



# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

<b>NIM</b>	<b>71230971</b>
<b>Nama Lengkap</b>	<b>James Marvin Santoso</b>
<b>Minggu ke / Materi</b>	<b>03 / Struktur Kontrol Percabangan</b>

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

### Boolean expression

Boolean expression digunakan untuk melakukan perbandingan antara nilai atau variabel dengan tujuan untuk menentukan apakah suatu kondisi benar atau salah. Dalam pemrograman, kita seringkali perlu melakukan pengecekan kondisi untuk mengontrol alur program. Dengan menggunakan boolean expression, kita dapat menuliskan kondisi-kondisi yang harus dipenuhi agar suatu blok kode dapat menjalankan kode. Bentuk tersebut dinamakan sebagai boolean expression, karena hasilnya hanya ada dua kemungkinan, yaitu True atau False, tergantung dari variabel pembelian.

Berikut contoh penggunaan dari boolean expression, antara lain :

```
contoh.py > ...
1  x = 5
2  y = 3
3  print(x > y)

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
True
```

Pada kode ini, terdapat dua variabel x dan y yang masing-masing diberi nilai 5 dan 3. Kemudian, terdapat pernyataan `print(x > y)` yang mencetak hasil perbandingan x dengan y. Karena nilai x (5) lebih besar dari nilai y (3), maka hasil perbandingan `x > y` akan bernilai True. Sehingga, hasil yang dicetak oleh pernyataan print adalah True.

### Logical operator

Logical operators digunakan untuk menggabungkan boolean values dalam boolean expressions. Mereka memungkinkan pembuatan struktur kondisional kompleks dengan menggabungkan beberapa kondisi. Dengan logical operators, alur program dapat dikontrol secara efisien. Ada tiga logical operator utama: and, or, dan not.

Berikut adalah contoh penggunaan ketiga operator tersebut, antara lain :

- a) Contoh penggunaan operator 'and' beserta contoh :

```
contoh.py > ...
1  x = 5
2  print(x > 0 and x < 10)
3  print(x < 0 and x < 10)

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
True
False
```

Pada kode ini, variabel `x` diberi nilai 5, kemudian digunakan operator `and` untuk mengevaluasi apakah `x` lebih besar dari 0 dan kurang dari 10, serta apakah `x` kurang dari 0 dan kurang dari 10.

b) Contoh penggunaan operator 'or' beserta contoh :

```
contoh.py > ...
1  x = 5
2  print(x > 0 or x < 0)
3  print(x > 0 or x > 10)
4  print(x < 0 or x > 10)

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
True
True
False
```

Kode tersebut menghasilkan nilai True, True, dan False secara berturut-turut, dengan menggunakan operator or untuk mengevaluasi apakah x lebih besar dari 0 atau kurang dari 0, serta apakah x lebih besar dari 0 atau lebih besar dari 10.

c) Contoh penggunaan operator 'not' beserta contoh :

```
contoh.py > ...
1  x = 5
2  print(not x > 0)
3  print(not x < 0)

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
False
True
```

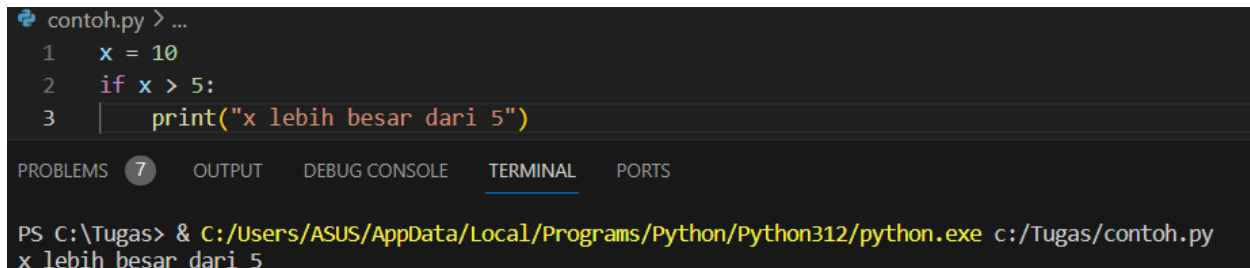
Kedua pernyataan tersebut menghasilkan nilai yang berturut-turut False dan True dengan menggunakan operator not untuk mengubah hasil evaluasi dari kondisi apakah x lebih besar dari 0 atau kurang dari 0.

## Bentuk-Bentuk Percabangan

### 1. Conditional (if)

Pernyataan if digunakan dalam Python untuk menjalankan kode tertentu hanya jika kondisi yang diberikan evaluasinya bernilai True. Ini memungkinkan kita untuk membuat kode yang responsif terhadap kondisi yang berubah.

Berikut adalah contoh dari Conditional, antara lain :



```
contoh.py > ...
1 x = 10
2 if x > 5:
3     print("x lebih besar dari 5")
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

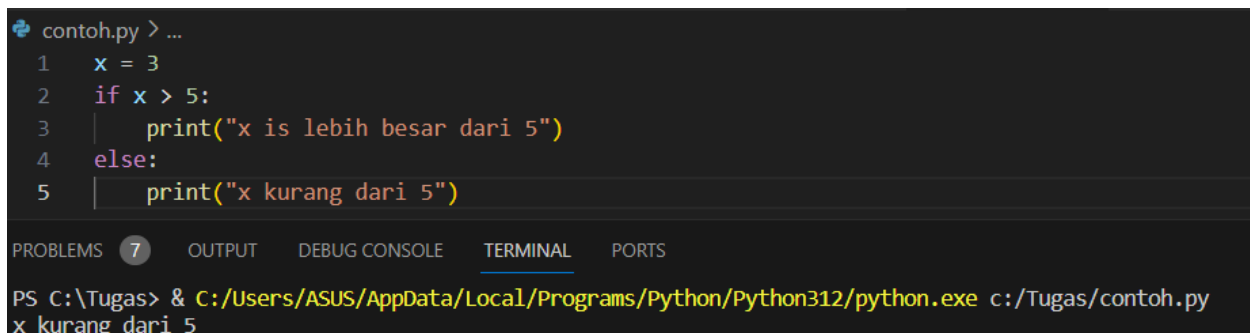
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py  
x lebih besar dari 5

Dalam contoh ini, if  $x > 5$ : mengevaluasi apakah nilai  $x$  lebih besar dari 5. Jika benar, pernyataan `print("x lebih besar dari 5")` dijalankan dan pesan "x lebih besar dari 5". Jika kondisi tersebut salah, kode di dalam pernyataan if bisa dijalankan.

### 2. Alternative (if-else)

Pernyataan alternative (if-else) digunakan dalam Python untuk memberikan pilihan kedua jika kondisi dalam pernyataan if tidak terpenuhi. Dengan kata lain, jika kondisi dalam pernyataan if bernilai False, maka kode dalam pernyataan else akan berjalan.

Berikut adalah contoh dari Alternative, antara lain :



```
contoh.py > ...
1 x = 3
2 if x > 5:
3     print("x is lebih besar dari 5")
4 else:
5     print("x kurang dari 5")
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py  
x kurang dari 5

Dalam contoh ini, jika nilai  $x$  kurang dari atau sama dengan 5, kondisi dalam pernyataan if akan bernilai False, sehingga blok kode dalam pernyataan else akan dijalankan. Sehingga, "x kurang dari 5" akan terlihat hasilnya.

### 3. Chained Conditional (if-elif-else)

Pernyataan chained conditional (if-elif-else) digunakan untuk mengevaluasi beberapa kondisi secara berurutan. Jika kondisi pertama (if) tidak terpenuhi, maka kondisi kedua (elif) dievaluasi, dan begitu seterusnya. Jika tidak ada kondisi yang terpenuhi, kode dalam pernyataan else akan dieksekusi.

Berikut adalah contoh dari Chained Conditional, antara lain :

```
contoh.py > ...
1  x = 10
2  if x > 15:
3      print("x lebih besar dari 15")
4  elif x > 5:
5      print("x lebih besar dari 5 tapi kurang dari 15")
6  else:
7      print("x kurang dari atau sama dengan 5")
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
x lebih besar dari 5 tapi kurang dari 15
```

Dalam contoh ini, jika nilai x adalah 10, kondisi pertama ( $x > 15$ ) akan bernilai False, maka kondisi kedua ( $x > 5$ ) akan dievaluasi. Karena kondisi kedua benar, maka blok kode dalam pernyataan elif akan dieksekusi dan pesan "x lebih besar dari 5 tapi kurang dari 15" akan dicetak.

## Penanganan Kesalahan Input Menggunakan Exception Handling

Exception handling digunakan dalam Python untuk menangani situasi-situasi di mana terjadi kesalahan atau kondisi yang tidak diharapkan selama eksekusi program. Dalam konteks penanganan kesalahan input, kita sering menggunakan try dan except untuk menangkap dan mengelola kesalahan yang terkait dengan input yang diberikan pengguna.

Sebagai contoh, pertimbangkan kasus di mana untuk memasukkan bilangan bulat, tetapi memasukkan input yang bukan bilangan bulat. Kita dapat menggunakan exception handling untuk menangani kesalahan ini. Berikut adalah contoh lengkapnya:

```
contoh.py > ...
1  try:
2      bilangan = int(input("Masukkan bilangan bulat: "))
3      print("Bilangan yang dimasukkan:", bilangan)
4  except ValueError:
5      print("Input tidak valid. Masukkan bilangan bulat yang benar.")
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Masukkan bilangan bulat: 4
Bilangan yang dimasukkan: 4
```

Pada contoh di atas, try digunakan untuk mencoba menjalankan kode di dalamnya. Jika memasukkan bilangan bulat, maka kode akan berjalan dengan lancar dan hasilnya akan dicetak. Namun, jika memasukkan input yang bukan bilangan bulat misalnya, huruf atau simbol, maka akan terjadi exception ValueError. Exception ini akan ditangkap oleh blok except, yang akan mencetak pesan kesalahan yang sesuai.

Dengan menggunakan exception handling, kita dapat memberikan umpan balik yang jelas kepada pengguna tentang kesalahan input yang mereka buat, sehingga memudahkan mereka untuk memperbaikinya.

## BAGIAN 2: LATIHAN MANDIRI

### SOAL 1.1

- Source Code :

```
Latihan 3,1.py > ...
1  try:
2      input_user = input("Masukkan suhu tubuh: ")
3      suhu = int(input_user)
4      if suhu >= 38 and suhu < 40:
5          print("Demam ringan")
6      elif suhu >= 40:
7          print("Segera konsultasikan dengan dokter, suhu tubuh tinggi")
8      else:
9          print("Suhu tubuh normal")
10 except ValueError:
11     print("Masukkan suhu tubuh dalam bentuk angka")
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,1.py"
Masukkan suhu tubuh: 36
Suhu tubuh normal
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,1.py"
Masukkan suhu tubuh: 40
Segera konsultasikan dengan dokter, suhu tubuh tinggi
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut meminta pengguna untuk memasukkan suhu tubuh, kemudian mencoba mengonversi input tersebut menjadi bilangan bulat. Jika konversi berhasil, kode akan memeriksa suhu tubuh untuk menentukan apakah termasuk dalam kategori demam ringan, demam tinggi, atau normal, lalu mencetak pesan sesuai. Jika konversi gagal karena input pengguna bukan angka, kode akan menangkap ValueError.

## SOAL 1.2

- Source Code :

```
Latihan 3,2.py > ...
1  try:
2      bilangan_str = input("Masukkan suatu bilangan: ")
3      bilangan = int(bilangan_str)
4      if bilangan == 0:
5          print("Bilangan nol")
6      elif bilangan > 0:
7          print("Bilangan positif")
8      else:
9          print("Bilangan negatif")
10 except ValueError:
11     print("Masukkan harus berupa bilangan bulat")
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,2.py"
Masukkan suatu bilangan: 1
Bilangan positif
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,2.py"
Masukkan suatu bilangan: -1
Bilangan negatif
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,2.py"
Masukkan suatu bilangan: 0
Bilangan nol
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut meminta pengguna untuk memasukkan suatu bilangan, kemudian mencoba mengonversi input tersebut menjadi bilangan bulat. Jika konversi berhasil, kode akan memeriksa apakah bilangan tersebut nol, positif, atau negatif, lalu mencetak pesan sesuai. Jika konversi gagal karena input pengguna bukan bilangan bulat, kode akan menangkap.



### SOAL 1.3

- Source Code :

```
Latihan 3,3.py > ...
1  try:
2      bulan = int(input("Masukkan bulan (1-12): "))
3      if bulan < 1 or bulan > 12:
4          print("Bulan yang dimasukkan tidak valid.")
5      else:
6          if bulan == 2:
7              jumlah_hari = 29
8          elif bulan in [1, 3, 5, 7, 8, 10, 12]:
9              jumlah_hari = 31
10         else:
11             jumlah_hari = 30
12         print(f"Jumlah hari pada bulan {bulan} adalah {jumlah_hari}.")
13 except ValueError:
14     print("Masukkan harus berupa angka.")
```

- Output :

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,3.py"
Masukkan bulan (1-12): 4
Jumlah hari pada bulan 4 adalah 30.
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut meminta pengguna untuk memasukkan angka yang mewakili bulan (1-12). Jika input valid, kode akan menentukan jumlah hari dalam bulan tersebut dan mencetaknya. Jika input tidak valid (bukan angka atau di luar rentang 1-12), kode akan menangkap ValueError.

#### SOAL 1.4

- Source Code :

```
Latihan 3,4.py > ...
1  try:
2      sisi_1 = float(input("Masukkan sisi 1: "))
3      sisi_2 = float(input("Masukkan sisi 2: "))
4      sisi_3 = float(input("Masukkan sisi 3: "))
5
6      if sisi_1 == sisi_2 == sisi_3:
7          print("3 sisi sama")
8      elif sisi_1 == sisi_2 or sisi_1 == sisi_3 or sisi_2 == sisi_3:
9          print("2 sisi sama")
10     else:
11         print("Tidak ada yang sama")
12 except ValueError:
13     print("Masukkan harus berupa angka.")
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 3,4.py"
Masukkan sisi 1: 2
Masukkan sisi 2: 2
Masukkan sisi 3: 2
3 sisi sama
PS C:\Tugas> 
```

- Penjelasan :

Kode tersebut meminta pengguna untuk memasukkan panjang sisi-sisi segitiga. Jika input valid, kode akan memeriksa apakah segitiga tersebut sama sisi, memiliki dua sisi yang sama, atau tidak ada sisi yang sama, lalu mencetak hasilnya. Jika input tidak valid, kode akan menangkap ValueError pesan.

- Link GitHub :

<https://github.com/Jamesmarvins/Tugas-Alpro-3.git>