menampilkan hasil fungsi.

SOAL 5.3

Membuat program menampilkan bilangan secara urut dari bilangan 1 dengan jumlah bilangan per baris dan jumlah baris bilangan yang mengikuti input dari pengguna. Jumlah bilangan per baris mengikuti lebar dan jumlah baris mengikuti tinggi input. Bilangan dalam baris ditampilkan urut dari 1 hingga nilai lebar × tinggi.

1. Source Code

```
[ ]  def deret(tinggi, lebar):
       i = 1
       while i <= tinggi * lebar:
         for j in range (1, tinggi+1):
           for k in range (1, lebar+1):
             print(i, end=" ")
             i += 1
           print()

     tinggi = int(input("Masukkan tinggi = "))
     lebar = int(input("Masukkan lebar = "))

     deret(tinggi, lebar)
```

2. Output

```
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
```

3. Penjelasan

Dalam fungsi deret dengan 2 nilai tinggi dan lebar didefinisikan variadiberikan. Variabel I berada diluar semua perulangan agar nilai I terus bertambah setiap perulangan terjadi.

- Perulangan while selama I <= tinggi × lebar, agar perulangan stop dengan jumlah bilangan perbaris dan baris yang sesuai.
- Di dalam while diberikan perulangan for dengan rentang 1 hingga tinggi + 1, perulangan ini untuk menghasilkan jumlah baris.
- Kemudian dalam perulangan for ada perulangan for lagi dengan rentang 1 sampai lebar + 1 untuk menampilkan bilangan per baris sesuai lebar yang diberikan. Lalu nilai I ditambahkan 1.
- Diberikan print() diluar for untuk mengganti baris setiap perulangan dilakukan. Perulangan while akan berhenti ketika bilangan I yang ditampilkan sudah mencapai lebar × tinggi.

Lalu seperti biasa, nilai-nilai yang diperlukan dimasukkan oleh pengguna dalam nama variabel yang diperlukan fungsi dan fungsi dipanggil.

Link Github:

https://github.com/rainiefch/Praktikum-Algoritma-Pemrograman_71230982