Penanganan Pengecualian

Pada aplikasi Python yang Anda buat bisa dilengkapi dengan penanganan terhadap pengecualian (exceptions handling) dari kelompok (tipe) kesalahan yang Anda tentukan. Proses penanganan pengecualian menggunakan pernyataan try yang berpasangan dengan *except*.

Misalnya kita ingin menangani pengecualian yang terjadi jika ada pembagian angka dengan nilai nol (0).

```
1. z = 0
2. 1 / z
```

Output:

```
1. Traceback (most recent call last):
2. File "<stdin>", line 1, in <module>
3. ZeroDivisionError: division by zero
```

Kode di atas memberikan output eror karena pembagian dengan nilai 0 (kita tidak dapat membagi dengan nilai 0).

Selanjutnya, perhatikanlah contoh berikut:

```
1. z = 0
2. try:
3.   x = 1 / z
4.   print(x)
5. except ZeroDivisionError:
6.   print('tidak bisa membagi angka dengan nilai nol')
```

Pada contoh tersebut, operasi aplikasi berhenti di x = 1 / z dan bagian print(x) tidak sempat dioperasikan. Hal ini karena aplikasi sudah mengalami pengecualian sehingga yang tercetak adalah operasi print(‘tidak bisa membagi angka dengan nilai nol’).

Pada operasi yang dicontohkan di atas, penanganan pengecualian untuk ZeroDivisionError dilakukan sehingga aplikasi tidak lagi keluar dari eksekusi karena kesalahan, tapi digantikan dengan mencetak pesan ke layar. Pada contoh ini kita fokus pada penanganan pengecualian, meskipun ada cara lain untuk menyelesaikannya, misal menggunakan kondisi (percabangan) untuk menghindari nilai nol.

DIBANTU

sebagai tuple satu elemen, jangan lupa dalam menuliskan tuple satu elemen harus tetap diakhiri dengan koma.

```
>>> try:
...     with open('contoh_tidak_ada.py') as file:
...         print(file.read())
... except (FileNotFoundError, ):
...     print('file tidak ditemukan')
...
file tidak ditemukan
```

Pada operasi di atas, aplikasi akan membuka dan mengakses file bernama contoh_tidak_ada.py, tetapi file tersebut tidak ada di direktori dimana aplikasi Python tersebut berada, selanjutnya akan terjadi pengecualian (exceptions) tetapi ditangani, dalam pasangan pernyataan try dan except, sehingga aplikasi tidak terhenti tetapi tercetak di layar bahwa file tidak ditemukan.

Dalam aplikasi yang lebih kompleks, penanganan pengecualian dapat menggunakan pernyataan except lebih dari satu. Di contoh berikutnya akan menggunakan pernyataan except lebih dari satu (untuk satu pernyataan try), maupun menggunakan satu pernyataan except yang menangani lebih dari satu tipe kesalahan yang digabung dalam sebuah tuple.

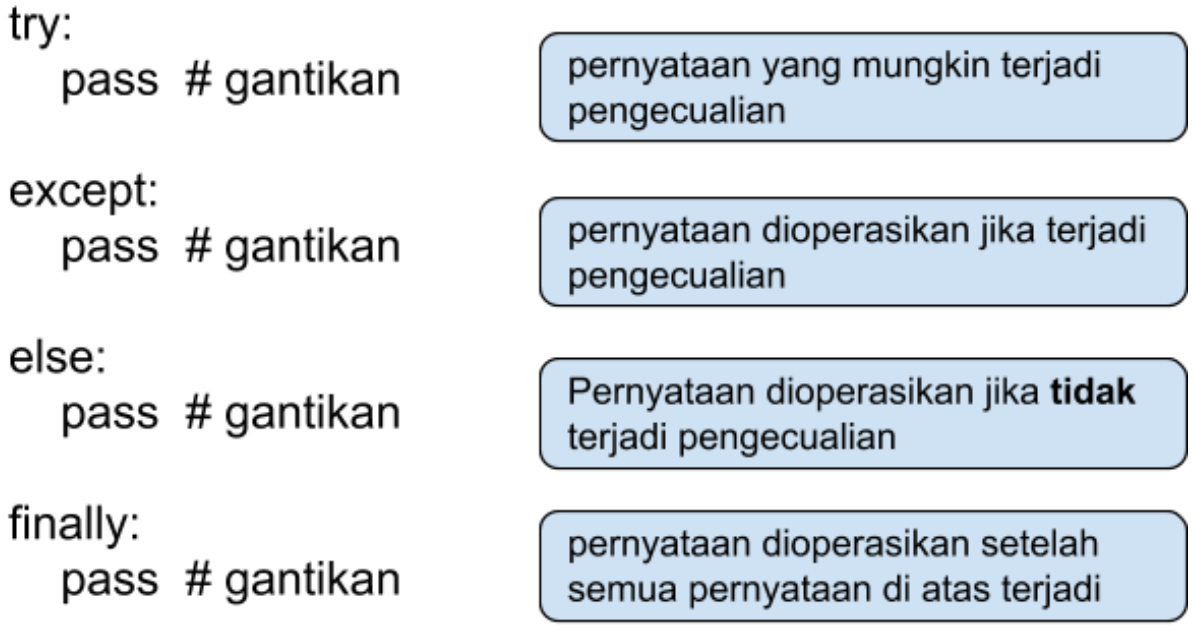
```
>>> d = {'ratarata': '10.0'}
>>> try:
...     print('rata-rata: {}'.format(d['rata_rata']))
... except KeyError:
...     print('kunci tidak ditemukan di dictionary')
... except ValueError:
...     print('nilai tidak sesuai')
...
kunci tidak ditemukan di dictionary
>>> try:
...     print('rata-rata: {}'.format(d['ratarata']/3))
... except KeyError:
...     print('kunci tidak ditemukan di dictionary')
... except (ValueError, TypeError):
...     print('nilai atau tipe tidak sesuai')
```

Pada contoh tersebut, yang paling awal terjadi pengecualian untuk tipe kesalahan KeyError karena dalam dictionary `d` tidak memiliki kunci (key) `rata_rata`, yang ada adalah kunci `ratarata`.

buah pernyataan `except` menangani tipe kesalahan ValueError atau TypeError, sehingga cocok salah satunya akan menampilkan ke layar bahwa nilai atau tipe tidak sesuai.

Di bagian paling akhir contoh, terjadi pengecualian untuk tipe kesalahan ValueError karena berusaha melakukan konversi (casting) dari sebuah string ke integer dengan format yang tidak sesuai (bilangan bulat seharusnya tidak memiliki titik dalam penulisannya). Dalam penulisan penanganan kesalahannya digunakan variasi lain untuk mendapatkan pesan kesalahan sebagai variabel `e` untuk kemudian variabel tersebut dicetak dalam pesan yang ditampilkan ke layar.

Bentuk lengkap dari pernyataan `try` dapat dilihat pada bagan berikut, terdiri dari pernyataan `except`, `else`, `finally`.



Menghasilkan Pengecualian

Dalam membuat aplikasi, ada kemungkinan Anda butuh untuk menghasilkan pengecualian (raise exceptions), salah satu caranya bisa dengan menggunakan pengecualian yang sudah ada, hanya ditambahkan informasi detailnya saja.

Misalnya dalam contoh berikut, Anda mewajibkan sebuah dictionary memiliki kunci (key) total.

```
>>> d = {'ratarata': '10.0'}
>>> if 'total' not in d:
...     raise KeyError('harus memiliki total')
...
Traceback (most recent call last):
 File "<stdin>", line 2, in <module>
KeyError: 'harus memiliki total'
```

DIBANTU

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Decode Ideas
Discover Potential

Tentang Kami

Blog
Reward
Showcase

Hubungi Kami
FAQ

Penghargaan

image
click bila
belum muncul

© 2022 Dicoding | Dicoding adalah merek milik PT Presentologics, perusahaan induk dari PT Dicoding Akademi Indonesia.

DIBANTU