Kondisional dan Loop

Penyusun Modul: Ratna Aditya Apsari
Editor: Dinar Hidayah
Fikri Nur Rachman
Rina Fitriyani
Silfa Rahma Aulia

If Statement

Potongan *code* di bawah mendemonstrasikan pernyataan `if` sederhana untuk membandingkan dua bilangan *a* dan *b*

```
a = 7
b = 9

if a < b:
    print(“a lebih kecil dari b”)
if a > b:
    print(“a lebih besar dari b”)
print(“Selesai”)
```

- a lebih kecil dari b, program **Python** akan menampilkan tulisan "a lebih kecil dari b", yang berarti bahwa a dan b memenuhi kondisi "a < b" pada pernyataan *if* pertama.
- Variabel a dan b tidak memenuhi kondisi pada pernyataan *if* kedua, maka *code* yang berada dalam pernyataan *if* tersebut tidak akan dieksekusi.
- Sedangkan perintah **print ("Selesai")** berada di luar pernyataan if.

Hal yang perlu diingat dalam penulisan pernyataan *if*:

- Baris *code* yang perlu dieksekusi dalam suatu kondisi `if` harus diberikan indentasi
- Tanda titik dua (:) harus mengakhiri baris pertama dari pernyataan `if`

Latihan

Buatlah code untuk mengecek apakah hasil perkalian dari dua bilangan a dan b yang didapatkan dari input pengguna dan berada dalam jangkauan 0-50 memenuhi kondisi-kondisi tersebut:

a. Hasil perkalian kurang dari 50
b. Hasil perkalian lebih dari 50
c. Hasil perkalian kurang dari atau sama dengan 50
d. Hasil perkalian lebih dari atau sama dengan 50
e. Hasil perkalian sama dengan 25
f. Hasil perkalian bukan 25

Jika kondisi tersebut terpenuhi, tampilkan tulisan “Hasil perkalian [nama kondisi]”

Solusi

```
a = input ("Input nilai a: ")
b = input ("Input nilai b: ")
if a*b < 50:
        print ("Hasil perkalian a dan b kurang dari 50")
if a*b > 50:
        print ("Hasil perkalian a dan b lebih dari 50")
if a*b <= 50:
        print ("Hasil perkalian a dan b kurang dari atau sama dengan 50")
if a*b >= 50:
        print ("Hasil perkalian a dan b sama dengan 25")
if a*b != 25:
        print ("Hasil perkalian a dan b sama dengan 25")
print("Selesai")
```

Mengecek dua kondisi

Pernyataan `if` dapat digunakan untuk mengecek dua kondisi yang berbeda menggunakan `and` dan `or`.

Menggunakan `and` untuk kedua kondisi tersebut akan membuat Python mengecek kondisional a < b dan a < c sebelum mengeksekusi perintah di dalam pernyataan `if`. Sehingga pernyataan if tersebut hanya dieksekusi ketika a < b dan c.

```
if a < b and a < c:
print ("a kurang dari b dan c")
```

Menggunakan `or` untuk kedua kondisi tersebut akan membuat Python mengecek kondisional a < b atau a < c. Sehingga pernyataan `if` tersebut akan dieksekusi ketika nilai a lebih kecil dari b, atau a lebih kecil dari c, atau bahkan a lebih kecil dari b dan c.

```
if a < b or a < c:
print ("a kurang dari b atau c")
```

Variabel Boolean

Variabel Boolean dapat digunakan sebagai pernyataan dalam pernyataan `if`. Pernyataan `if` akan mengevaluasi apakah suatu ekspresi menghasilkan `true` atau `false`. Jika variabel yang digunakan sudah menghasilkan `true` atau `false`, maka komparasi tidak lagi diperlukan.

```
a = true
b = false
if a and b:
     print("a dan b adalah true")
```

Karena a adalah *true* dan b adalah *false*, maka a *and* b akan menghasilkan *false*. Dengan itu, perintah di dalam pernyataan *if* tidak akan dieksekusi karena kondisi yang ada tidak memenuhi kondisi pada pernyataan *if*.

Variabel Boolean

Perlu diingat bahwa nilai apapun selain 0, misalnya *string* atau bilangan berapapun, pada kondisional pernyataan `if` akan menghasilkan kondisi `true`. Lihatlah potongan *code* di bawah dan cobalah pada IDE Anda.

```
if 0:
  print("Nol")
if 1:
  print("Satu")
if "A":
  print("A")
```

Else dan else if

ketika perlu membuat pertanyaan if untuk kondisi, kita dapat menggunakan else dan else if. cntohnya dapat dilihat pada code di bawah yang dibuat untuk mengecek apakah suhu ruangan yang diinput oleh pengguna termaksuk panas atau dingin.

```
suhu = int (input (Berapakah suhu ruangan
dalam Celsius? "))
if suhu > 40:
        print ("Suhu ruangan panas")
else:
        print ("Suhu ruangan tidak panas")
```

pada *code* tersebut, pertama python akan mengecek kondisi pada pertanyaan *if*. jika *suhu* melebihi 40, maka perintah dalam pernyataan *if* tersebut akan dieksekusi. Namun, ketika *suhu* kurang atau sama dengan 40, maka perintah dalam pernyataan *if* tidak akan dieksekusi dan python akan berpindah ke baris selanjutnya. karena kurang dari dan sama dengan 40 tidak termaksuk pada pernyataan sebelumnya, maka pada baris *else,* ia akan dieksekusi. *else* akan mengeksekusi kondisi apapun selain kondisi yang diberikan pada *if.*

Else dan else if

Pernyataan if dapat digunakan untuk mendefinisikan beberapa kondisional. Hal ini dilakukan menggunakan else if. Di Python, pernyataan else if ditulis sebagai elif.

```
suhu = int (input ("Berapakah suhu ruangan dalam Celsius? "))
if suhu > 40:
     print ("Suhu ruangan panas")
elif suhu < 20:
     print ("Suhu ruangan dingin")
else:
    print ("Suhu ruangan tidak panas dan tidak dingin")
```

Pada code di atas, kita memberikan satu pernyataan kondisional tambahan dengan elif. Ketika suhu ruangan di atas 40, maka tulisan "Suhu ruangan panas" akan ditampilkan. Ketika suhu ruangan di bawah 20, maka tulisan "Suhu ruangan dingin" akan ditampilkan. Ketikakeduakondisional tidak terpenuhi, yaitu pada suhu ruangan 21 hingga 40, bagian else yang akan dieksekusi. Sehingga tulisan "Suhu ruangan tidak panas dan tidak dingin" akan ditampilkan.

Membandingkan teks

Permyataan *if* dapat digunakan untuk membandingan teks dengan kondisi yang kita inginkan. Lihatlah contoh di bawah.

```
user name = input ("Siapa nama Anda? ")
if user_name == "Bambang":
     print ("Nama Anda bagus.")
else:
     print ("Nama Anda ok.")
```

Contoh di atas akan mengecek *string* nama yang di-input oleh pengguna dan mengecek apakah nama tersebut “Bambang".Jika pengguna meng-input nama “Bambang", maka Pythonakan menampilkan tulisan "Nama Anda bagus". Nama lain yang di-input oleh pengguna tidak akan mengeksekusi perintah dalam pernyataan *if* karena tidak sesuai dengan kondisi yang didefiniskan, sehingga akan mengeksekusi perintah dalam else.

Membandingkan teks

Perlu diingat bahwa perbandingan teks atau *string* bersifat *case-sensitive*, sehingga penulisan teks secara *lowercase* dan *uppercase* akan memengaruhinya. Jika pengguna meng-input "BAMBANG" atau "bambang", program akan mengeksekusi perintah di dalam *else*.

Agar perbandingan teks dapat bersifat *case-insensitive,* perlu dilakukan konversi untuk mengakalinya. Cara termudah adalah menggunakan perintah *lower* untuk mengubah teks menjadi *lowercase*. Dengan itu, teks akan dibandingkan dengan versi *lowercase*-nya. Kalau kita membandingan teks yang telah diubah menggunakan *lower* dengan *string* yang memiliki *uppercase*, maka kondisinya tidak akan sesuai (*match*)

Membandingkan teks

Perhatikan potongan *code* di bawah:

```
user_name = input ("Siapa nama Anda? ")
if user_name.lower () "bambang":
     print ("Nama Anda bagus.")
else:
    print ("Nama Anda ok.")
```

Jika kita ingin membandingkan beberapa teks menggunakan pernyataan *if*, kita dapat menuliskannya sebagai berikut:

```
if user_name "Bambang" or user name == "Jackson"
```

Membandingkan teks

Ubahlah pernyataan *if* pada *code* sebelumnya dengan potongan *code* di atas. Kemudian, ubahah *code* tersebut dengan potongan *code* di bawah dan cobalah pada IDE Anda. Bandingkan kedua hasilnya.

```
if user_name == "Bambang" or "Jackson"
```

Potongan *code* di atas mungkin tidak bekerja dengan baik. Karena nilai apapun yang didefinisikan pada kondisional *if* akan menghasilkan *true* jika bukan 0. Maka, "Bambang" menghasilkan *true*. "Jackson" akan selahu menghasilkan *true*.

Latihan

1. Buatlah code yang akan menerima input bilangan dari pengguna dan menampilkan tulisan ketika bilangan tersebut adalah bilangan genap, ganjil atau prima (Hint: gunakan modulus). Gunakanlah prinsip `if elif else` yang telah dipelajari!
2. Perhatikan code di bawah. Terdapat dua hal yang salah dari code tersebut. Identifikasi kesalahan tersebut dan koreksilah!

```
X == 4
if x >= 0:
      print ("x bilangan positif")
else:
      print ("x bilangan negatif")
```

Latihan

3. Perhatikan code di bawah. Terdapat beberapa hal yang salah dari code tersebut. Identifikasi kesalahan tersebut dan koreksilah! Pemrogram ingin menentukan variabel jumlah uang sesuai dengan pekerjaan yang dipilih oleh pengguna.

```
print ("A. Bankir")
print ("B. Peternak lele")
print ("C, Tukang bubur")
user_input = input ("Apakah pekerjaan Anda? ")
if user input = A:
      uang = 100
else if user_input B:
      uang = 70
else if user input = C:
      uang = 40
```

Latihan

4. Buatlah codepermainan suit untuk dua pemain. Gunakan input untuk mendapatkan pilihan gunting kertas, atau batu dari pengguna, kemudian gunakan pernyataan if untuk membandingkan pilihan kedua pemain. Ingat bahwa peraturan suit adalah sebagai berikut: Gunting mengalahkan kertas Kertas mengalahkan batu Batu mengalahkan gunting

```
ingatlah bahwa peraturan suit adalah sebagai berikut
```

- `Gunting mengalahkan kertas`
- `Kertas mengalahkan batu`
- `Batu mengalahkan gunting`

Solusi

Solusi (httns:/ ww practicenython.org/solution 2014/04/02/08-rock-paper-seissors solutions.html):

```
user1 = input ("What's your name?")
user2 = input ("And your name?")
user1_answer = input ("%s, do you want to choose rock, paper or scissors?" %user1)
user2_answer = input ("%s, do you want to choose rock, paper or scissors?" %user2)
def compare (u1, u2):
    if u1==u2:
        return ("It's a tie!")
    elif u1 == 'rock':
        if u2 == 'scissors':
            return ("Rock wins!")
        else:
            return ("Paper wins!")
    elif u1 == 'scissors':
        if u2 == 'paper':
            return ("Scissors win!")
        else:
            return ("Rock wins!")
    elif u1 == 'paper':
        if u2 == 'rock':
            return ("Paper wins!")
        else:
            return ("Scissors win!")
    else:
        return ("Invalid input! You have not entered rock, paper or scissors, try again.")


print (compare (user1_answer, user2_answer))
```

For loop

For lop **di Python digunakan untuk mengeksekusi hal-hal yang berulang. Hal ini dilakukan dengan cara mengiterasikan obyek dalam suatu sekuens, seperti list, tuple, atau string, atau obyek lainnya yang dapat diiterasikan. *For loop* ditulis dengan sintaks yang sama dengan pernyataan *if*.**

Contoh *for loop* yang digunakan untuk mengulang suatu perintah sebanyak 4 kali adalah sebagai berikut:

```
for i in range (4): print ("Hello World")
```

Jika kita menjalankan code tersebut, kita akan mendapatkan output:

```
Hello World
Hello World
Hello World
Hello World
```

For loop

Pada code di atas, kita dapat identifikasi bahwa variabel i adalah variabel yang diiterasikan sebanyak nilai yang ditentukan, yaitu4. Bagian range(4) merupakan sekuens iterasi yangdigunakan. Fungsirange(n) akan membuat list berisikan angka dari 0 hingga n.

Contoh lain dari for loop menggunakan list sebagai sekuensi iterasinya adalah sebagai berikut:

```
angka = [1, 3, 4, 9]
for i in angka:
    hasil = l + i
    print ("Hasilnya adalah ". hasil)
```

For loop

Dari code tersebut, output yang kita dapatkan:

```
Hasilnya adalah 2
Hasilnya adalah 4
Hasilnya adalah 5
Hasilnya adalah 10
```

Jika kita modifikasi variabel hasil supaya mendapatkan hasil penambahan dari semua angkapada list angka sebagai berikut:

```
for i in angka:
    hasil = hasil + i
    print ("Hasilnya adalah ", hasi1)
```

Maka output yang akan didapatkan dari code tersebut:

```
Hasilnya adalah 17
```

Nesting Loop

Nesting *loop* mengacu pada loop yang berada di dalam *loop* lainnya. *Loop* yang digunakan dapat berupa *loop* apapun, namun pada bagian ini kita akan fokus padafor *loop.*

Perhatikan code di bawah:

```
for i in range (3):
    print (“a”)
for j in range (3):
    print (“b”)
```

Cobalah pada IDE Python Anda dan lihatlah outputnya. Pada dasamya, *loop* yang berada di dalam perlu menyelesaikan iterasinyasebelum foop luar akan pindah ke iterasi selanjutnya. Kemudian, setelah *loop* luar plndah ke iterasi selanjutnya, *loop* dalam akan kembali dieksekusi hingga selesai. Hal ini akan tenis berlanjut hingga iterasi loop luar telah selesai.

Latihan

1. Buatlah code yang dapat menghasilkan output berupa pattern seperti berikut:

```
*
**
***
****
*****
```

petunjuk: gunakan nested for loop.

Solusi (Sumber: https://pynative.com/python-nested-loops/):

```
rows = 5
for i in range(1, rows + 1):
      for j in range(1, i + 1):
            print("*", end="")
      print(' ')
```

2. Buatlah program yang akan menampilkan bilangan ganjil dari 0 hingga 100 menggunakan *for loop*

3. Lihatlah *code* di bawah. Program ini meminta 3 angka dari *user* dan menampilkan jumlah ketiga angka tersebut. Temukan 3 hal yang salah dan koreksilah!

```
total = 0
for i in range(3)
      x = input("Masukkan sebuah angka: ")
      total = total + i
print("Totalnya adalah ", x)
```

While loop

While loop digunakan untuk mengulang suatu perintah ketika kondisi tertentu terpenuhi. Karena itu, format dari *while loop* sangat mirip dengan pernyataan if. Ketika suatu kondisi terpenuhi, maka *code* yang berada dalam *loop* tersebut akan terulang hingga kondisi tersebut tidak lagi terpenuhi.

Contoh dari *while loop* adalah sebagai berikut:

```
i = 0
while i < 10:
    print  (i)
    i += 1
```

Code di atas akan menjalankan *looping* ketika variabel i kurang dari 10. Pada baris pertama, kita telah mendeklarasikan bahwa i memiliki nilai awal 0. Dalam *while loop,* kita menjalankan perintah print yang dilanjutkandengan `i+=1`, yaitu sebuah operator *increment*. Operator *increment* tersebut akan menambah nilai 1 ke variabel i, karena merupakan bentuk singkat dari `i=i+1`. Sehingga, *code* dalam *while loop* akan terus berulang hingga i memiliki nilai 9.

Prediksilah output dari contoh code di bawah ini sebelum mencoba pada IDE Anda:

```
i= 1
while i <= 2 ** 32:
    print(i)
    i *= 2
```

Loop Hingga Pengguna Ingin Berhenti

- Cara menghentikan *loop* yaitu dengan menggunakan variabel Boolean sebagai *trigger event*-nya

Contoh

```
done = False
While not done:
            quit = input (“Do you want to quit(y/n)/ “)
            if quit == “y” :
                    done = True
```

- Ketika pengguna menginput "y", looping akan berakhir.
- Variabel `done` akan diubah menjadi `True` pada pernyataan `if` jika variabel *input* `quit` adalah `"y"`.
- Ingat, bahwa suatu pernyataan kondisional hanya akan dieksekusi ketika kondisinya `True`. Oleh karena itu, *while loop* akan berhenti karena kondisi `not done` akan menjadi `False` ketika pengguna memberikan input `"y"`

Loop Hingga Pengguna Ingin Berhenti

contoh sebelumnya juga dapat ditulis langsung mengguakan perintah *break,* yang berfungsi menghentikan program.

Pada program ini, variabel `done` diubah menjadi 1 agar program akan melakukan *loop* selamanya hingga pengguna memberikan input `"y"`

```
while 1:
        quit = input (Do you want to quit (y/n)? ")
        if quit == "y":
                break
```

While-else

1

Pernyataan *while loop* dapat digabungkan dengan klausa *else*

2

Klausa *else* akan mengeksekusi perintah yang ditulis di dalamnya ketika kondisi pada *while loop* sudah tidak terpenuhi.

3

Jika *loop* dihentikan dengan pernyataan *break*, maka bagian dalam klausa *else* juga tidak akan dieksekusi

While-else

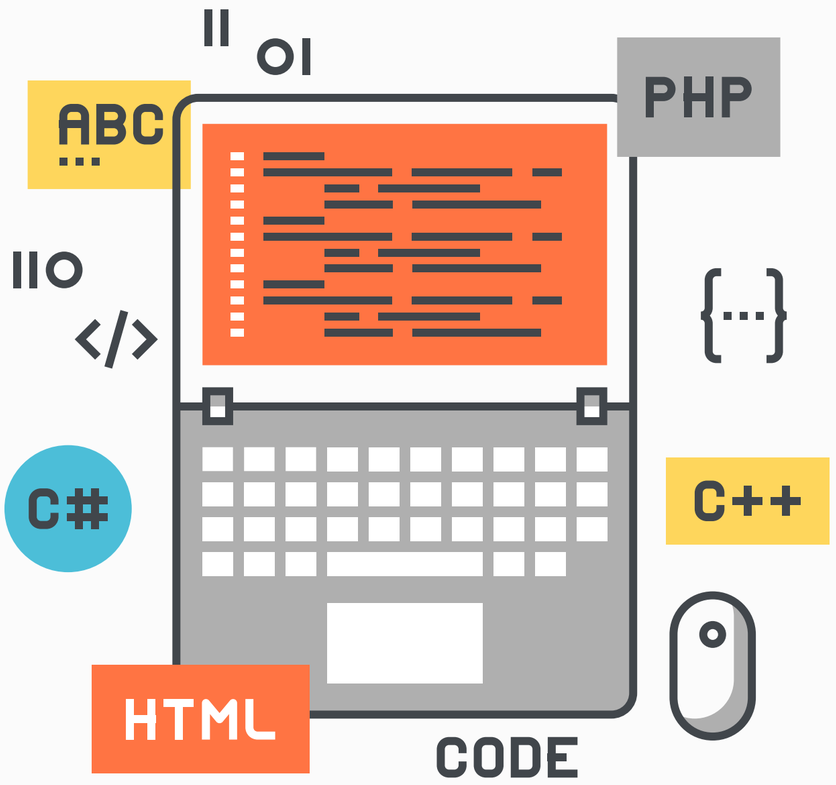
Contoh code *while loop* tanpa pernyataan *break*

```
n = 5
while n > 0:
    n -= 1
    print (n)
else:
    print ("Loop berakhir")
```

Contoh code dimana *while loop* akan dihentikan oleh pernyataan *Break*

```
n = 5
while n > 0:
    n -= 1
    print (n)
    if n == 2:
        break
else:
    print ("Loop berakhir")
```

Latihan



- Buatlah program yang akan menghitung mundur dari 10 hingga 0 mengggunakan *while loop*. Ketika mencapai 0, tampilkan tulisan "Let's go!"
- Buatlah program tebak-tebakan angka. Program akan memberikan angka random antara 1 dan 9 (termasuk angka 1 dan 9), kemudian pengguna diminta untuk menebak angka tersebut. program akan memberikan *feedback* ke pengguna ketika tebakan pengguna kurang atau lebih dari angka yang diberikan, juga ketika pengguna menebak dengan benar.

Petunjuk: gunakan fungsi *random* untuk memberikan angka random kepada pengguna

Solusi

```
import random

rd = random.randint(1,9)
guess = 0
c = 0
while guess != rd and guess != "exit":
    guess = input("Enter a guess between 1 to 9")

    if guess == "exit":
        break

    guess = int(guess)
    c += 1

    if guess < rd:
        print("Too low")
    elif guess > rd:
        print("Too high")
    else:
        print("Right!")
        print("You took only", c, "tries!")
input ()
```