

Modul 3: Pernyataan Kondisional

3.1 Pendahuluan

Pernyataan kondisional dalam pemrograman Python adalah konsep yang memungkinkan program untuk membuat keputusan berdasarkan kondisi tertentu. Pada dasarnya, pernyataan kondisional berfungsi untuk memeriksa apakah suatu kondisi atau ekspresi tertentu bernilai benar (`True`) atau salah (`False`). Jika kondisi tersebut benar, maka program akan menjalankan perintah atau blok kode yang terkait dengan kondisi tersebut. Sebaliknya, jika kondisi tersebut salah, program akan menjalankan perintah atau blok kode lain yang biasanya diatur dalam cabang alternatif. Pada akhir modul ini, diharapkan Anda akan memahami:

1. Pernyataan **if, elif, dan else**.
2. Cara menggunakan **kondisional bersarang**.
3. Menerapkan **logika kondisional** dengan contoh kehidupan nyata.

3.2 Pernyataan Kondisional

Pernyataan kondisional merupakan suatu mekanisme yang memungkinkan program untuk membuat keputusan berdasarkan kondisi atau syarat tertentu yang diberikan. Dalam kehidupan sehari-hari, kita seringkali membuat keputusan berdasarkan pertanyaan atau situasi yang membutuhkan jawaban "ya" atau "tidak". Misalnya, kita mungkin bertanya, "Apakah langit cerah hari ini?", dan berdasarkan jawabannya kita membuat keputusan apakah akan keluar rumah atau tidak. Dalam pemrograman, pernyataan kondisional berfungsi dengan cara yang serupa: ia membantu program menentukan tindakan yang harus diambil berdasarkan hasil dari suatu kondisi yang diuji. Pernyataan kondisional sangat berguna dalam berbagai aplikasi pemrograman, mulai dari aplikasi sederhana hingga aplikasi yang lebih kompleks. Di dunia nyata, ini sering digunakan untuk memeriksa validitas input, mengelola alur program, dan menangani situasi yang memerlukan keputusan berdasarkan beberapa kondisi. Di dalam pemrograman Python menyediakan konstruksi kondisional berikut:

- **if statement:** Menjalankan blok kode jika kondisi benar.
- **elif statement:** Memeriksa kondisi tambahan jika kondisi sebelumnya salah.
- **else statement:** Menjalankan blok kode jika semua kondisi sebelumnya salah.

Sintaks:

```
Python Code
if kondisi:
    # Blok kode yang dijalankan jika kondisi benar
elif kondisi_lain:
    # Blok kode yang dijalankan jika kondisi sebelumnya salah, tetapi
    # yang ini benar
else:
    # Blok kode yang dijalankan jika semua kondisi salah
```

Penjelasan

Kode ini menunjukkan cara program membuat keputusan berdasarkan kondisi yang diberikan. Pertama kode memeriksa kondisi setelah kata kunci `if`. Jika kondisi tersebut benar, maka blok kode di bawah `if` akan dijalankan. Jika kondisi pertama salah, program akan memeriksa kondisi di bawah `elif`. Jika kondisi ini benar, kode setelah `elif` akan dijalankan. Jika kedua kondisi tersebut salah, maka kode dalam blok `else` akan dijalankan sebagai pilihan terakhir. Ini memungkinkan program untuk mengambil keputusan berbeda berdasarkan kondisi yang diuji.

3.3 Contoh : Pernyataan Kondisional Dasar

Mari kita mulai dengan contoh sederhana `if else` untuk menggambarkan pernyataan kondisional, perhatikan kode berikut :

```
#pernyataan kondisional sederhana (2 kondisi)

cuaca = input("Apakah sedang hujan? (ya/tidak): ").lower()

if cuaca == "ya":
    print("Pakai payung!")

else:
    print("Tidak perlu pakai payung!")
```

Penjelasan :

Kode di atas merupakan contoh sederhana dari penggunaan pernyataan kondisional yang hanya memiliki 2 kondisi, untuk menentukan apakah perlu pakai payung atau tidak, berdasarkan pertanyaan “Apakah sedang hujan?”. Pertama, program meminta pengguna

memasukkan kondisi cuaca dengan menggunakan fungsi `input()`. Input tersebut secara default berupa teks (`string`), sehingga kita tidak perlu menentukan tipe data.

Jika kondisi pertama terpenuhi (jawaban ya), program akan mencetak pesan "Pakai payung". Jika kondisinya tidak terpenuhi, maka program akan mencetak "Tidak perlu pakai payung!" .

```
#pernyataan kondisional dengan 3 kondisi
angka = int(input("Masukkan sebuah angka: ")) #inputan
if angka > 0:
    print("Angka tersebut positif.")
elif angka < 0:
    print("Angka tersebut negatif.")
else:
    print("Angka tersebut nol.")
```

Penjelasan :

Kode tersebut merupakan contoh sederhana dari penggunaan pernyataan kondisional untuk memeriksa apakah sebuah angka yang dimasukkan oleh pengguna adalah positif, negatif, atau nol. Pertama, program meminta pengguna memasukkan sebuah angka dengan menggunakan fungsi `input()`. Input tersebut secara default berupa teks (`string`), sehingga harus dikonversi menjadi bilangan bulat (`integer`) menggunakan fungsi `int()`.

Setelah angka diperoleh dan dikonversi, program mulai memeriksa nilai angka tersebut menggunakan pernyataan kondisional `if`, `elif`, dan `else`. Pada bagian `if`, program mengecek apakah angka lebih besar dari nol (`angka > 0`). Jika kondisi ini benar, maka program akan mencetak "Angka tersebut positif.".

Jika kondisi pertama tidak terpenuhi (angka tidak lebih besar dari nol), program akan memeriksa kondisi berikutnya menggunakan `elif`. Pada bagian ini, program mengecek apakah angka lebih kecil dari nol (`angka < 0`). Jika kondisi ini benar, program akan mencetak "Angka tersebut negatif.".

Jika kedua kondisi sebelumnya tidak terpenuhi (angka tidak lebih besar dari nol dan tidak lebih kecil dari nol), maka program akan menjalankan blok kode di dalam `else`. Ini berarti angka adalah nol, dan program akan mencetak "Angka tersebut nol.". Dengan cara ini, program dapat menentukan dengan tepat apakah angka yang dimasukkan positif, negatif,

atau nol, dan memberikan respons yang sesuai berdasarkan hasil pemeriksaan kondisi. Struktur kode ini sangat berguna untuk memproses keputusan logis dalam berbagai situasi.

3.4 Pernyataan Kondisional Bersarang

Pernyataan kondisional bersarang merupakan struktur logika di mana sebuah pernyataan kondisional diletakkan di dalam pernyataan kondisional lainnya. Dengan kata lain, ini adalah situasi di mana sebuah blok `if`, `elif`, atau `else` berisi pernyataan kondisional lain. Tujuan dari pernyataan kondisional bersarang adalah untuk menangani situasi yang memerlukan lebih dari satu tingkat pengambilan keputusan, sehingga memungkinkan program untuk menangani skenario yang lebih kompleks.

Untuk memahami ini, kita dapat mengambil ilustrasi dengan pengambilan keputusan dalam kehidupan sehari-hari. Misalnya, Anda mungkin bertanya kepada seseorang, "Apakah kamu suka minuman hangat?" Jika jawabannya "ya", Anda mungkin melanjutkan dengan bertanya, "Apakah kamu lebih suka teh atau kopi?" Keputusan kedua ini hanya relevan jika jawaban untuk pertanyaan pertama adalah "ya." Konsep yang sama diterapkan dalam pernyataan kondisional bersarang di Python. Anda dapat meletakkan satu pernyataan `if` di dalam pernyataan lain untuk membuat logika pengambilan keputusan yang lebih kompleks. Adapun contoh kondisional bersarang adalah sebagai berikut :

```
# Contoh pernyataan kondisional bersarang
umur = int(input("Masukkan umur Anda: "))
if umur >= 18:
    if umur >= 60:
        print("Anda adalah warga senior.")
    else:
        print("Anda adalah orang dewasa.")
else:
    print("Anda adalah anak di bawah umur.")
```

Penjelasan :

Kode tersebut menggunakan pernyataan kondisional bersarang untuk menentukan kategori usia berdasarkan umur yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan umur mereka menggunakan fungsi `input()`. Nilai yang dimasukkan, yang berupa teks (`string`), kemudian dikonversi menjadi bilangan bulat (`integer`) menggunakan fungsi `int()`. Nilai ini disimpan dalam variabel `umur`.

Program kemudian memeriksa nilai umur dengan menggunakan pernyataan `if`. Kondisi pertama adalah apakah umur (`umur`) lebih besar atau sama dengan 18. Jika kondisi

ini benar, program masuk ke dalam blok `if` dan memeriksa kondisi kedua: apakah umur lebih besar atau sama dengan 60. Jika kondisi kedua ini juga benar, program mencetak "Anda adalah warga senior." untuk menunjukkan bahwa orang tersebut termasuk dalam kategori usia lanjut.

Jika kondisi kedua tidak terpenuhi (umur kurang dari 60 tetapi lebih besar atau sama dengan 18), maka program menjalankan pernyataan di blok `else` yang terkait dengan kondisi kedua dan mencetak "Anda adalah orang dewasa." Namun, jika kondisi awal (`umur >= 18`) salah, berarti umur kurang dari 18, dan program langsung menjalankan blok `else` dari kondisi pertama. Dalam kasus ini, program mencetak "Anda adalah anak di bawah umur."

Secara keseluruhan, kode ini mengelompokkan umur pengguna ke dalam tiga kategori: anak di bawah umur (kurang dari 18 tahun), orang dewasa (18 hingga 59 tahun), dan warga senior (60 tahun ke atas). Logika bersarang ini memastikan bahwa setiap kondisi diperiksa secara bertahap, dan output yang sesuai diberikan berdasarkan nilai umur yang dimasukkan.

3.5 Operator Logika dalam pernyataan Kondisional

Operator logika dalam pernyataan kondisional merupakan mekanisme yang digunakan untuk menghubungkan dua atau lebih kondisi sehingga memungkinkan program mengambil keputusan berdasarkan kombinasi kondisi tersebut. Operator ini sangat penting dalam pemrograman karena memberikan fleksibilitas yang besar dalam menentukan alur logika program. Terdapat tiga operator logika utama di Python, yaitu `and`, `or`, dan `not`. Setiap operator memiliki fungsi khusus dalam mengevaluasi ekspresi logis dan bekerja dengan nilai boolean, yaitu `True` dan `False`.

- Operator `and` digunakan ketika kita ingin memastikan bahwa beberapa kondisi harus terpenuhi sekaligus agar keseluruhan ekspresi bernilai benar. Jika salah satu kondisi bernilai salah (`False`), maka seluruh ekspresi juga akan dianggap salah. Misalnya, jika kita ingin memeriksa apakah seseorang berusia di antara 18 dan 30 tahun, kita dapat menggunakan operator `and` untuk memastikan kedua kondisi tersebut terpenuhi.
- Operator `or`, di sisi lain, digunakan ketika hanya salah satu dari beberapa kondisi yang perlu terpenuhi agar ekspresi dianggap benar. Dengan kata lain, ekspresi akan bernilai salah hanya jika semua kondisi bernilai salah

- Operator terakhir adalah `not`, yang digunakan untuk membalik nilai sebuah kondisi. Jika kondisi bernilai benar, maka `not` akan mengubahnya menjadi salah, dan sebaliknya. Operator ini sering digunakan untuk situasi di mana kita ingin memastikan bahwa kondisi tertentu tidak terpenuhi.

Untuk contoh pernyataan kondisional dengan menggunakan operator logika dicontoh di kode berikut :

```
# Contoh menggunakan operator logika
suhu = int(input("Masukkan suhu dalam Celcius: "))
if suhu > 20 and suhu < 30:
    print("Cuacanya hangat.")
elif suhu <= 20:
    print("Cuacanya dingin.")
else:
    print("Cuacanya panas.")
```

3.6 Contoh Latihan Pernyataan Kondisional dalam kehidupan sehari-hari

Latihan 1: Pernyataan Kondisional (Sistem ATM)

Tugas: Anda diminta untuk membuat sebuah program autentikasi sederhana. Program ini akan memverifikasi apakah PIN yang dimasukkan pengguna sesuai dengan PIN yang telah ditentukan sebelumnya. Berikut adalah skenario yang harus Anda implementasikan:

1. Tentukan sebuah PIN yang telah ditetapkan sebelumnya, misalnya 1234.
2. Mintalah pengguna untuk memasukkan PIN melalui input.
3. Periksa apakah PIN yang dimasukkan pengguna sesuai dengan PIN yang telah ditentukan.
4. Jika PIN sesuai, tampilkan pesan "**Autentikasi berhasil!**". Jika PIN tidak sesuai, tampilkan pesan "**PIN salah. Akses ditolak.**"

Solusi

```
# Contoh: Autentikasi PIN
pin_benar = 1234 # PIN yang sudah ditentukan
pin_dimasukkan = int(input("Masukkan PIN Anda: ")) # Input dari pengguna

# Memeriksa apakah PIN yang dimasukkan benar
if pin_dimasukkan == pin_benar:
    print("Autentikasi berhasil!")
```

```

else:
    print("PIN salah. Akses ditolak.")

```

Penjelasan :

Source code ini adalah contoh sederhana dari sistem autentikasi menggunakan PIN. Program dimulai dengan menetapkan sebuah nilai PIN yang benar, yaitu 1234, yang disimpan dalam variabel `pin_benar`. Nilai ini dianggap sebagai PIN tetap yang harus dimasukkan oleh pengguna untuk mendapatkan akses. Program kemudian meminta pengguna untuk memasukkan PIN mereka melalui fungsi `input()`. Input yang dimasukkan oleh pengguna berupa teks (string) secara default, sehingga perlu diubah menjadi tipe data bilangan bulat (integer) menggunakan fungsi `int()`. Nilai yang dimasukkan oleh pengguna disimpan dalam variabel `pin_dimasukkan`.

Setelah mendapatkan input dari pengguna, program memeriksa apakah nilai yang dimasukkan (`pin_dimasukkan`) sama persis dengan nilai PIN yang benar (`pin_benar`). Pemeriksaan ini dilakukan menggunakan pernyataan kondisional `if`. Jika nilai yang dimasukkan benar, yaitu sama dengan 1234, program mencetak pesan "Autentikasi berhasil!" untuk memberi tahu pengguna bahwa mereka telah berhasil melakukan autentikasi. Namun, jika nilai yang dimasukkan tidak sesuai dengan nilai PIN yang benar, program akan menjalankan blok `else`. Dalam blok ini, program mencetak pesan "PIN salah. Akses ditolak." untuk memberi tahu pengguna bahwa mereka tidak diizinkan mengakses karena PIN yang dimasukkan salah.

Secara keseluruhan, kode ini menunjukkan bagaimana program dapat digunakan untuk membandingkan input pengguna dengan nilai yang telah ditentukan sebelumnya, memberikan respons yang sesuai berdasarkan hasil perbandingan tersebut. Contoh ini relevan dalam sistem keamanan sederhana, seperti autentikasi ATM atau aplikasi berbasis PIN.

Latihan 2: Pernyataan Kondisional (Transaksi Peminjaman Buku)

Tugas: Anda diminta untuk membuat sebuah program sederhana untuk sistem peminjaman buku di perpustakaan. Program ini akan memeriksa apakah jumlah buku yang ingin dipinjam

oleh pengguna tersedia dalam stok atau tidak. Berikut adalah situasi yang harus Anda perhatikan:

1. Ada **stok buku** awal yang disimpan dalam sebuah variabel, misalnya `stok_buku = 5`.
2. Pengguna akan memasukkan jumlah buku yang ingin dipinjam. Jumlah ini disimpan dalam variabel `jumlah_peminjaman`, misalnya `jumlah_peminjaman = 6`.
3. Program harus memeriksa apakah jumlah buku yang diminta lebih kecil atau sama dengan stok yang tersedia.
4. Jika stok mencukupi, program mencetak pesan "**Peminjaman berhasil. Buku tersedia.**", dan stok buku dikurangi sesuai jumlah peminjaman.
5. Jika stok tidak mencukupi, program mencetak pesan "**Peminjaman gagal. Stok tidak mencukupi.**" dan memberitahukan jumlah stok yang masih tersedia.

Solusi

```
# Inisialisasi variabel
stok_buku = 5

# Input jumlah peminjam
jumlah_peminjaman = int(input("Masukkan jumlah buku yang ingin dipinjam: "))

# Cek apakah peminjaman buku dapat dilakukan
if jumlah_peminjaman <= stok_buku:
    print("Peminjaman berhasil. Buku tersedia.")
    stok_buku = stok_buku - jumlah_peminjaman
else:
    print("Peminjaman gagal. Stok tidak mencukupi.")
    print("Stok tersedia hanya:", stok_buku)
```

Penjelasan :

Kode ini digunakan untuk menentukan apakah peminjaman buku di perpustakaan dapat dilakukan berdasarkan jumlah stok yang tersedia. Pada awal program, terdapat variabel `stok_buku` yang bernilai 5, artinya perpustakaan memiliki 5 buku. Kemudian, terdapat variabel `jumlah_peminjaman` dengan nilai yang didapatkan dari masukan pengguna, yang menunjukkan jumlah buku yang ingin dipinjam oleh pengguna.

Program menggunakan pernyataan kondisional `if-else` untuk memeriksa kondisi tertentu. Pertama, program mengecek apakah jumlah buku yang ingin dipinjam

(jumlah_peminjaman) kurang dari atau sama dengan jumlah buku yang tersedia (stok_buku). Jika kondisi ini benar, program akan mencetak pesan "Peminjaman berhasil. Buku tersedia." dan mengurangi jumlah stok buku dengan jumlah yang dipinjam (stok_buku = stok_buku - jumlah_peminjaman). Namun, jika kondisi tersebut tidak terpenuhi (jumlah buku yang ingin dipinjam lebih besar daripada stok yang tersedia), program akan mencetak pesan "Peminjaman gagal. Stok tidak mencukupi." dan menampilkan jumlah stok buku yang tersisa.

Apabila pengguna ingin meminjam 6 buku sementara stok yang tersedia hanya 5, program akan mencetak pesan bahwa peminjaman gagal dan menyatakan bahwa stok yang tersedia hanya 5. Program ini memberikan contoh sederhana tentang bagaimana logika pengambilan keputusan dapat digunakan untuk memeriksa ketersediaan sumber daya sebelum memberikan respons.

Latihan 3: Pernyataan Kondisional (Transaksi Pemesanan Tiket Kereta Api)

Tugas: Agen kereta api menyediakan tiket dengan harga Rp150.000 per orang. Jika jumlah penumpang lebih dari 4 orang, mereka akan mendapatkan diskon sebesar 10%. Program berikut digunakan untuk menghitung total harga tiket yang harus dibayar berdasarkan jumlah penumpang.

Solusi

```
# Inisialisasi variabel
harga_tiket = 150000
jumlah_penumpang = 5
total_harga = harga_tiket * jumlah_penumpang

# Diskon untuk jumlah penumpang lebih dari 4
if jumlah_penumpang > 4:
    diskon = 0.1 # 10%
    total_harga_diskon = total_harga * (1 - diskon)
    print("Anda mendapatkan diskon 10%!")
    print("Total Harga Setelah Diskon: Rp", total_harga_diskon)
else:
    print("Tidak ada diskon.")
    print("Total Harga: Rp", total_harga)
```

Penjelasan :

Program ini digunakan untuk menghitung total harga tiket perjalanan berdasarkan jumlah penumpang yang memesan. Pada awalnya, harga tiket ditentukan sebesar Rp150.000 per orang, dan jumlah penumpang yang ingin membeli tiket adalah 5 orang. Langkah pertama dalam program ini adalah menghitung total harga yang harus dibayar jika tidak ada diskon, yaitu mengalikan harga tiket dengan jumlah penumpang. Dalam hal ini, total harga awal adalah Rp150.000 dikali 5, yang menghasilkan Rp750.000.

Setelah itu, program memeriksa apakah jumlah penumpang lebih dari 4. Jika ya, program memberikan diskon 10%. Diskon ini dihitung dengan cara mengalikan total harga dengan angka 0.1, yang berarti 10% dari total harga. Program kemudian mengurangi total harga dengan diskon yang didapatkan, menghasilkan total harga setelah diskon. Dalam contoh ini, total harga setelah diskon adalah Rp750.000 dikurangi 10%, yaitu Rp675.000.

Jika jumlah penumpang tidak lebih dari 4, program akan mencetak pesan bahwa tidak ada diskon, dan menampilkan total harga yang tetap sebesar Rp750.000. Dengan demikian, program ini memeriksa jumlah penumpang, memberikan diskon jika memenuhi syarat, dan menghitung total harga yang harus dibayar berdasarkan kondisi tersebut.

Latihan 4: Pernyataan Kondisional (Perencanaan Anggaran Pribadi)

Tugas: Seseorang ingin memeriksa kondisi keuangannya berdasarkan nilai sisa anggaran. Buat sebuah program dapat memeriksa jumlah uang.

- Jika sisa_anggaran lebih besar dari nol, program akan mencetak bahwa “*anggaran aman dan menampilkan jumlah sisa anggaran*”.
- Jika sisa_anggaran sama dengan nol, program akan mencetak bahwa “*anggaran tepat tanpa sisa*”.
- Namun, jika sisa_anggaran kurang dari nol, program akan mencetak bahwa “*anggaran melebihi batas dan menampilkan jumlah kekurangan dengan nilai positif*”.

Solusi

```
# Inisialisasi variabel
sisa_anggaran = 1000000

# Cek status anggaran
if sisa_anggaran > 0:
    print("Anggaran aman. Sisa anggaran sebesar Rp", sisa_anggaran)
elif sisa_anggaran == 0:
    print("Anggaran tepat! Tidak ada sisa.")
```

```

else:
    print("Anggaran melebihi batas! Anda kekurangan Rp", abs(sisa_anggaran))

```

Penjelasan :

Kode tersebut bertujuan untuk memeriksa kondisi keuangan berdasarkan variabel `sisa_anggaran` dan memberikan informasi yang relevan kepada pengguna. Program ini mengevaluasi apakah anggaran masih tersisa, tepat habis, atau justru mengalami defisit. Penjelasan ini dirancang agar dapat dipahami bahkan oleh orang yang tidak memiliki latar belakang pemrograman.

Pada awalnya, variabel `sisa_anggaran` diberi nilai 1.000.000, yang mewakili jumlah uang yang masih tersedia setelah menghitung pemasukan dan pengeluaran. Program kemudian memeriksa nilai variabel ini menggunakan pernyataan bersyarat untuk menentukan status anggaran.

Jika nilai `sisa_anggaran` lebih besar dari nol, program menyimpulkan bahwa masih ada sisa uang, atau dengan kata lain, anggaran berada dalam kondisi aman. Dalam situasi ini, program akan mencetak pesan, seperti: "Anggaran aman. Sisa anggaran sebesar Rp 1000000." Hal ini memberikan informasi kepada pengguna bahwa masih ada uang yang bisa digunakan.

Namun, jika nilai `sisa_anggaran` sama dengan nol, program menyimpulkan bahwa seluruh anggaran telah digunakan dengan tepat, tanpa ada kelebihan atau kekurangan. Dalam kasus ini, program akan mencetak pesan: "Anggaran tepat! Tidak ada sisa." Hal ini menunjukkan bahwa pengguna telah mengelola keuangan dengan sangat presisi.

Terakhir, jika nilai `sisa_anggaran` kurang dari nol, program mengidentifikasi bahwa anggaran telah melampaui batas yang ditentukan, artinya terjadi defisit. Dalam kasus ini, program mencetak pesan: "Anggaran melebihi batas! Anda kekurangan Rp x," di mana x adalah jumlah uang yang kurang. Fungsi `abs()` digunakan di sini untuk mengubah nilai negatif menjadi positif agar mudah dipahami oleh pengguna. Misalnya, jika `sisa_anggaran` bernilai -50000, maka program akan mencetak: "Anggaran melebihi batas! Anda kekurangan Rp 50000." Secara keseluruhan, kode ini adalah contoh yang sederhana namun sangat efektif untuk memberikan laporan keuangan kepada pengguna. Dengan

memeriksa nilai dari variabel anggaran, program ini dapat memberikan masukan yang jelas dan langsung tentang kondisi keuangan pengguna.

3.7 Berikut Tugas Fokus pada (if, if-else, if-elif-else)

Ketentuan Tugas : setiap mahasiswa diminta mengerjakan 2 soal: 1 soal sesuai dengan NIM-nya, dan 1 soal bebas memilih.

Tugas 1 : Mengidentifikasi Predikat dari IPK Mahasiswa

Tulis program untuk mengidentifikasi predikat dari IPK Mahasiswa.

Rentang dan predikat :

IPK 2,00-2,75 Memuaskan.

IPK 2,76-3,50 Sangat memuaskan.

IPK 3,51-4,00 Dengan pujian (Cumlaude)

Contoh Input/Output:

Masukkan IPK mahasiswa: 3.75

Predikat: Dengan Pujian (Cumlaude)

Tugas 2: Menentukan Bilangan Ganjil atau Genap

Deskripsi:

Buat program untuk memeriksa apakah bilangan yang dimasukkan adalah ganjil atau genap.

Contoh Input/Output:

Masukkan bilangan: 8

Hasil: Genap

Tugas 3: Menentukan Kelulusan Siswa

Deskripsi:

Buat program yang menentukan apakah seorang siswa lulus berdasarkan nilai yang dimasukkan. Syarat kelulusan: nilai ≥ 60 .

Contoh Input/Output:

Masukkan nilai: 75

Hasil: Lulus

Tugas 4: Menentukan Diskon Berdasarkan Jumlah Belanja

Deskripsi:

Buat program untuk menentukan diskon berdasarkan jumlah belanja:

- Jika belanja $\geq 1.000.000$, diskon 10%.
- Jika belanja $< 1.000.000$, tidak ada diskon.

Contoh Input/Output:

Masukkan jumlah belanja: 1200000

Total setelah diskon: 1080000.0

Tugas 5: Menentukan Kategori Usia

Deskripsi:

Tulis program yang menentukan kategori usia berdasarkan input umur:

- Anak-anak (0-12 tahun)
- Remaja (13-17 tahun)
- Dewasa (18 tahun ke atas)

Contoh Input/Output:

Masukkan usia: 16

Kategori: Remaja

Tugas 6: Menentukan Nilai Huruf

Deskripsi:

Buat program untuk mengonversi nilai angka menjadi nilai huruf:

- A: nilai ≥ 85
- B: $70 \leq \text{nilai} < 85$
- C: $50 \leq \text{nilai} < 70$
- D: nilai < 50

Contoh Input/Output:

Masukkan nilai: 78

Nilai huruf: B

Tugas 7: Menentukan Tahun Kabisat

Deskripsi:

Tulis program untuk memeriksa apakah sebuah tahun adalah tahun kabisat atau bukan.

Contoh Input/Output:

Masukkan tahun: 2024

Hasil: Tahun kabisat

Tugas 8: Menentukan Angka Terbesar

Deskripsi:

Buat program untuk menemukan angka terbesar dari tiga angka yang dimasukkan pengguna.

Contoh Input/Output:

Masukkan angka pertama: 5

Masukkan angka kedua: 12

Masukkan angka ketiga: 8

Angka terbesar: 12

Tugas 9: Menentukan Jenis Segitiga

Deskripsi:

Tulis program untuk menentukan jenis segitiga berdasarkan panjang tiga sisinya:

- Sama Sisi: semua sisi sama panjang
- Sama Kaki: dua sisi sama panjang
- Sembarang: semua sisi berbeda panjang

Contoh Input/Output:

Masukkan sisi pertama: 5

Masukkan sisi kedua: 5

Masukkan sisi ketiga: 8

Jenis segitiga: Sama Kaki

Tugas 10: Sistem Penilaian Sederhana

Deskripsi:

Buat program untuk menentukan status siswa berdasarkan nilai ujian dan nilai tugas:

- Lulus: nilai ujian ≥ 60 dan nilai tugas ≥ 70

- Tidak Lulus: selain itu.

Contoh Input/Output:

Masukkan nilai ujian: 65

Masukkan nilai tugas: 75

Hasil: Lulus

Kesimpulan

Memahami pernyataan kondisional adalah kunci untuk menulis program Python yang efektif.

Dengan menguasai konsep ini, Anda dapat mengontrol alur program dan menangani berbagai situasi secara dinamis. Latihan-latihan di atas diharapkan dapat membantu Anda memperkuat pemahaman tentang penggunaan pernyataan kondisional dalam berbagai konteks. Selalu ikuti **praktik penulisan kode terbaik** agar kode Anda tetap bersih, mudah dibaca, dan efisien.