**Percabangan dan Ternary Operators**

Beberapa contoh aplikasi dasar untuk pengenalan percabangan yang sering digunakan adalah:

* Genap atau Ganjil.
* Positif atau Negatif.
* Positif, Negatif, atau Nol.

**If**

Seperti bahasa pemrograman lainnya, Python juga memiliki statemen percabangan IF. Di Python, expression diletakkan setelah if, dan keputusan ditentukan berdasarkan nilai kebenaran dari expression tersebut.

Tip: Python menganggap setiap nilai non-zero dan non-null sebagai True dan nilai zero/null sebagai False.

Jika expression dievaluasi sebagai True, maka blok statement di dalam if statement akan dieksekusi. Sesuai konvensi, blok ini memiliki indentasi masuk setelah tanda titik dua (:). Jika expression dievaluasi sebagai False, maka blok selanjutnya (setelah statement IF) yang akan dijalankan. Contoh:

1. kelerengku = 10
2. if kelerengku:
3. print ("Cetak ini jika benar")
4. print (kelerengku)

Output:

|  |
| --- |
| Cetak ini jika benar 10 |

Anda dapat menyingkat penulisan statement yang akan dieksekusi jika ia terwakili dalam 1 baris. Misalnya:

1. if kerelengku: hitung()

**Else**

Statement Else dapat dikombinasikan dengan IF Statement, sebagai jalan keluar saat kondisi / hasil evaluasi bernilai False. Else bersifat opsional dan tunggal. Mari kita implementasikan dalam kasus pengecekan tinggi badan pengunjung di suatu wahana.

1. tinggi\_badan = int(input("Masukkan tinggi badan Anda : "))
2. if tinggi\_badan>=160:
3. print ("Silakan, Anda boleh masuk")
4. else:
5. print ("Maaf, Anda belum boleh masuk")

Output 1:

|  |
| --- |
| Masukkan tinggi badan Anda : 160 Silakan, Anda boleh masuk |

Output 2:

|  |
| --- |
| Masukkan tinggi badan Anda : 140 Maaf, Anda belum boleh masuk |

Mari implementasikan pada kasus yang berbeda, kali ini kasusnya adalah pengecekan bilangan ganjil atau genap pada suatu variabel.

1. bilangan = 4
2. if bilangan % 2 == 0:
3. print('Bilangan {} adalah genap'.format(bilangan))
4. else:
5. print('Bilangan {} adalah ganjil'.format(bilangan))

Output:

|  |
| --- |
| Bilangan 4 adalah genap |

**Elif - Alternatif untuk Switch/Case dan IF bertingkat di python**

Elif adalah kependekan dari else if, dan merupakan alternatif untuk if bertingkat atau switch/case di beberapa bahasa pemrograman lain. Sebuah IF Statement dapat diikuti satu atau lebih statement elif (opsional & tidak dibatasi). Mari kita implementasikan pada kasus penilaian tugas siswa.

1. nilai = int(input("Masukkan nilai tugas Anda : "))
2. if nilai>80:
3. print("Selamat! Anda mendapat nilai A")
4. print("Pertahankan!")
5. elif nilai>70:
6. print("Hore! Anda mendapat nilai B")
7. print("Tingkatkan!")
8. elif nilai>60:
9. print("Hmm.. Anda mendapat nilai C")
10. print("Ayo semangat!")
11. else:
12. print("Waduh, Anda mendapat nilai D")
13. print("Yuk belajar lebih giat lagi!")

Output 1:

|  |
| --- |
| Masukkan nilai tugas Anda : 85 Selamat! Anda mendapat nilai A Pertahankan! |

Output 2:

|  |
| --- |
| Masukkan nilai tugas Anda : 75 Hore! Anda mendapat nilai B Tingkatkan! |

Output 3:

|  |
| --- |
| Masukkan nilai tugas Anda : 65 Hmm.. Anda mendapat nilai C Ayo semangat! |

Output 4:

|  |
| --- |
| Masukkan nilai tugas Anda : 30 Waduh, Anda mendapat nilai D Yuk belajar lebih giat lagi! |

Catatan: Jika sudah memenuhi salah satu kondisi if/elif, maka program akan keluar dari blok IF Statement. Anda harus memastikan urutan secara logika, IF, Elif, dan Else dalam tingkatan yang tepat. Contoh yang perlu diperhatikan adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| Input Nilai | #Kasus 1  if nilai>80: print("Selamat! Anda mendapat nilai A") print("Pertahankan!") elif nilai>70: print("Hore! Anda mendapat nilai B") print("Tingkatkan!") elif nilai>60: print("Hmm.. Anda mendapat nilai C") print("Ayo semangat!") else: print("Waduh, Anda mendapat nilai D") print("Yuk belajar lebih giat lagi!") | #Kasus 2  if nilai>0: print("Selamat! Anda mendapat nilai A") print("Pertahankan!") elif nilai<80: print("Hore! Anda mendapat nilai B") print("Tingkatkan!") elif nilai<70: print("Hmm.. Anda mendapat nilai C") print("Ayo semangat!") else: print("Waduh, Anda mendapat nilai D")     print("Yuk belajar lebih giat lagi!") |
| 85 | Masukkan nilai tugas Anda: 85 Selamat! Anda mendapat nilai A Pertahankan! | Masukkan nilai tugas Anda: 85 Selamat! Anda mendapat nilai A Pertahankan! |
| 65 | Masukkan nilai tugas Anda: 65 Hmm.. Anda mendapat nilai C Ayo semangat! | Masukkan nilai tugas Anda: 65 Selamat! Anda mendapat nilai A Pertahankan! |
| 30 | Masukkan nilai tugas Anda: 30 Waduh, Anda mendapat nilai D Yuk belajar lebih giat lagi! | Masukkan nilai tugas Anda: 30 Selamat! Anda mendapat nilai A Pertahankan! |

Pada #Case 2, elif dan else tidak pernah dijalankan karena nilai berapapun (yang bernilai positif) akan selalu masuk pada IF (klausa pertama).

Mari kita implementasikan pada kasus pengecekan bilangan positif, negatif, atau nol.

1. bilangan = -3
2. if bilangan > 0:
3. print('Bilangan {} adalah positif'.format(bilangan))
4. elif bilangan < 0:
5. print('Bilangan {} adalah negatif'.format(bilangan))
6. else:
7. print('Bilangan {} adalah nol'.format(bilangan))

Output:

|  |
| --- |
| Bilangan -3 adalah negatif |

**Ternary Operators**

Ternary operator lebih dikenal sebagai conditional expressions pada Python. Operator menentukan sesuatu berdasarkan kondisi True atau False. Jika statement atau klausa if Anda cukup sederhana, maka ternary Operators akan sangat membantu.

Perbandingan klausa IF dengan ternary Operators:

| **IF** | **Ternary** |
| --- | --- |
| if (condition): condition\_if\_true else: condition\_if\_false | condition\_if\_true if condition else condition\_if\_false |
| lulus = True if (lulus): kata = ”selamat” else: kata = “perbaiki” | lulus = True kata = "selamat" if lulus else "perbaiki" |

Opsi lain dari ternary operators melibatkan tuples. Contoh kodenya berikut:

| **IF** | **Ternary\_Tuples** |
| --- | --- |
| if (condition): condition\_if\_true else: condition\_if\_false | (condition\_if\_false, condition\_if\_true)[condition] |
| lulus = True if (lulus): kata=”selamat” else: kata=“perbaiki” | nice = True kata= ("perbaiki", "selamat")[lulus] |

Pada tuple ini, dimanfaatkan nilai [0] sebagai False dan [1] sebagai True.

Aplikasi kedua ini menurut beberapa aktivis kurang ‘pythonic’, salah satunya karena cukup membingungkan untuk meletakkan klausa saat True atau False. Selain itu, kedua nilai akan tetap dievaluasi walaupun hanya dibutuhkan salah satunya. Lihat contoh berikut:

1. kondisi = **True**
2. print(2 **if** kondisi**else** 1/0)
3. #Output is 2
5. print((1/0, 2)[kondisi])
6. #Eror Pembagian Nol akan muncul

Ternary-tuples sebaiknya dihindari, terutama untuk kode (dan klausa True/False) yang kompleks. Ternary dapat digunakan untuk menyingkat kode saat klausa True/False Anda cukup pendek - misalnya sebuah fungsi tanpa parameter.

**ShortHand Ternary**

Selain Ternary Operators, dikenal juga shorthand ternary tag yang mungkin membantu Anda untuk memeriksa kode/hasil dari sebuah fungsi dan memastikan outputnya tidak menyebabkan error (atau minimal memberikan informasi relevan saat error):

1. hasil = None
2. pesan = hasil or "Tidak ada data"
3. print(pesan)

Output:

|  |
| --- |
| Tidak ada data |

## Perulangan

### For

Seperti di bahasa pemrograman lainnya, Python juga memiliki fungsi for. Bedanya di Python, For tidak hanya untuk perulangan dengan jumlah finite (terbatas), melainkan lebih ke fungsi yang dapat melakukan perulangan pada setiap jenis variabel berupa kumpulan atau urutan. Variabel yang dimaksud bisa berupa list, string, ataupun range. Jika sebuah list atau urutan berisi expression, maka Ia akan dievaluasi terlebih dahulu. Kemudian item pertama pada urutan/list akan diassign sebagai variabel iterating\_var. Setelahnya, blok statement akan dieksekusi, berlanjut ke item berikutnya, berulang, hingga seluruh urutan habis.

1. for huruf in 'Dicoding':  # Contoh pertama
2. print('Huruf: {}'.format(huruf))
4. flowers = ['mawar', 'melati', 'anggrek']
5. for flower in flowers:  # Contoh kedua
6. print('Flower: {}'.format(flower))

Output:

|  |
| --- |
| Huruf : D Huruf : i Huruf : c Huruf : o Huruf : d Huruf : i Huruf : n Huruf : g Flower: mawar Flower: melati Flower: anggrek |

Anda juga dapat melakukan perulangan berdasarkan indeks atau range dengan memanfaatkan fungsi len():

1. flowers = ['mawar', 'melati', 'anggrek']
2. for index in range(len(flowers)):
3. print('Flowers: {}'.format(flowers[index]))

Output:

|  |
| --- |
| Flower : mawar Flower : melati Flower  : anggrek |

### While

While pada bahasa Python digunakan untuk mengeksekusi statement selama kondisi yang diberikan terpenuhi (True). Kondisi dapat berupa expression apapun, dan harap diingat bahwa **True di Python termasuk semua nilai non-zero**. Saat kondisi menjadi False, program akan melanjutkan ke baris setelah blok statement.

Tip: Python tidak memiliki do.. while statement

Seperti for dan semua statement percabangan, blok statement yang mengikuti kondisi while dan memiliki posisi indentasi yang sama, dianggap blok statement yang akan dieksekusi.

Contoh:

1. count = 0
2. while (count < 7):
3. print('Hitungannya adalah: {}'.format(count))
4. count = count + 1

Output:

|  |
| --- |
| Hitungannya adalah: 0 Hitungannya adalah: 1 Hitungannya adalah: 2 Hitungannya adalah: 3 Hitungannya adalah: 4 Hitungannya adalah: 5 Hitungannya adalah: 6 |

Seperti pada bahasa lainnya, eksekusi statement while mungkin bersifat infinit / infinite loop saat sebuah kondisi tidak pernah bernilai False. Contohnya sebagai berikut:

1. var = 1
2. while var == 1:  # This constructs an infinite loop
3. num = input('Masukkan angka: ')
4. print('Anda memasukkan angka: {}'.format(num))

7. while True:  # This constructs an infinite loop
8. num = input('Masukkan angka: ')
9. print('Anda memasukkan angka: {}'.format(num))

Potongan kode di atas tidak akan pernah bernilai False karena nilai var tidak pernah berubah. Untuk menghentikan infinite loop, gunakan CTRL (atau CMD⌘) - C untuk menghentikannya dan keluar dari program.

Anda juga dapat menyingkat penulisan blok statement While jika statement Anda cukup terwakili oleh satu baris.

1. while (var1): do\_something()

### Perulangan Bertingkat

Ada kalanya Anda perlu untuk melakukan perulangan bertingkat, misalnya untuk menghasilkan contoh print-out berikut:

|  |
| --- |
| \*\*\*\*\*\* \*\*\*\*\* \*\*\*\* \*\*\* \*\* \* |

Anda dapat melakukannya dengan kode berikut:

1. for i in range(0, 6):
2. for j in range(0, 6 - i):
3. print('\*', end='')
4. print()

Tip: Tambahkan parameter end pada print untuk mengatur karakter yang mengakhiri pencetakan string/teks Anda. Secara default, karakter end ini adalah newline ('\n').