Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2025/2026

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.

NIM	**71251230**
Nama Lengkap	**Okky Alexander**
Minggu ke / Materi	**03 / Struktur Kontrol Percabangan**

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2026

MATERI BOOLEAN EXPRESSION DAN LOGICAL OPERATOR

Penggunaan voucher diskon dengan syarat belanja minimal Rp100.000 merupakan contoh penerapan Boolean Expression dalam Python, yang direpresentasikan melalui kode pembelian >= 100000. Ekspresi ini berfungsi untuk mengevaluasi apakah suatu kondisi terpenuhi, di mana hasilnya hanya memiliki dua kemungkinan nilai, yaitu True atau False. Dalam mode interaktif Python, hasil evaluasi tersebut akan muncul secara otomatis berdasarkan nilai variabel yang dimasukkan.

Tabel 4.1: 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?

Penyusunan Boolean expression yang tepat sangat bergantung pada kemampuan menganalisis masalah dan memilih operator perbandingan yang sesuai. Hal utama yang harus diperhatikan adalah hasil akhirnya yang selalu biner (True atau False) serta kejelian dalam menerjemahkan kata kunci kontekstual (seperti "minimum", "maksimum", atau "tidak lebih dari") ke dalam simbol matematika yang benar. Selain itu, beberapa ekspresi dapat dikombinasikan menjadi logika yang lebih kompleks menggunakan Logical Operator seperti and, or, dan not.

Tabel 4.2: Operator-operator perbandingan (comparison).

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

Penggunaan operator logika seperti and dan or memungkinkan penggabungan beberapa syarat menjadi satu kesatuan *boolean expression*. Pada kasus wahana *Rollercoaster*, operator and digunakan karena kedua syarat—usia minimal 10 tahun dan tinggi minimal 110 cm—harus terpenuhi secara bersamaan (usia >= 10 and tinggi >= 110). Sebaliknya, operator or digunakan dalam logika pemberian diskon di mana cukup salah satu syarat yang terpenuhi, baik status sebagai member atau total pembelian di atas Rp500.000 (member == True or pembelian > 500000).

MATERI BENTUK-BENTUK PERCABANGAN

Tiga struktur utama percabangan dalam Python untuk mengatur alur program berdasarkan kondisi tertentu. Pertama adalah conditional sederhana menggunakan if yang hanya menjalankan perintah jika syarat terpenuhi, seperti pemberian sertifikat bagi nilai di atas 70. Kedua, alternative conditional menggunakan if-else untuk menangani dua kemungkinan pilihan, misalnya menentukan status "Lulus" atau "Tidak Lulus". Ketiga, chained conditional dengan struktur if-elif-else digunakan untuk menangani lebih dari dua kemungkinan hasil, seperti pada sistem diskon bertingkat yang sering menggabungkan beberapa *boolean expression* dengan operator logika and untuk menentukan rentang nilai tertentu. Selain ketiga bentuk standar tersebut, Python juga menyediakan ternary operator sebagai cara ringkas untuk menuliskan instruksi if-else dalam satu baris kode guna meningkatkan efisiensi penulisan program.

Contoh contoh bentuk percabangan:

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

Gambar 4.3: contoh percabangan sederhana menjalankan jika nilai lebih tinggi dari 70.

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

Gambar 4.3: contoh percabangan code mencari tahu apakah lulus atau tidak.

```
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
```

Gambar 4.3: contoh percabangan code mencari diskon.

MATERI PENANGANAN KESALAHAN INPUT MENGGUNAKAN EXCEPTION HANDLING

Penting bagi pengembang untuk mengantisipasi potensi kesalahan saat memproses input dari pengguna agar program tetap berjalan dengan stabil. Sebagai ilustrasi, pertimbangkan sebuah program klasifikasi kelompok usia yang meminta pengguna memasukkan umur mereka. Program ini dirancang untuk menentukan apakah seseorang tergolong dalam kategori balita, kanak-kanak, remaja, dewasa, atau lansia berdasarkan kriteria rentang usia tertentu.

Berikut adalah pembagian kategori usia yang digunakan dalam logika program tersebut:

- **Balita:** 0 hingga 5 tahun.
- **Kanak-kanak:** 6 hingga 11 tahun.
- **Remaja:** 12 hingga 25 tahun.
- **Dewasa:** 26 hingga 45 tahun.
- **Lansia:** Di atas 45 tahun.

Program tersebut adalah sebagai berikut:

```python
usia = int(input("Masukkan usia anda: "))
if usia <= 5:
    print("Balita")
elif usia >= 6 and usia <= 11:
    print("Kanak-kanak")
elif usia >=12 and usia <= 25:
    print("Remaja")
elif usia >= 26 and usia <= 45:
    print("Dewasa")
elif usia > 45:
    print("Lansia")
```

Gambar 4.4: Program kategori usia.

Setelah dijalankan berulang kali dengan berbagai variasi input, program tersebut terbukti memberikan hasil yang akurat sesuai dengan logika yang telah ditetapkan. Dokumentasi keberhasilan eksekusi ini dapat diamati pada Gambar 4.5. Adapun instruksi utama yang berfungsi untuk menerima data masukan dari pengguna terletak pada baris pertama, yakni melalui perintah usia = int(input("Masukkan usia anda: ")).

```
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: 33
Dewasa
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: 7
Kanak-kanak
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: 68
Lansia
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: 18
Remaja
```

Gambar 4.5: 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 4.3 yang menunjukkan pengguna memasukkan input yang tidak sesuai.

```
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: dua puluh
Traceback (most recent call last):
  File "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py", line 1, in <module>
    usia = int(input("Masukkan usia anda: "))
ValueError: invalid literal for int() with base 10: 'dua puluh'
PS E:\Koolyeah\github\PrakAlPro-71251230>
```

Untuk mengatasi masalah input yang tidak valid, Python menyediakan mekanisme penanganan kesalahan menggunakan blok **try** dan **except**. Metode ini memungkinkan program untuk "mencoba" menjalankan baris kode yang berisiko galat—seperti konversi string ke angka—dan menyiapkan rencana cadangan jika kesalahan benar-benar terjadi. Dengan cara ini, alih-alih berhenti mendadak dengan pesan *error* teknis, program dapat memberikan respon yang lebih terkendali dan ramah pengguna. Penerapan teknik ini pada kasus klasifikasi usia dapat dipelajari melalui struktur kode berikut:

```python
inputuser = input("Masukkan usia anda: ")
try:
    usia = int(inputuser)
    if usia <= 5:
        print("Balita")
    elif usia >= 6 and usia <= 11:
        print("Kanak-kanak")
    elif usia >=12 and usia <= 25:
        print("Remaja")
    elif usia >= 26 and usia <= 45:
        print("Dewasa")
    elif usia > 45:
        print("Lansia")
except:
    print("Anda salah memasukkan input usia")
```

Gambar 4.6: Program kategori usia menggunakan try dan except.

Jika dijalankan dan diberi input yang tidak sesuai, program tidak mengalami kesalahan seperti sebelumnya. Hasilnya jika dijalankan dapat dilihat pada Gambar 4.7.

```
PS E:\Koolyeah\github\PrakAlPro-71251230> python -u "e:\Koolyeah\github\PrakAlPro-71251230\Minggu 4\Latihan 4.1 .py"
Masukkan usia anda: dua puluh
Anda salah memasukkan input usia
PS E:\Koolyeah\github\PrakAlPro-71251230>
```

Gambar 4.7: Hasil program kategori usia.

Source Github : https://github.com/tuangkeman/71251230_Okky.git

LATIHAN 4.1

```python
# SOAL 4.1
try:
    derajat = int(input("Masukkan suhu tubuh: "))

    if derajat >= 38:
        print("Anda demam")
    else:
        print("Anda tidak demam")

except ValueError:
    print("Anda salah memasukan input, tolong masukan input yang benar kembali ")
```

Gambar 4.8: kode soal 4.1.

Tanpa try...except, jika pengguna menginput "Tiga Puluh", program akan langsung mati. Dengan kode kamu, program akan memberikan instruksi yang sopan: *"Anda salah memasukan input, tolong masukan input yang benar kembali"*.

```python
# SOAL 4.2
try:
    bilangan = int(input("Masukkan suatu bilangan: "))
    if bilangan > 0:
        print("Positif")
    elif bilangan < 0:
        print("Negatif")
    elif bilangan == 0:
        print("Nol")

except ValueError:
    print("Eror, coba masukan input dengan benar")
```

Gambar 4.9: kode soal 4.2.

Sama seperti sebelumnya, ini memastikan bahwa variabel bilangan benar-benar berisi angka agar operasi perbandingan (> 0 atau < 0) bisa berjalan tanpa error sistem.

```python
# SOAL 4.3
try:
    o = int(input("Masukkan bilangan pertama: "))
    p = int(input("Masukkan bilangan kedua: "))
    m = int(input("Masukkan bilangan ketiga: "))

    if o > p and o > m:
        print("Terbesar: ", o)
    elif p > o and p > m:
        print("Terbesar: ", p)
    elif m > o and m > p:
        print("Terbesar: ", m)

except ValueError:
    print("Input kamu salah, coba lagi ")
```

Gambar 4.10: kode soal 4.3.

Di sini penanganan kesalahan menjadi lebih krusial karena ada tiga input sekaligus. Jika salah satu saja dari a, b, atau c bukan angka, maka blok except akan langsung menangkapnya.

LATIHAN 4.2

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

Gambar 4.11: kode soal 4.2.

Ternary Operator berantai untuk meringkas struktur if-elif-else menjadi satu baris. Logikanya bekerja dari kiri ke kanan: program pertama kali mengecek apakah angka > 0 untuk memberikan hasil "Positif". Jika salah, program akan mengecek kondisi kedua yaitu angka < 0 untuk hasil "Negatif". Jika kedua kondisi tersebut tidak terpenuhi (berarti angka adalah 0), maka bagian else terakhir akan memberikan hasil "Nol".

Penggunaan teknik ini membuat kode lebih ringkas dan efisien secara penulisan, namun tetap membutuhkan blok try-except agar program tidak *crash* saat pengguna memasukkan input berupa huruf atau simbol.

LATIHAN 4.3

```
def jmlrek(p):
    if p < 1 or p > 12:
        return None
    m = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    return m[p - 1]
try:
    x = int(input("Masukkan bulan dalam bentuk angka: "))
    z = jmlrek(x)

    if z is None:
        print("Bulan tidak valid.")
    else:
        print("Jumlah hari:", z)
except ValueError:
    print("Input tidak valid. Masukkan angka antara 1 sampai 12.")
```

Gambar 4.12: kode soal 4.3.

Kode ini berfungsi untuk menentukan jumlah hari dalam satu bulan berdasarkan angka (1-12) menggunakan bantuan List sebagai tabel referensi. Di dalam fungsi jmlrek(p), program terlebih dahulu

memvalidasi apakah angka yang dimasukkan berada dalam rentang bulan yang sah; jika tidak, fungsi akan mengembalikan nilai None. Jika angka valid, program mengambil data hari dari list m menggunakan indeks p - 1 karena urutan list dalam Python dimulai dari angka 0. Seluruh proses ini dibungkus dalam blok try...except untuk memastikan bahwa jika pengguna memasukkan karakter selain angka (seperti huruf), program tidak akan *crash*, melainkan memberikan pesan peringatan yang informatif bahwa input tersebut tidak valid.

LATIHAN 4.4

```
try:
    sisipertama = int(input("Masukan angka untuk sisi 1: "))
    sisidua = int(input("Masukan angka untuk sisi 2: "))
    akhirrek = int(input("Masukan angka untuk sisi 3: "))
    if sisipertama == sisidua == akhirrek:
        print("3 sisi sama")
    elif sisipertama == sisidua or sisipertama == akhirrek or sisidua == akhirrek:
        print("2 sisi sama")
    else:
        print("Tidak ada yang sama")
except ValueError:
        print("Anda salah memasukan input, Tolong masukan input kembali dengan benar")
```

Gambar 4.12: kode soal 4.4.

Kode ini berfungsi untuk mengelompokkan jenis segitiga berdasarkan kesamaan panjang ketiga sisinya melalui input pengguna. Di dalam blok try, program mencoba mengubah input menjadi angka bulat, lalu mengevaluasinya secara bertingkat: pertama menggunakan perbandingan berantai untuk mendeteksi jika ketiga sisi identik (3 sisi sama), kemudian menggunakan operator logika or untuk mengecek jika hanya ada dua sisi yang kembar (2 sisi sama), dan terakhir menangani kondisi jika semua sisi berbeda. Seluruh alur ini dilindungi oleh blok except ValueError agar jika pengguna salah memasukkan karakter (seperti huruf), program tidak akan *crash* melainkan menampilkan pesan peringatan yang informatif.