Modul 3 Praktikum Algoritma Pemrograman

Topik Praktikum : Pengkondisian dengan If dalam Pemrograman Python

Tujuan Praktikum:

1. Mahasiswa dapat memahami dan menggunakan operator perbandingan
2. Mahasiswa dapat memahami dan menggunakan operator logika
3. Mahasiswa dapat memahami dan menggunakan Algoritma Control Flow dengan If-Else

1 Pendahuluan

Dalam modul ini kita akan membahas pernyataan if dalam Python. Pernyataan ini digunakan untuk mengontrol alur eksekusi dalam sebuah program berdasarkan suatu kondisi. Kondisi ini mewakili titik pilihan yang akan dievaluasi menjadi True atau False. Untuk melakukan evaluasi ini, biasanya digunakan operator perbandingan (misalnya untuk memeriksa apakah suhu lebih tinggi dari ambang batas tertentu). Dalam banyak kasus, perbandingan ini perlu mempertimbangkan beberapa nilai, dan dalam situasi tertentu operator logika dapat digunakan untuk menggabungkan dua atau lebih ekspresi perbandingan.

2 Operator Perbandingan

Sebelum mempelajari pernyataan **if**, kita perlu membahas operator perbandingan. Operator operator ini mengembalikan nilai Boolean. Operator ini sangat penting dalam elemen kondisional dari pernyataan alur kontrol seperti **if**.
Operator perbandingan adalah operator yang melakukan suatu bentuk pengujian dan mengemba likan **True** atau **False**.
Operator-operator ini sering kita gunakan dalam kehidupan sehari-hari. Sebagai contoh, apakah saya memiliki cukup uang untuk membeli makan siang, atau apakah sepatu ini sesuai dengan ukuran saya, dan sebagainya.

Dalam Python, terdapat berbagai operator perbandingan yang biasanya direpresentasikan dengan satu atau dua karakter. Berikut ini adalah operator-operator tersebut:

Operator	Deskripsi	Contoh
==	Menguji apakah dua nilai sama	3 == 3
!=	Menguji apakah dua nilai tidak sama	2 != 3
<	Menguji apakah nilai di sebelah kiri lebih kecil dari nilai di kanan	2 < 3
>	Menguji apakah nilai di sebelah kiri lebih besar dari nilai di kanan	3 > 2
<=	Menguji apakah nilai di sebelah kiri lebih kecil atau sama dengan nilai di kanan	3 <= 4
>=	Menguji apakah nilai di sebelah kiri lebih besar atau sama dengan nilai di kanan	5 >= 4

3 Operator Logika

Selain operator perbandingan, Python juga memiliki operator logika. Operator logika dapat digu nakan untuk menggabungkan ekspresi Boolean. Biasanya, operator ini digunakan bersama dengan operator perbandingan untuk menciptakan kondisi yang lebih kompleks. Kita menggunakan op erator ini dalam kehidupan sehari-hari, misalnya kita mungkin mempertimbangkan apakah kita mampu membeli es krim **dan** apakah kita akan makan malam segera, dll.

Terdapat tiga operator logika dalam Python, yang tercantum di bawah ini:

Operator	Deskripsi	Contoh
and	Mengembalikan nilai **True** jika kedua nilai di kiri dan kanan bernilai **True**	(3 < 4) and (5 > 4)
or	Mengembalikan nilai **True** jika salah satu dari nilai kiri atau kanan bernilai **True**	(3 < 4) or (3 > 5)
not	Mengembalikan nilai **True** jika nilai yang diuji adalah **False**	not (3 < 2)

4 Pernyataan If

Pernyataan **if** digunakan sebagai bentuk pemrograman kondisional; sesuatu yang mungkin Anda lakukan setiap hari dalam kehidupan nyata. Artinya, Anda perlu memutuskan apakah akan minum teh atau kopi, atau memutuskan apakah akan makan roti panggang atau muffin untuk sarapan, dll. Dalam setiap kasus pernyataan, Anda membuat pilihan, biasanya berdasarkan informasi seperti “Saya sudah minum kopi kemarin, jadi saya akan minum teh hari ini.”

Dalam Python, pilihan semacam ini direpresentasikan secara programatis melalui pernyataan **if**. Dalam struktur ini, jika suatu kondisi benar, maka suatu tindakan akan dilakukan, dan secara opsional jika kondisi tersebut tidak benar, maka tindakan lain dapat dilakukan sebagai gantinya.

4.1 Bekerja dengan Pernyataan If

Dalam bentuk paling dasar, pernyataan **if** adalah:

```
if <kondisi-yang-dievaluasi-menjadi-boolean>:
    statement
```

Perhatikan bahwa kondisi harus dievaluasi menjadi **True** atau **False**. Jika kondisi **True**, maka pernyataan yang diindentasi akan dieksekusi.

Perhatikan indentasi, hal ini sangat penting dalam Python; sebenarnya, tata letak kode sangatlah penting dalam Python. Indentasi digunakan untuk menentukan bagaimana satu bagian kode harus diasosiasikan dengan bagian kode lainnya.

Mari kita lihat contoh sederhana:

```
num = int(input('Masukkan sebuah angka: '))
if num < 0:
    print(num, 'adalah angka negatif')
```

Dalam contoh ini, pengguna memasukkan sebuah angka; jika angka tersebut kurang dari nol, pesan yang menyatakan hal ini akan dicetak kepada pengguna. Jika angkanya positif, maka tidak ada output.

Contohnya:

```
Masukkan sebuah angka: -1
-1 adalah angka negatif
```

Jika kita ingin mengeksekusi beberapa pernyataan ketika kondisi kita **True**, kita bisa mengindentasi beberapa baris; pada kenyataannya, semua baris yang diindentasi ke tingkat yang sama setelah pernyataan **if** secara otomatis akan menjadi bagian dari pernyataan **if**. Contohnya:

```
num = int(input('Masukkan angka lain: '))
if num > 0:
    print(num, 'adalah angka positif')
    print(num, 'kuadratnya adalah', num * num)
    print('Bye')
```

Jika kita menjalankan program ini dan memasukkan 2, maka kita akan melihat:

```
Masukkan angka lain: 2
2 adalah angka positif
2 kuadratnya adalah 4
Bye
```

Namun, jika kita memasukkan nilai −1, maka kita mendapatkan:

```
Masukkan angka lain: -1
Bye
```

Perhatikan bahwa tidak ada baris yang diindentasi yang dieksekusi. Ini karena dua baris yang diin dentasi terkait dengan pernyataan **if** dan hanya akan dieksekusi jika kondisi Boolean mengevaluasi menjadi **True**. Namun, pernyataan print('Bye') bukan bagian dari pernyataan **if**; ini hanyalah pernyataan selanjutnya yang dieksekusi setelah pernyataan **if** (dan pernyataan print() terkait) selesai dieksekusi. “‘

4.2 Else dalam Pernyataan If

Kita juga dapat mendefinisikan bagian **else** dalam pernyataan **if**; elemen ini dapat dijalankan jika bagian kondisional dari pernyataan **if** mengembalikan **False**. Contohnya:

```
num = int(input('Masukkan angka lain lagi: '))
if num < 0:
    print('Angkanya negatif')
else:
    print('Angkanya tidak negatif')
```

Sekarang, ketika kode ini dijalankan, jika angka yang dimasukkan kurang dari nol, maka pernyataan **print()** pertama akan dijalankan, sedangkan jika tidak (else), pernyataan **print()** kedua akan dijalankan. Namun, kita dijamin bahwa setidaknya satu (dan paling banyak satu) dari perny ataan **print()** akan dieksekusi.

Contohnya, pada eksekusi pertama, jika kita memasukkan angka 1:

Masukkan angka lain lagi: 1
Angkanya tidak negatif

Dan pada eksekusi kedua, jika kita memasukkan angka −1:

Masukkan angka lain lagi: -1
Angkanya negatif

4.3 Penggunaan elif

Dalam beberapa kasus, mungkin ada beberapa kondisi yang ingin Anda uji, di mana setiap kondisi diuji jika kondisi sebelumnya gagal. Skenario else-if ini didukung di Python dengan elemen **elif** dalam pernyataan **if**.

Elemen **elif** dalam pernyataan **if** mengikuti bagian **if** dan muncul sebelum bagian **else** (opsional). Formatnya adalah sebagai berikut:

```
elif <kondisi-yang-dievaluasi-menjadi-boolean>:
    statement
```

Contohnya:

```
savings = float(input("Masukkan jumlah tabungan Anda: "))
if savings == 0:
    print("Maaf, tidak ada tabungan")
elif savings < 500:
    print('Bagus sekali')
elif savings < 1000:
    print('Lumayan ya')
elif savings < 10000:
    print('Selamat Berjuang!')
else:
    print('Terima kasih')
```

Jika kita menjalankan ini:

Masukkan jumlah tabungan Anda: 500
Lumayan ya

Di sini kita dapat melihat bahwa kondisi **if** pertama gagal (karena tabungan tidak sama den gan 0). Namun, **elif** berikutnya juga mengembalikan **False** karena tabungan lebih besar dari 500. Faktanya, pernyataan **elif** kedua yang mengembalikan **True** sehingga pernyataan terkait **print('Lumayan ya')** dijalankan. Setelah pernyataan ini dieksekusi, pernyataan **if** dihentikan (bagian **elif** dan **else** yang tersisa diabaikan).

5 Menyusun Nested Pernyataan If

Adalah mungkin untuk menyusun (nesting) satu pernyataan **if** di dalam pernyataan **if** lainnya. Istilah **nesting** menunjukkan bahwa satu pernyataan **if** ditempatkan di dalam bagian dari perny ataan **if** lainnya dan dapat digunakan untuk memperhalus perilaku kondisional dari program.

Contoh diberikan di bawah ini. Perhatikan bahwa contoh ini memungkinkan beberapa tindakan dilakukan sebelum dan setelah pernyataan **if** bersarang dijalankan. Juga perhatikan bahwa in dentasi sangat penting di sini karena ini adalah cara Python menentukan apakah pernyataan **if** tersebut bersarang atau tidak.

```
snowing = True
temp = -1
if temp < 0:
      print('It is freezing')
      if snowing:
            print('Put on boots')
      print('Time for Hot Chocolate')
print('Bye')
```

Dalam contoh ini, jika suhu kurang dari nol maka kita akan masuk ke dalam blok kode **if**. Jika suhu tidak kurang dari nol, kita akan melewati seluruh pernyataan **if** dan melompat ke pernyataan print('Bye') yang berada setelah kedua pernyataan **if**.

Pada kasus ini, suhu diatur menjadi −1, jadi kita akan masuk ke dalam pernyataan **if**. Kita kemudian akan mencetak string 'It is freezing'. Pada titik ini, pernyataan **if** lainnya bersarang di dalam pernyataan **if** yang pertama. Sebuah pemeriksaan akan dilakukan untuk melihat apakah sedang turun salju. Perhatikan bahwa **snowing** sudah merupakan nilai Boolean, sehingga akan bernilai **True** atau **False**, dan ini menggambarkan bahwa nilai Boolean secara langsung dapat digunakan di sini.

Karena sedang turun salju, kita akan mencetak 'Put on boots'. Namun, pernyataan yang mencetak 'Time for Hot Chocolate' bukan bagian dari **if** bersarang. Pernyataan itu adalah bagian dari **if** luar (di sinilah pentingnya indentasi). Jika Anda ingin mencetaknya hanya jika sedang turun salju, maka harus diindentasi ke tingkat yang sama dengan pernyataan pertama di dalam blok **if** bersarang, seperti contoh berikut:

```
snowing = True
temp = -1
if temp < 0:
      print('It is freezing')
      if snowing:
            print('Put on boots')
            print('Time for Hot Chocolate')
print('Bye')
```

Cara ini mengubah **if** bersarang menjadi memiliki dua pernyataan **print** yang terkait dengannya.

Hal ini mungkin tampak sepele, tetapi sangat penting untuk memahami bagaimana Python meng gunakan tata letak untuk menghubungkan pernyataan-pernyataan individu.

6 Ekspresi If

Ekspresi **if** adalah bentuk singkat dari pernyataan **if** yang mengembalikan nilai. Faktanya, perbe daan antara ekspresi dan pernyataan dalam bahasa pemrograman adalah ekspresi mengembalikan nilai, sedangkan pernyataan tidak.

Cukup umum untuk ingin menetapkan nilai spesifik ke sebuah variabel yang bergantung pada suatu kondisi. Misalnya, jika kita ingin memutuskan apakah seseorang adalah remaja atau tidak, kita bisa memeriksa apakah mereka berusia di atas 12 tahun dan di bawah 20 tahun. Kita bisa menulisnya seperti ini:

```
age = 15
status = None
if (age > 12) and age < 20:
    status = 'remaja'
else:
    status = 'bukan remaja'
print(status)
```

Jika kita menjalankan ini, kita akan mendapatkan string 'remaja' dicetak. Namun, ini cukup panjang dan mungkin tidak jelas bahwa maksud sebenarnya dari kode ini adalah menetapkan nilai yang tepat ke variabel **status**.

Alternatifnya adalah menggunakan ekspresi **if**. Format ekspresi **if**

adalah: <result1> if <kondisi-dipenuhi> else <result2>

Artinya, hasil yang dikembalikan dari ekspresi **if** adalah nilai pertama kecuali jika kondisi gagal, dalam hal ini hasil yang dikembalikan adalah nilai setelah **else**. Ini mungkin tampak membin gungkan pada awalnya, tetapi menjadi lebih mudah ketika Anda melihat contohnya.

Sebagai contoh, dengan menggunakan ekspresi **if**, kita dapat melakukan tes untuk menentukan nilai yang akan diberikan ke **status** dan mengembalikannya sebagai hasil dari ekspresi **if**. Contohnya:

```
status = ('remaja' if age > 12 and age < 20 else 'bukan remaja')
print(status)
```

Sekali lagi, hasil yang dicetak adalah 'remaja', namun sekarang kodenya jauh lebih ringkas, dan jelas bahwa tujuan dari tes ini adalah untuk menentukan hasil yang akan diberikan ke **status**.

7 Catatan tentang True dan False

Python sebenarnya cukup fleksibel mengenai apa yang digunakan untuk merepresentasikan **True** dan **False**. Faktanya, aturan berikut berlaku: - **0**, '' (string kosong), **None** dianggap sebagai **False** - Angka selain 0, string yang tidak kosong, dan objek apa pun dianggap sebagai **True**

Namun, kami menyarankan untuk tetap menggunakan hanya **True** dan **False** karena ini sering kali merupakan pendekatan yang lebih bersih dan lebih aman.

8 Petunjuk

Satu hal yang perlu sangat diperhatikan dalam Python adalah tata letak (layout). Berbeda dengan bahasa seperti Java dan C#, tata letak program Anda adalah bagian dari program itu sendiri. Tata letak menentukan bagaimana pernyataan-pernyataan saling terkait dan bagaimana elemen alur kontrol, seperti pernyataan **if**, memengaruhi pernyataan mana yang dieksekusi.

Selain itu, berhati-hatilah dengan pernyataan **if** dan penggunaan karakter '**:**'. Karakter ini sangat penting untuk memisahkan bagian kondisional dari pernyataan **if** dengan pernyataan-pernyataan yang akan dieksekusi tergantung pada apakah kondisinya **True** atau **False**.

9 Dokumentasi Lainnya

1. https://docs.python.org/3/library/stdtypes.html#boolean-operations-and-or-not Boolean Operations.
2. https://docs.python.org/3/library/stdtypes.html#comparisons Comparison operators.
3. https://docs.python.org/3/tutorial/controlflow.html the online Python flow of control tuto rial.

10 Latihan

10.1 Periksa Input Positif atau Negatif

Tujuan dari latihan ini adalah menulis program kecil untuk menguji apakah sebuah bilangan bulat positif atau negatif. Program Anda harus:

Meminta pengguna untuk memasukkan sebuah angka (gunakan fungsi input()). Anda dapat men gasumsikan bahwa input akan berupa semacam angka. Ubah string menjadi bilangan bulat meng gunakan fungsi int(). Sekarang periksa apakah bilangan bulat tersebut positif atau negatif. Anda juga bisa menambahkan tes untuk melihat apakah angkanya nol.

[2]:
```
# Langkah 1: Minta pengguna untuk memasukkan angka
angka = input("Silakan masukkan sebuah angka: ")

# Langkah 2: Ubah string menjadi bilangan bulat
angka = int(angka)

# Langkah 3 dan 4: Periksa apakah angka positif, negatif, atau nol
if angka > 0:
    print(f"{angka} adalah angka positif.")
elif angka < 0:
    print(f"{angka} adalah angka negatif.")
else:
    print("Angka tersebut adalah nol.")
```

1 adalah angka positif.

10.2 Uji Apakah Sebuah Angka Ganjil atau Genap

Latihan ini mengharuskan Anda untuk menulis program yang menerima input dari pengguna dan menentukan apakah angka tersebut ganjil atau genap. Sekali lagi, Anda bisa mengasumsikan bahwa pengguna akan memasukkan bilangan bulat yang valid.
Cetak pesan kepada pengguna untuk memberi tahu hasilnya.
Untuk menguji apakah angka tersebut genap, Anda bisa menggunakan:

(num % 2) == 0

yang akan mengembalikan **True** jika angka tersebut genap (catatan: tanda kurung bersifat op sional, tetapi membuatnya lebih mudah dibaca).

[3]:
```
# Langkah 1: Minta pengguna untuk memasukkan angka
angka = int(input("Silakan masukkan sebuah angka: "))

# Langkah 2: Tentukan apakah angka tersebut ganjil atau genap
if (angka % 2) == 0:
    print(f"{angka} adalah angka genap.")
else:
    print(f"{angka} adalah angka ganjil.")
```

1 adalah angka ganjil.

10.3 Konverter Kilometer ke Mil

Dalam latihan ini, program harus: 1. Modifikasi program Anda sehingga memverifikasi bahwa pengguna telah memasukkan jarak positif (yaitu mereka tidak dapat memasukkan angka negatif). 2. Sekarang modifikasi program Anda untuk memverifikasi bahwa input adalah sebuah angka; jika input bukan angka, tidak lakukan apa pun; jika input adalah angka, konversikan jarak ke mil.

Berikut algoritma nya: 1. Program meminta pengguna untuk memasukkan jarak dalam kilometer menggunakan **input()**. 2. Program memeriksa apakah input hanya berisi angka menggunakan **isnumeric()**. Jika input valid (hanya berisi angka), program melanjutkan ke langkah berikutnya. 3. String input diubah menjadi bilangan bulat menggunakan **int()**. 4. Program memeriksa apakah jarak yang dimasukkan adalah bilangan positif. Jika ya, program mengonversi kilometer ke mil menggunakan faktor konversi 1 km = 0.621371 mil. 5. Jika input bukan angka atau jarak negatif, program akan menampilkan pesan kesalahan yang sesuai.

Program ini memastikan input valid dan hanya mengonversi jarak positif dari kilometer ke mil. Untuk memeriksa apakah sebuah string hanya berisi digit, gunakan metode **isnumeric()**, misalnya '42'.isnumeric(), yang mengembalikan **True** jika string hanya berisi angka. Catatan: metode ini hanya berfungsi untuk bilangan bulat positif, tetapi ini sudah cukup untuk contoh ini.

```
[4]: # Langkah 1: Minta pengguna untuk memasukkan jarak dalam kilometer
            input_km = input("Masukkan jarak dalam kilometer: ")

  # Langkah 2: Verifikasi apakah input adalah angka menggunakan isnumeric() if
  input_km.isnumeric():
        # Langkah 3: Ubah string menjadi bilangan bulat
        km = int(input_km)



        # Langkah 4: Pastikan jarak positif
        if km > 0:
              # Konversi kilometer ke mil (1 km = 0.621371 mil) miles =
              km * 0.621371
              print(f"{km} kilometer setara dengan {miles:.2f} mil.") else:
              print("Silakan masukkan jarak positif.")
  else:
        print("Input tidak valid. Silakan masukkan angka.") 1

 kilometer setara dengan 0.62 mil.
```