

3. Struktur Kontrol Percabangan

3.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menyusun boolean expression dengan benar.
2. Dapat merangkai beberapa boolean expression menggunakan operator logical dengan benar.
3. Dapat menyusun percabangan dengan if sesuai dengan permasalahan yang dihadapi.
4. Dapat menangani jika terjadi kesalahan masukan pengguna menggunakan bentuk exception sederhana.

3.2 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).

3.3 Materi

3.3.1 Boolean Expression dan Logical Operator

Perhatikan kasus berikut: Voucher diskon 30% dapat dipakai jika minimum pembelian anda adalah Rp. 100.000. Minimum pembelian adalah syarat yang harus dipenuhi untuk mendapatkan diskon. Minimum pembelian tersebut dapat dinyatakan dalam Python sebagai berikut:

```
pembelian >= 100000
```

Bentuk tersebut dinamakan sebagai boolean expression, karena hasilnya hanya ada dua kemungkinan, yaitu **True** atau **False**, tergantung dari variabel pembelian. Jika anda memasukkan perintah tersebut ke dalam Python mode interaktif, hasilnya dapat dilihat pada Gambar 3.1.

```
yuanlukito@linuxstation: ~  
yuanlukito@linuxstation:~$ python3  
Python 3.7.5 (default, Nov 20 2019, 09:21:52)  
[GCC 9.2.1 20191008] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> pembelian = 120000  
>>> pembelian >= 100000  
True  
>>>  
>>> pembelian = 45000  
>>> pembelian >= 100000  
False  
>>>  
>>> pembelian = 100000  
>>> pembelian >= 100000  
True  
>>> █
```

Gambar 3.1: Kemungkinan hasil dari boolean expression.

Boolean expression dapat disusun menggunakan operator-operator perbandingan seperti pada Tabel 3.1.

Tabel 3.1: Operator-operator perbandingan (comparison).

Operator	Keterangan
x == y	Apakah x sama dengan y?
x != y	Apakah x tidak sama dengan y?
x > y	Apakah x lebih besar dari y?
x >= y	Apakah x lebih besar atau sama dengan y?
x < y	Apakah x lebih kecil dari y?
x <= y	Apakah x lebih kecil atau sama dengan y?
x is y	Apakah x sama dengan y?
x is not y	Apakah x tidak sama dengan y?

Anda harus bisa menyusun bentuk boolean expression dan memilih operator yang tepat sesuai dengan permasalahan yang dihadapi. Beberapa hal yang perlu anda perhatikan saat menyusun bentuk boolean expression:

- Bentuk boolean expression pasti hasilnya **hanya ada dua**, yaitu True atau False.
- Perhatikan kata-kata khusus seperti **minimum, maksimum, tidak lebih dari, tidak kurang dari, tidak sama, tidak berbeda**.
- Perhatikan dengan seksama dan tentukan variabel yang perlu dibandingkan dengan benar sesuai dengan permasalahan.

Beberapa contoh permasalahan dan bentuk boolean expression-nya dapat dilihat pada Tabel 3.2.

Beberapa boolean expression dapat digabungkan dengan menggunakan logical operator. Logical operator pada Python adalah **and**, **or** dan **not**. Sebagai contoh, misalnya wahana Rollercoaster

Tabel 3.2: Operator-operator perbandingan (comparison).

Contoh masalah	Boolean expression
Untuk lulus dibutuhkan IPK minimum 2.25	<code>ipk >= 2.25</code>
Golden Button hanya diberikan untuk Youtuber dengan subscriber lebih dari 1 juta	<code>subscriber > 1000000</code>
Pengendara dengan kecepatan lebih dari 90 km/jam akan mendapatkan tilang	<code>kecepatan > 90</code>
Wahana Rollercoaster hanya bisa dinaiki oleh mereka yang tinggi badannya lebih dari 110 cm	<code>tinggi > 110</code>
Nilai ujian Hanna adalah 75 sedangkan Robby mendapatkan nilai 75. Apakah nilai keduanya sama?	<code>hanna is robby</code>
Junaedi memiliki 10 sepatu, Ricky punya 15 sepatu dan Arnold punya 20 sepatu. Apakah gabungan sepatu Junaedi dan Ricky lebih banyak dari sepatu milik Arnold?	<code>junaedi + ricky > arnold</code>

hanya dapat dinaiki oleh penumpang dengan usia minimal 10 tahun dan tinggi badan minimal 110 cm. Kedua persyaratan tersebut dapat dinyatakan dalam bentuk sebagai berikut:

`usia >= 10 and tinggi >= 110`

Diskon diberikan kepada member atau jumlah pembelian lebih dari Rp. 500.000:

`member == true or pembelian > 500000`

3.3.2 Bentuk-bentuk Percabangan

Percabangan pada Python secara umum ada tiga bentuk, yaitu: **conditional**, **alternative** dan **chained conditional**. Bentuk conditional secara umum dinyatakan dalam kode program sebagai berikut:

```
if <kondisi>:
    <lakukan ini>
    <lakukan ini>
    ...
```

Sebagai contoh, jika nilai akhir > 70 maka akan mendapatkan sertifikat kelulusan. Kode programnya sebagai berikut:

```
if nilai_akhir > 70:
    print("Anda lulus dan mendapatkan sertifikat kelulusan!")
```

Bentuk alternative conditional adalah bentuk percabangan yang memiliki dua alternatif langkah yang harus dijalankan berdasarkan kondisi tertentu. Secara umum bentuknya adalah sebagai berikut:

```
if <kondisi>:
    <lakukan ini>
    <lakukan ini>
    ...
else:
    <lakukan itu>
    <lakukan itu>
    ...
```

Sebagai contoh, jika nilai akhir > 60, tampilkan tulisan Lulus. Jika tidak, tampilkan tulisan Tidak Lulus. Pada contoh ini, ada dua kemungkinan tulisan yang muncul, yaitu Lulus atau Tidak Lulus. Implementasinya sebagai berikut:

```
if nilai_akhir > 60:
    print("Lulus")
else:
    print("Tidak Lulus")
```

Bentuk chained conditional digunakan jika kemungkinan langkah yang harus dijalankan berikutnya lebih dari dua. Bentuknya secara umum adalah sebagai berikut:

```
if <kondisi 1>:
    <lakukan A1>
    <lakukan A2>
    ...
elif <kondisi 2>:
    <lakukan B1>
    <lakukan B2>
    ...
elif <kondisi 3>:
    <lakukan C1>
    <lakukan C2>
    ...
...
else
    <lakukan ...>
    <lakukan ...>
    ...
```

Sebagai contoh, misalnya sebuah toko pakaian memberi diskon yang besarnya ditentukan oleh nilai pembelian anda. Untuk pembelian di atas Rp. 1.000.000 mendapatkan diskon 30%. Pembelian lebih dari Rp. 500.000 sampai Rp. 1.000.000 mendapatkan diskon 20%. Pembelian dari Rp. 100.000 sampai Rp. 500.000 mendapatkan diskon 15%. Pembelian di bawah Rp. 100.000 tidak mendapatkan diskon. Kemungkinan diskon yang diberikan ada 4, yaitu: 30%, 20%, 15% dan 0% (tidak diskon). Bagaimana implementasinya? Perhatikan implementasi berikut ini:

```
if pembelian > 1000000:
    diskon = 0.3          # diskon 30%
elif pembelian > 500000 and pembelian <= 1000000:
    diskon = 0.2          # diskon 20%
elif pembelian >= 100000 and pembelian <= 500000:
    diskon = 0.15         # diskon 15%
else:
    diskon = 0            # tidak ada diskon
```

Pernyataan "Pembelian lebih dari Rp. 500.000 sampai Rp. 1.000.000 mendapatkan diskon 20%" diimplementasikan dalam bentuk gabungan dari dua boolean expression, yaitu:

```
pembelian > 500000 and pembelian <= 1000000
```

Umumnya untuk menyatakan rentang nilai tersebut kita menggunakan logical operator and, karena kedua kondisi tersebut harus terpenuhi agar bernilai True.

Selain bentuk-bentuk percabangan tersebut, Python juga memiliki sintaks alternatif untuk menuliskan percabangan yang biasa disebut sebagai **ternary operator**. Sebagai contoh bentuk percabangan berikut ini:

```
pembelian = int(input("Jumlah pembelian: "))
diskon = 0.1 if pembelian > 100000 else 0
```

Bentuk ternary tersebut merupakan bentuk lain dari if-else berikut ini:

```
pembelian = int(input("Jumlah pembelian: "))
if pembelian > 100000:
    diskon = 0.1
else:
    diskon = 0
```

3.3.3 Penanganan Kesalahan Input Menggunakan Exception Handling

Dalam menangani input dari pengguna, kita juga perlu memperhatikan potensi kesalahan yang mungkin terjadi sehingga program tidak bisa berjalan dengan semestinya. Untuk lebih jelasnya, perhatikan program yang meminta input usia pengguna, kemudian program akan menampilkan apakah pengguna termasuk lansia, dewasa, remaja, kanak-kanak atau balita. Kategori usia tersebut mengikuti aturan sebagai berikut:

- Balita: 0-5 tahun.
- Kanak-kanak: 6-11 tahun.
- Remaja: 12-25 tahun.
- Dewasa: 26-45 tahun.
- Lansia: > 45 tahun.

Program tersebut adalah sebagai berikut:

```
1  usia = int(input("Masukkan usia anda: "))
2  if usia <= 5:
3      print("Balita")
4  elif usia >= 6 and usia <= 11:
5      print("Kanak-kanak")
6  elif usia >=12 and usia <= 25:
7      print("Remaja")
8  elif usia >= 26 and usia <= 45:
9      print("Dewasa")
10 elif usia > 45:
11     print("Lansia")
```

Jika program tersebut dijalankan beberapa kali dengan input yang berbeda-beda, hasilnya telah sesuai dengan yang diharapkan. Hasil dari program tersebut dapat dilihat pada Gambar 3.2.

Bagian dari program tersebut yang meminta pengguna memasukkan usianya ada di baris 1, yaitu:

```
usia = int(input("Masukkan usia anda: "))
```

```
yuanlukito@linuxstation: ~/Documents
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: 33
Dewasa
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: 7
Kanak-kanak
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: 68
Lansia
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: 18
Remaja
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: 2
Balita
yuanlukito@linuxstation:~/Documents$
```

Gambar 3.2: Hasil program kategori usia.

Seperti yang telah dipelajari pada Bab sebelumnya, fungsi **input()** digunakan untuk membaca masukan/input yang diberikan oleh pengguna. Fungsi **input()** akan menghasilkan string, sedangkan usia seharusnya merupakan angka/bilangan sehingga perlu dikonversi menjadi bilangan bulat dengan fungsi **int()**. Program tersebut akan berjalan dengan baik selama pengguna tidak memasukkan input yang salah atau tidak sesuai yang diharapkan. Sebagai contoh, perhatikan Gambar 3.3 yang menunjukkan pengguna memasukkan input yang tidak sesuai.

```
yuanlukito@linuxstation: ~/Documents
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: dua puluh
Traceback (most recent call last):
  File "kategoriusia.py", line 1, in <module>
    usia = int(input("Masukkan usia anda: "))
ValueError: invalid literal for int() with base 10: 'dua puluh'
yuanlukito@linuxstation:~/Documents$
```

Gambar 3.3: Jika input tidak sesuai yang diharapkan, program akan berhenti.

Bagaimana cara menangani input yang tidak sesuai? Salah satu cara yang dapat digunakan adalah menggunakan **try** dan **except**. Penggunaannya dalam kasus kategori usia dapat dilihat pada source code program berikut ini:

```
1 inputuser = input("Masukkan usia anda: ")
2 try:
3     usia = int(inputuser)
4     if usia <= 5:
5         print("Balita")
6     elif usia >= 6 and usia <= 11:
7         print("Kanak-kanak")
8     elif usia >=12 and usia <= 25:
9         print("Remaja")
10    elif usia >= 26 and usia <= 45:
```

```

11         print("Dewasa")
12     elif usia > 45:
13         print("Lansia")
14 except:
15     print("Anda salah memasukkan input usia")

```

Jika dijalankan dan diberi input yang tidak sesuai, program tidak mengalami kesalahan seperti sebelumnya. Hasilnya jika dijalankan dapat dilihat pada Gambar 3.4.



```

yuanlukito@linuxstation: ~/Documents
yuanlukito@linuxstation:~/Documents$ python3 kategoriusia.py
Masukkan usia anda: dua puluh
Anda salah memasukkan input usia
yuanlukito@linuxstation:~/Documents$

```

Gambar 3.4: Hasil program kategori usia.

3.4 Kegiatan Praktikum

Kegiatan praktikum yang akan dilakukan adalah sebagai berikut:

1. Studi kasus masalah-masalah yang membutuhkan percabangan (conditional).
2. Studi kasus penanganan input dari pengguna dan penggunaan try-except.

3.4.1 Contoh Masalah-masalah Percabangan

■ Contoh 3.1 Demam atau tidak?

Buatlah sebuah program yang meminta input suhu tubuh dari pengguna, kemudian program akan menentukan apakah pengguna mengalami demam atau tidak. Kriteria demam jika suhu tubuh lebih besar atau sama dengan 38 derajat Celcius. Alur dari program ini dapat dinyatakan dalam potongan flowchart seperti pada Gambar 3.5.

Hal pertama yang perlu anda pikirkan adalah apa input/masukan yang diperlukan untuk masalah ini? Dari deskripsi masalah kita bisa menentukan bahwa dibutuhkan informasi besarnya suhu tubuh untuk bisa menentukan apakah seseorang mengalami demam atau tidak. Masalah demam atau tidak ini memiliki dua kemungkinan hasil, yaitu demam ($\text{suhu} \geq 38$) dan tidak demam. Berarti kriteria untuk demam sudah diketahui yaitu $\text{suhu} \geq 38$. Bagaimana dengan kriteria untuk tidak demam? Karena kemungkinan hasilnya cuma dua, maka kriteria tidak demam merupakan kebalikan dari kriteria demam, yaitu $\text{suhu} < 38$. Cara lain untuk menyusun kriterianya adalah dengan cara menuliskan nilai-nilai yang memenuhi, seperti berikut:

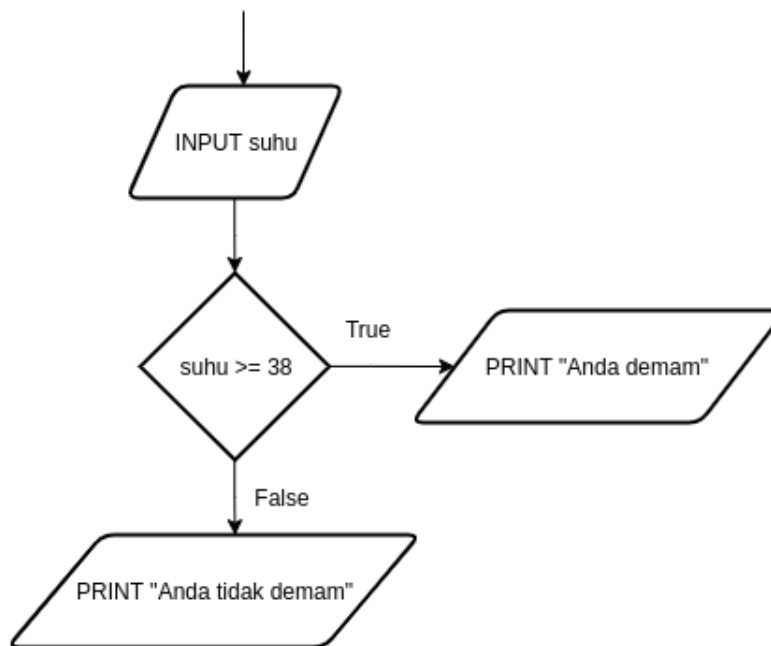
- Demam: 38, 39, 40, 41, ... (dan seterusnya). Dapat dinyatakan dalam bentuk $\text{suhu} \geq 38$.
- Tidak demam: 37, 36, 35, ... (dan seterusnya). Dapat dinyatakan dalam bentuk $\text{suhu} < 38$. Bisa juga dalam bentuk $\text{suhu} \leq 37$.

Program untuk memecahkan masalah ini adalah sebagai berikut:

```

1 suhu = int(input("Masukkan suhu tubuh: "))
2 if suhu >= 38:
3     print("Anda demam")

```



Gambar 3.5: Menentukan demam atau tidak berdasarkan suhu tubuh.

```
4 else:
5     print("Anda tidak demam")
```

Program tersebut jika dijalankan dan dicoba dengan beberapa input, hasilnya dapat dilihat pada Gambar 3.6.

The screenshot shows a terminal window with the title 'yuanlukito@linuxstation: ~/Documents'. It contains two runs of a Python script named 'demam.py'. In the first run, the user enters '42' for body temperature, and the program outputs 'Anda demam'. In the second run, the user enters '28', and the program outputs 'Anda tidak demam'. The prompt 'yuanlukito@linuxstation:~/Documents\$' is shown at the end of each line.

Gambar 3.6: Menentukan demam atau tidak berdasarkan suhu tubuh.

■ Contoh 3.2 Positif atau Negatif?

Buat sebuah program yang dapat menentukan apakah bilangan yang dimasukkan oleh pengguna merupakan bilangan positif, bilangan negatif atau nol. Contoh input dan output yang diharapkan dapat dilihat pada Tabel 3.3.

Jika anda perhatikan kemungkinan hasil yang harus ditangani, maka ada tiga kemungkinan: positif, negatif dan nol. Kriteria untuk masing-masing kemungkinan tersebut adalah:

- Positif jika input bilangan > 0
- Negatif jika input bilangan < 0

Tabel 3.3: Contoh input dan output yang diharapkan.

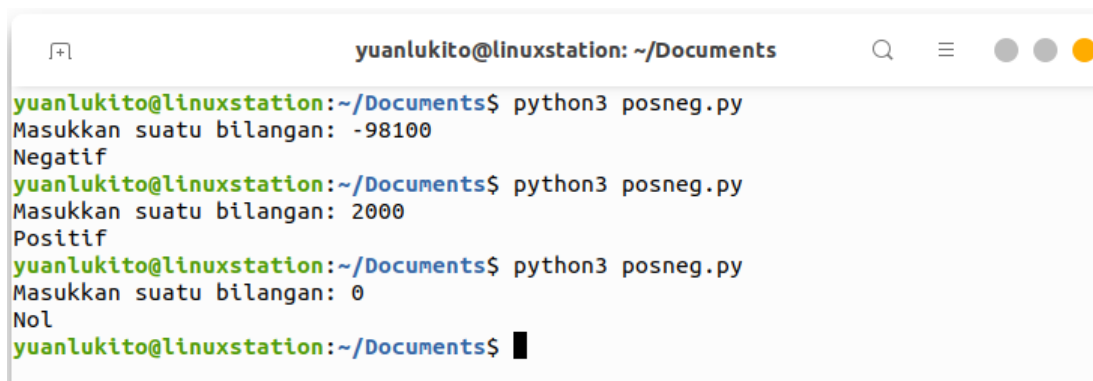
Input	Output yang diharapkan
28	Positif
-90	Negatif
2000	Positif
0	Nol

- Nol jika input bilangan == 0

Setelah anda berhasil menyusun kemungkinan-kemungkinan dan semua kriterianya, anda sudah bisa menyusun programnya seperti contoh berikut ini:

```
1 bilangan = int(input("Masukkan suatu bilangan: "))
2 if bilangan > 0:
3     print("Positif")
4 elif bilangan < 0:
5     print("Negatif")
6 elif bilangan == 0:
7     print("Nol")
```

Jika program tersebut dijalankan dan diberikan beberapa input yang berbeda, hasilnya sudah sesuai dengan yang diharapkan seperti yang dapat anda lihat pada Gambar 3.7.



```
yuanlukito@linuxstation: ~/Documents
yuanlukito@linuxstation:~/Documents$ python3 posneg.py
Masukkan suatu bilangan: -98100
Negatif
yuanlukito@linuxstation:~/Documents$ python3 posneg.py
Masukkan suatu bilangan: 2000
Positif
yuanlukito@linuxstation:~/Documents$ python3 posneg.py
Masukkan suatu bilangan: 0
Nol
yuanlukito@linuxstation:~/Documents$
```

Gambar 3.7: Hasil dari program Positif-Negatif.

■ Contoh 3.3 Nilai Terbesar

Buat sebuah program yang meminta tiga input bilangan dari pengguna, kemudian program akan menampilkan bilangan yang terbesar dari ketiga input bilangan tersebut.

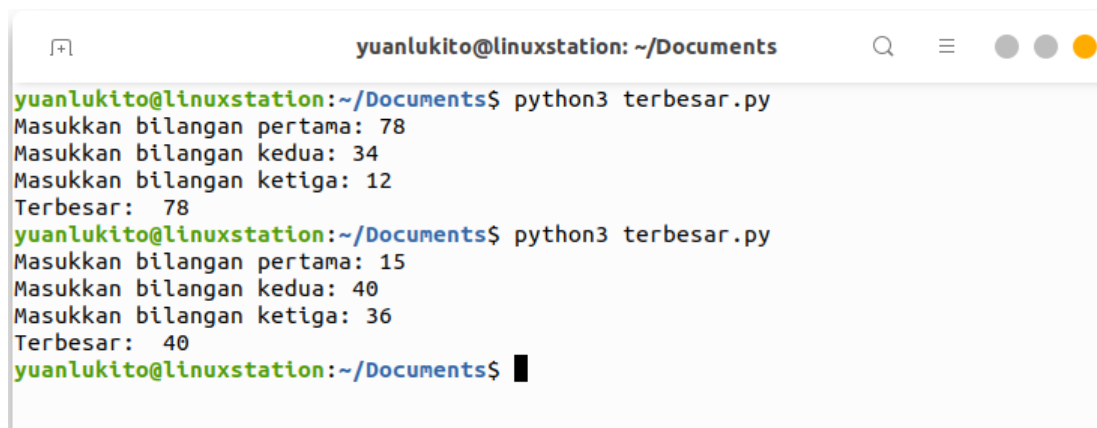
Untuk menangani masalah ini, kita andaikan terlebih dahulu ketiga bilangan tersebut adalah a, b dan c. Bilangan yang terbesar berarti ada tiga kemungkinan, yaitu bisa a, bisa b atau bisa c. Dari mana kita tahu bahwa a adalah yang terbesar? Kriterianya adalah sebagai berikut:

- A adalah yang terbesar jika $a > b$ dan $a > c$.
- B adalah yang terbesar jika $b > a$ dan $b > c$.
- C adalah yang terbesar jika $c > a$ dan $c > b$.

Setelah kita berhasil menentukan ketiga kriteria tersebut, maka kita dapat menyusun programnya sebagai berikut:

```
1 # input a, b dan c
2 a = int(input("Masukkan bilangan pertama: "))
3 b = int(input("Masukkan bilangan kedua: "))
4 c = int(input("Masukkan bilangan ketiga: "))
5
6 # secara berurutan tulis kriteria untuk a, b, dan c
7 if a > b and a > c:
8     print("Terbesar: ", a)
9 elif b > a and b > c:
10    print("Terbesar: ", b)
11 elif c > a and c > b:
12    print("Terbesar: ", c)
```

Jika program tersebut dijalankan dan dimasukkan beberapa macam input yang berbeda, hasilnya sudah sesuai dengan yang diharapkan. Anda dapat melihat hasilnya pada Gambar 3.8



The screenshot shows a terminal window titled 'yuanlukito@linuxstation: ~/Documents'. It displays two runs of a Python script named 'terbesar.py'. In the first run, the user inputs 78, 34, and 12, and the program outputs 'Terbesar: 78'. In the second run, the user inputs 15, 40, and 36, and the program outputs 'Terbesar: 40'.

```
yuanlukito@linuxstation: ~/Documents
yuanlukito@linuxstation:~/Documents$ python3 terbesar.py
Masukkan bilangan pertama: 78
Masukkan bilangan kedua: 34
Masukkan bilangan ketiga: 12
Terbesar: 78
yuanlukito@linuxstation:~/Documents$ python3 terbesar.py
Masukkan bilangan pertama: 15
Masukkan bilangan kedua: 40
Masukkan bilangan ketiga: 36
Terbesar: 40
yuanlukito@linuxstation:~/Documents$
```

Gambar 3.8: Hasil dari program mencari bilangan terbesar.

3.5 Latihan Mandiri

Latihan 3.1 Implementasikan penanganan kesalahan input pengguna dari program-program pada Contoh 3.1, 3.2 dan 3.3.

Latihan 3.2 Implementasikan percabangan pada Contoh 3.2 (Positif-Negatif) menggunakan ternary operator.

Latihan 3.3 Buatlah sebuah program yang dapat menampilkan jumlah hari dalam suatu bulan di tahun 2020. Program meminta pengguna memasukkan nomor bulan (1-12), kemudian program akan menampilkan jumlah hari pada bulan tersebut. Sebagai contoh, perhatikan input dan output berikut ini:

```
Masukkan bulan (1-12): 7
Jumlah hari: 31
```

Lengkapi program tersebut dengan penanganan kesalahan jika pengguna memasukkan bulan yang salah. Penanganan kesalahan dalam bentuk memunculkan pesan bahwa bulan yang diinputkan oleh pengguna tersebut tidak valid.

Latihan 3.4 Sebuah program meminta pengguna memasukkan ketiga panjang sisi suatu segitiga (berarti pengguna memasukkan tiga bilangan). Jika ketiga sisi segitiga tersebut semuanya sama, tampilkan pesan: "3 sisi sama". Jika hanya ada dua sisi yang sama panjang, tampilkan pesan "2 sisi sama". Jika tidak ada yang sama maka tampilkan pesan: "Tidak ada yang sama". Sebagai contoh, perhatikan input dan output berikut ini:

```
Masukkan sisi 1: 14
Masukkan sisi 2: 18
Masukkan sisi 3: 11
Tidak ada yang sama
```

```
Masukkan sisi 1: 22
Masukkan sisi 2: 22
Masukkan sisi 3: 22
3 sisi sama
```

```
Masukkan sisi 1: 8
Masukkan sisi 2: 9
Masukkan sisi 3: 8
2 sisi sama
```

Lengkapi program tersebut dengan penanganan kesalahan jika pengguna memasukkan input yang tidak valid.