
# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

| | |
|---|---|
| **NIM** | **71230970** |
| **Nama Lengkap** | **Gregorius Daniel Jodan Perminas** |
| **Minggu ke / Materi** | **03 / Struktur Kontrol Percabangan** |

**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.

## Boolean dan Logical Operator

Boolean Expression adalah suatu tipe data yang hanya memiliki dua nilai kemungkinan yaitu True (benar) atau False (salah). Biasanya dalam program tipe data Boolean ini sering digunakan dalam pengambilan keputusan atau pengujian dari sebuah kondisi. Contoh kasus misalnya ketika berbelanja seseorang mendapat Voucher Diskon 30%, namun Voucher tersebut hanya bisa dipakai dengan syarat tertentu. Voucher Diskon 30% tersebut hanya dapat dipakai bila pembelian minimal Rp 100.000,

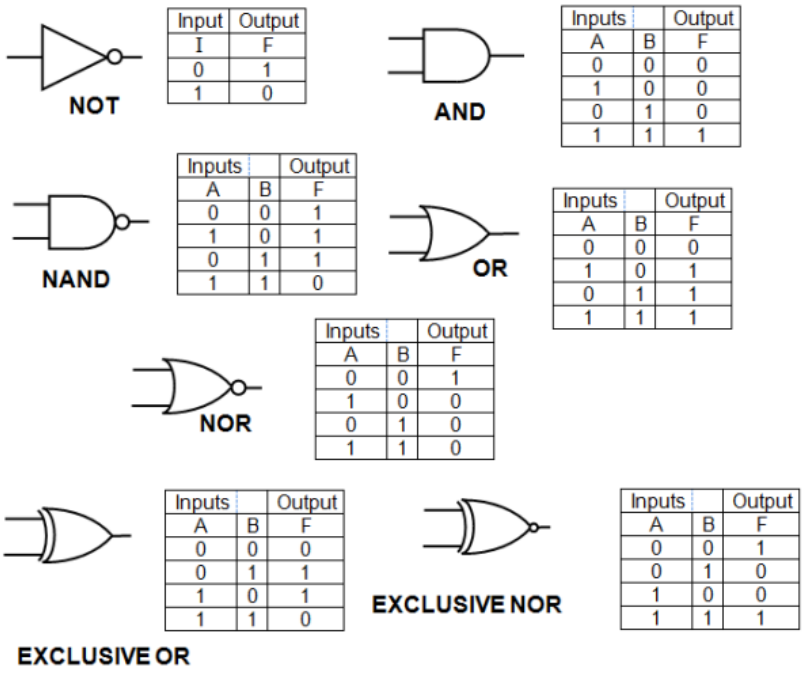
Bentuk matematisnya adalah **jika pembelian >= 100000 maka mendapat diskon**. Jawaban dari bentuk matematis ini hanya ada dua yaitu True atau False.

```
hargaBeli = int(input("Masukkan harga pembelian dalam Rp: "))
if hargaBeli >= 100000:
    print(True)
else:
    print(False)
```

```
PS D:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere> python -u "d:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere\
modul3.py"
Masukkan harga pembelian dalam Rp: 100000
True
PS D:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere> python -u "d:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere\
modul3.py"
Masukkan harga pembelian dalam Rp: 50000
False
PS D:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere>
```

*Gambar 1 - Program Python Boolean Expression*

Operator Boolean adalah operator yang mengambil input dari Boolean dan mengembalikan hasil Boolean. Karena nilai Python Boolean hanya memiliki dua opsi yang memungkinkan yaitu True atau False, maka perlu ada hasil yang ditetapkan untuk setiap kemungkinan kombinasi masukan. Hal ini biasanya disebut tabel kebenaran. Di sini tabel kebenaran dapat diartikan sebagai tabel yang berisi kombinasi-kombinasi variabel masukan (input) yang menghasilkan keluaran (output) yang logis. Di bawah ini merupakan contoh dari beberapa tabel kebenaran:




*Gambar 2 - Beberapa Tabel Kebenaran*

Berikut ini beberapa operator perbandingan yang dapat digunakan untuk boolean expression

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? |

*Gambar 3 - Tabel Operator Perbandingan*

Beberapa hal penting yang perlu diperhatikan dalam penyusunan boolean expression:

- Bentuk boolean expression hanya dua yaitu True atau False
- Perhatikan operator dengan baik seperti minimum, maksimum, kurang dari, tidak sama, dll
- Perhatikan dengan baik variabel apa yang perlu dibandingankan sesuai dengan permasalahan

## Bentuk-bentuk Percabangan

Pada umumnya ada tiga bentuk percabangan di Python yaitu Conditional, Alternative, dan Chained Conditional. Conditional atau Percabangan adalah aliran yang mengontrol kode program Python berdasarkan pengujian pernyataan bersyarat. Beberapa statement yang sering digunakan dalam Percabangan Python yaitu “if”, ”if-else”, dan “if-elif-else”

Contoh bentuk Conditional:

```
if nilai > 70:
    print("Anda Lulus!")
```

*Gambar 4 - Program Python Conditional*

Penjelasan: Bagian condition ini biasanya berperan sebagai penentu dari struktur percabangan. Jika condition terpenuhi maka kode program akan dijalankan, namun jika tidak maka kode program tidak akan dijalankan. Maka program hanya akan menyatakan seseorang lulus jika nilainya di atas 70.

Contoh bentuk Alternative Conditional:

```
if nilai > 70:
    print("Anda Lulus!")
else:
    print("Anda Tidak Lulus, Tapi Boleh Mengulang!")
```

*Gambar 5 - Program Python Alternative Conditional*

Penjelasan: Bentuk percabangan ini memiliki dua alternatif langkah yang bisa dijalankan. Ketika nilai di atas 70 maka program akan menampilkan ”Anda Lulus!” namun jika kondisi atas tidak terpenuhi maka alternatif langkah kedua akan dijalankan yaitu menampilkan tulisan ”Anda Tidak Lulus, Tapi Boleh Mengulang!”

Contoh bentuk Chained Conditional:

```
if nilai > 70:
    print("Anda Lulus!")
elif nilai > 30:
    print("Anda Tidak Lulus, Tapi Boleh Mengulang!")
else:
    print("Anda Tidak Lulus, dan Tidak Boleh Mengulang!")
```

*Gambar 6 - Program Python Chained Conditional*

Penjelasan: Bentuk percabangan ini digunakan jika alternatif langkah yang ingin dijalankan lebih dari dua. Sebagai contoh program diatas akan menampilkan ”Anda Lulus!” jika nilai di atas 70, kemudian program

akan menampilkan ”Anda Tidak Lulus, Tapi Boleh Mengulang!” untuk nilai di atas 30 sampai 70, lalu alternatif langkah ketiga yaitu program akan menampilkan ”Anda Tidak Lulus, dan Tidak Boleh Mengulang!” ketika nilai yang didapat 30 ke bawah.

## Penanganan Kesalahan Input dari Pengguna

Ketika membuat sebuah program, kita juga perlu memperhatikan potensi kesalahan yang mungkin terjadi yang membuat program tidak berjalan sebagaimana mestinya. Sebagai contoh program di bawah ini:

```
nilai = int(input("Masukkan nilai Anda: "))

if nilai > 70:
    print("Anda Lulus!")
elif nilai > 30:
    print("Anda Tidak Lulus, Tapi Boleh Mengulang!")
else:
    print("Anda Tidak Lulus, dan Tidak Boleh Mengulang!")
```

```
Masukkan nilai Anda: 75
Anda Lulus!
```

*Gambar 7 - Program Python Tanpa Try-Except*

Ini adalah asumsi program berjalan dengan baik dimana pengguna memasukkan input sesuai dengan permintaan program yaitu dalam bentuk integer. Namun apa yang terjadi jika input salah atau tidak sesuai dengan yang kita harapkan, contohnya ada pada gambar berikut ini:

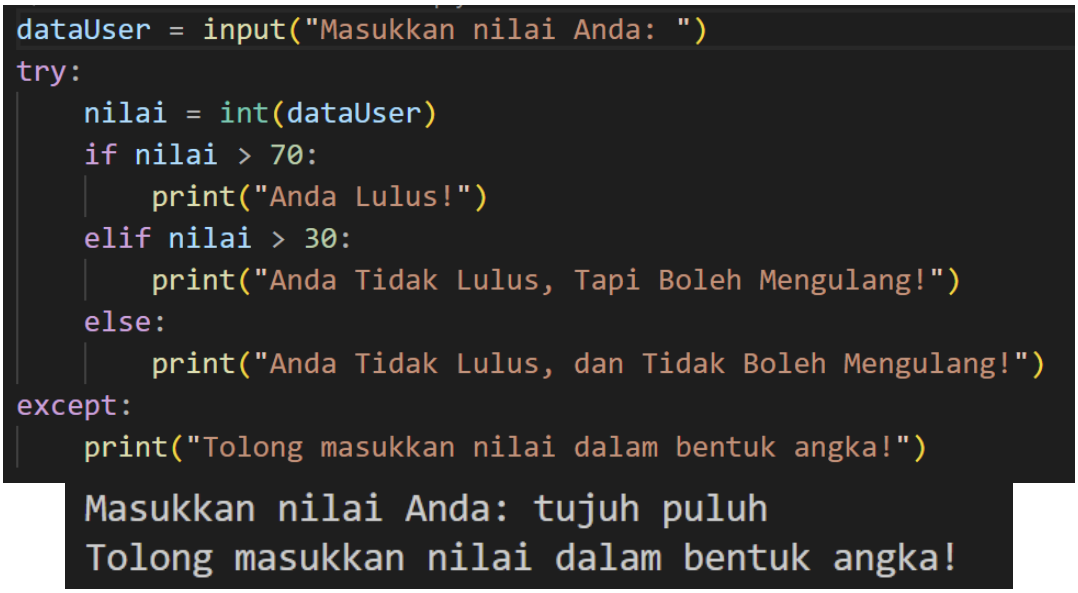
```
Masukkan nilai Anda: delapan puluh
Traceback (most recent call last):
  File "d:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere\modul3.py",
line 1, in <module>
    nilai = int(input("Masukkan nilai Anda: "))
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'delapan puluh'
```

*Gambar 8 - Hasil Program Tanpa Try-Except*

Ketika input yang dimasukkan sesuai maka program akan berhenti berjalan dan menampilkan pesan error seperti di atas. Untuk mengatasi hal ini salah satu cara yang dapat digunakan adalah menggunakan Try dan Except dalam program.

Metode Try-Except dapat membantu program tetap berjalan dengan lancar tanpa berhenti secara tiba-tiba akibat error. Ketika kamu menggunakan try-except, Python akan mencoba menjalankan kode dalam blok “try”. Jika kode tersebut berjalan dengan sukses, maka program akan melanjutkan eksekusi

setelah blok “try”. Namun, jika terjadi kesalahan selama eksekusi di dalam blok “try”, Python akan melompat ke blok “except” yang sesuai dengan jenis kesalahan yang terjadi. Dalam blok “except”, kamu dapat menulis kode untuk menangani exception dan memberikan tanggapan yang sesuai. Ini sangat membantu dalam menghindari crash program dan memberikan kesempatan untuk melakukan tindakan perbaikan. Contoh penggunaannya bisa dilihat pada gambar berikut ini:




*Gambar 9 - Program Python dengan Try-Except*

Kelebihan pada program ini yaitu walaupun input yang dimasukkan tidak sesuai, namun program dapat tetap berjalan dan tidak menampilkan pesan error. Hal ini sangat membantu dalam menghindari crash pada program dan memberikan kesempatan bagi *user* untuk melakukan tindakan perbaikan.

# 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/PrakAlPro3_A_71230970.git

## SOAL 1

```
# Latihan Mandiri 3.1
```

```
# Contoh 3.1

dataInput = input("Masukkan suhu tubuh (Celcius): ")
try:
    suhu = int(dataInput)
    if suhu >= 38:
        print("Anda demam")
    else:
        print("Anda tidak demam")
except:
    print("Maaf tolong masukkan suhu tubuh dalam bentuk angka")
```

```
Masukkan suhu tubuh (Celcius):  tiga puluh
Maaf tolong masukkan suhu tubuh dalam bentuk angka
```

*Gambar 10 - Penggunaan Try-Except Pada Contoh 3.1*

```
# Contoh 3.2

dataInput = input("Masukkan suatu bilangan: ")
try:
    bilangan = int(dataInput)
    if bilangan > 0:
        print("Positif")
    elif bilangan < 0:
        print("Negatif")
    elif bilangan == 0:
        print("Nol")
except:
    print("Maaf tolong masukkan bilangan dalam bentuk angka")
```

```
Masukkan suatu bilangan:  lima
Maaf tolong masukkan bilangan dalam bentuk angka
```

*Gambar 11 - Penggunaan Try-Except Pada Contoh 3.2*

```
# Contoh 3.3

# input a, b dan c
input1 = input("Masukkan bilangan pertama: ")
input2 = input("Masukkan bilangan kedua: ")
input3 = input("Masukkan bilangan ketiga: ")

# secara berurutan tulis kriteria untuk a, b, dan c
try:
    a = int(input1)
    b = int(input2)
    c = int(input3)
    if a > b and a > c:
        print("Terbesar: ", a)
    elif b > a and b > c:
        print("Terbesar: ", b)
    elif c > a and c > b:
        print("Terbesar: ", c)
except:
    print("Maaf tolong masukkan bilangan dalam bentuk angka")
```

```
Masukkan bilangan pertama:  5
Masukkan bilangan kedua:  6
Masukkan bilangan ketiga:  tujuh
Maaf tolong masukkan bilangan dalam bentuk angka
```

*Gambar 12 - Penggunaan Try-Except Pada Contoh 3.3*

Pada program-program Contoh 3.1, 3.2 dan 3.3 penanganan kesalahan input dari pengguna sudah diimplementasikan melalui metode Try-Except. Pertama program akan membebaskan user untuk memberi input dalam bentuk apapun. Setelah itu program akan melakukan pengecekan pada bagian Try untuk memastikan apakah tipe data yang dimasukkan sudah sesuai dengan yang diharapkan, bisa dalam bentuk *integer, float, strings,* dan lain-lain tergantung kebutuhan. Bila sudah sesuai maka program akan menjalankan blok Try, namun bila tidak maka program akan menjalankan blok Except. Hal ini membuat program dapat terus berjalan dengan baik meskipun terdapat kesalahan *input* dan menghindari terjadinya *crash.*

## SOAL 2

```
# Latihan Mandiri 3.2
```

```
bilangan = int(input("Masukkan suatu bilangan: "))
print("Positif" if bilangan > 0 else "Negatif" if bilangan < 0 else "Nol")
```

```
Masukkan suatu bilangan:  1
Positif
```

*Gambar 13 - Penggunaan Ternary Operator pada Contoh 3.2*

Ternary Operator merupakan salah satu sintaks alternatif untuk menuliskan percabangan pada Python, sehingga hal ini tidak akan merubah apapun hanya beda cara penulisan program saja. Pada soal ini saya diminta untuk mengubah dari bentuk Chained Conditional seperti pada gambar 11 ke dalam bentuk Ternary. Bilangan akan dikatakan positif bila lebih dari 0, kemudian negatif bila kurang dari 0, kemudian sisanya berarti nol.

Aturan sintaks dalam Ternary Operator yaitu:

*Syntax: [on_true] if [expression] else [on_false]*

Sehingga hasil programnya tampak seperti pada gambar 13 di atas. Ternary Operator ini memang membuat program tampak menjadi lebih ringkas, namun sebagai *programmer* kita harus berhati-hati ketika membacanya karena sedikit lebih sulit daripada metode pada gambar 11.

## SOAL 3

```
# Latihan Mandiri 3.3
```

```
# Function untuk menentukan jumlah hari
def jumlahHari (inputBulan):
    if inputBulan in [1,3,5,7,8,10,12]: # List bulan dengan jumlah hari 31
        return "31"
    elif inputBulan == 2: # List bulan dengan jumlah hari 29
        return "29"
    elif inputBulan in [4,6,9,11]: # List bulan dengan jumlah hari 30
        return "30"
    else:
        return "Bulan yang diinput tidak valid!" # Penanganan ketika input di luar interval 1-12

try:
    inputData = int(input("Masukkan bulan 1-12: "))
    print (jumlahHari(inputData))
except:
    print("Masukkan bulan dalam bentuk angka!")
```

*Gambar 14 - Program Latihan 3.3*

```
Masukkan bulan 1-12:  1
31
```

*Gambar 15 - Hasil Program Latihan 3.3*

```
Masukkan bulan 1-12:  14
Bulan yang diinput tidak valid!
```

*Gambar 16 - Ketika Bulan Yang Dimasukkan Diluar Interval 1-12*

```
Masukkan bulan 1-12:  tujuh
Masukkan bulan dalam bentuk angka!
```

*Gambar 17 - Penanganan Kesalahan Input Pada Program Latihan 3.3*

Program pada gambar 14 di atas dapat menampilkan jumlah hari dalam suatu bulan di tahun 2020. Di sini bulan dikelompokkan berdasarkan jumlah hari, misal bulan Januari, Maret, Mei, Juli, Agustus, November, Desember memiliki 31 hari, maka ketujuh elemen bulan ini akan dimasukkan dalam satu list kemudian diberikan return 31 yang artinya terdapat 31 hari untuk ketujuh bulan tersebut. Hal ini juga berlaku untuk list yang lain.

Pada program ini terdapat 2 penanganan kesalahan. Penanganan kesalahan yang pertama terdapat dalam function ”jumlahHari” yaitu ketika pengguna memasukkan bulan yang berada diluar dari interval 1-12 maka program akan menampilkan ”Bulan yang diinput tidak valid!” lalu penanganan kesalahan yang kedua terdapat pada metode Try-Except. Ketika *input* yang dimasukkan oleh pengguna dalam bentuk *integer* maka program akan menjalankan blok Try, namun apabila tidak maka program akan menjalankan blok Except.

## SOAL 4

```
# Latihan Mandiri 3.4
```

```
# Function untuk mengecek sisi
def pengecekanSisi(x,y,z):
    if x == y == z:
        return "3 sisi sama"
    elif x == y or x == z or y == z:
        return "2 sisi sama"
    else:
        return "Tidak ada yang sama"

try:
    sisi1 = int(input("Masukkan sisi 1: "))
    sisi2 = int(input("Masukkan sisi 2: "))
    sisi3 = int(input("Masukkan sisi 3: "))
    print(pengecekanSisi(sisi1,sisi2,sisi3))
except:
    print("Maaf tolong masukkan input dalam format angka")
```

*Gambar 18 - Program Latihan 3.4*

```
Masukkan sisi 1:  5
Masukkan sisi 2:  5
Masukkan sisi 3:  5
3 sisi sama
```

*Gambar 19 - Hasil Ketika Ketiga Sisi Sama*

```
Masukkan sisi 1:  13
Masukkan sisi 2:  8
Masukkan sisi 3:  13
2 sisi sama
```

*Gambar 20 - Hasil Ketika Dua Sisi Sama*

```
Masukkan sisi 1:  15
Masukkan sisi 2:  17
Masukkan sisi 3:  3
Tidak ada yang sama
```

*Gambar 21 - Hasil Ketika Tidak Ada Sisi Yang Sama*

```
Masukkan sisi 1:  tujuh
Maaf tolong masukkan input dalam format angka
```

*Gambar 22 - Penanganan Kesalahan Input Program Latihan 3.4*

Program pada gambar 18 di atas digunakan untuk menguji apakah dari ketiga sisi yang diberikan pengguna terdapat tiga sisi yang sama, dua sisi yang sama, atau bahkan tidak ada yang sama. Pengecekan atau pengujian ini terdapat pada function ”pengecekanSisi.” Ketika ketiganya sama berarti syaratnya adalah x == y == z. Ketika hanya ada dua sisi yang sama maka syaratnya adalah x == y atau x == z atau y == z, ketika salah satu syarat terpenuhi maka sudah bisa dianggap dua sisi sama. Kemudian ketika kedua syarat di atas tidak terpenuhi maka sudah pasti tidak ada sisi yang sama. Program ini juga dilengkapi dengan penanganan kesalahan *input* oleh pengguna, sehingga ketika pengguna memasukkan *input* dalam bentuk *strings* program tidak akan *crash* atau *error* namun menampilkan pesan untuk meminta pengguna memasukkan *input* dalam bentuk integer.