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 kelerengku: hitung()
```

Else

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 (opsion dibatasi). Mari kita implementasikan pada kasus penilaian tugas siswa.

```
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:

Nilai		

	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!")
		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).

Pertanyaannya, bagaimana kalau bilangan yang kita masukkan bernilai negatif atau nol? Mari kita lihat hasilnya pada kasus pengecekan bilangan 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:

Yup! Anda tentu sudah menduga, pada kasus pengecekan bilangan negatif, perintah elif yang dijalankan.

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”
	lulus = 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 membin
untuk meletakkan klausa saat True atau False. Selain itu, kedua nilai akan tetap dievaluasi walaupu
dibutuhkan salah satunya. Lihat contoh berikut:

```
print(2 if kondisi else 1/0)
#Output is 2

print((1/0, 2)[kondisi])
#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):

```
hasil = None
pesan = hasil or "Tidak ada data"
print(pesan)
```

Output:

```
Tidak ada data
```

Sebelumnya

Selanjutnya

Belum muncul