Pertemuan 13 (Praktikum)

EXCEPTION HANDLING

1. Konsep Exception Handling
   - Ketika suatu kesalahan terjadi sewaktu suatu pernyataan dieksekusi, Python segera menghentikan eksekusi dan menampilkan pesan kesalahan.
   - Python memunculkan eksepsi sehingga muncul pesan kesalahan dan eksekusi segera dihentikan dan pernyataan berikutnya tidak dijalankan.
   - Eksepsi berarti kesalahan yang terjadi ketika suatu pernyataan dieksekusi. Keadaan seperti ini bisa terjadi kapan saja ketika suatu perintah dieksekusi oleh interpreter Python. Penyebabnya bisa bervariasi seperti berikut ini:
     » Objek yang dilibatkan dalam suatu perintah tidak ada
     » Terjadi pembagian dengan bilangan nol
     » Konversi ke bilangan bulat tidak dapat dilakukan karena string yang dikonversi tidak menyatakan bilangan

2. Jenis Eksepsi

   Daftar eksepsi pada Python dapat diketahui dengan menuliskan perintah:

```
dir(__builtin__)
```

3. Penanganan Eksepsi

   Penanganan eksepsi (exception handling) dimaksudkan untuk menangani keadaan ketika suatu eksepsi dimunculkan. Penulisan pernyataan seperti berikut ini:

```
try:
     pernyataan-1
except [nama_eksepsi]:
     pernyataan-2
[else:
     Pernyataan-3
]
```

   Program untuk menangani masukan:

```
# Penanganan Eksepsi (menangani masukan)

while True:
    try:
        bil=int(input('Masukan bilangan: '))
        break
    except ValueError:
        print('Salah dalam memasukkan bilangan!')
print('Bilangan =', bil)
```

```
# Penanganan Eksepsi (menangani masukan)

while True:
    try:
        bil=int(input('Masukan bilangan: '))
    except ValueError:
        print('Salah dalam memasukkan bilangan!')
    else:
        break
print('Bilangan =', bil)
```

Pesan bawaan untuk suatu eksepsi bisa ditampilkan dengan menambahkan **as e** setelah nama ekspesi, seperti program berikut ini:

```
# Penanganan Eksepsi (menampilkan pesan bawaan eksepsi)

while True:
    try:
        bil=int(input('Masukan bilangan: '))
        break
    except ValueError as e:
        print(str(e))
print('Bilangan =', bil)
```

Bagian except bisa lebih dari satu. Hal ini bermanfaat untuk menangani beberapa jenis eksepsi, seperti pada program berikut ini:

```
# Penanganan Eksepsi (multicatch)

while True:
    try:
        bil=int(input('Masukan bilangan: '))
    except ValueError:
        print('Salah dalam memasukkan bilangan!')
    except KeyboardInterrupt:
        #print()
        print('Interupsi kernel tidak diperbolehkan')
print('Bilangan =', bil)
```

4. Penggunaan Pass pada Except

Pernyataan **pass** disertakan apabila tidak dikehendaki untuk melakukan tindakan apa-apa sewaktu suatu eksepsi terjadi. Berikut contoh programnya:

```
# Penanganan Eksepsi (penggunaan pass)

infoMobil = ["Avanza",2018,"Biru",1300,5]
jumlah=0

for nilai in infoMobil:
    try:
        bilangan=int(nilai)
        jumlah=jumlah+1
    except ValueError:
        pass
print('Jumlah elemen berupa bilangan:', jumlah)
```

5. Pernyataan Try…..Finally

Pada bagian finally berisi pernyataan-pernyataan yang akan selalu dieksekusi meskipun terdapat eksepsi pada pernyataan ataupun tidak. Berikut bentuk lengkapnya:

```
try:
    pernyataan-1
[except [nama_eksepsi]:
    pernyataan-2
```

```
else:
        pernyataan-3
]
finally:
        pernyataan
```

Berikut contoh penggunaan dalam program:

```
# Penggunaan try...finally
# Pembacaan data berkas teks

import sys
if len(sys.argv)==1:
    print('Masukkan nama berkas yang ingin dibaca')
else:
    berkas=sys.argv[1]
    try:
        bukaberkas=open(berkas,'r')
        konten=bukaberkas.read()
        print('Isi berkas:')
    except FileNotFoundError:
        print('*****', berkas, 'tidak ada')
    finally:
        print('Operasi pembacaan berkas telah berakhir')
print('*****')
```

6. Pemunculan Eksepsi

Adakalanya diperlukan untuk melontarkan eksepsi jika terdapat keadaan yang tidak valid. Hal ini diperlukan ketika membuat suatu fungsi.

Pemunculan eksepsi dilakukan dengan pernyataan **raise**. Bentuk pemakaiannya sebagai berikut:

```
# Contoh pemakaian raise versi 2

def faktorial(n):
    if n<0:
        raise Exception('Argumen faktorial() harus ' + 'berupa bilangan positif')
    hasil=1
    for bil in range(1, n+1):
        hasil=hasil*bil;
    return hasil

def infoFaktorial(n):
    try:
        print(n, '!=', faktorial(n), sep='')
    except:
        print('Argumen', n, 'tidak diperkenankan')

#Bagian utama
infoFaktorial(6)
infoFaktorial(-6)
print('--------------')
```

7. Penggunaan AssertionError

Eksepsi bernama AssertionError dapat dimanfaatkan untuk menghentikan eksekusi skrip jika kondisi tertentu dijumpai. Bentuk pernyataan sebagai berikut:

```
assert(kondisi), 'pesan error'
```

Berikut contoh penggunaan di program:

```python
def luas_segiempat(x0, y0, x1, y1):
    #Menghitung luas segi empat berdasarkan koordinat 2 titik: pojok kiri bawah (x0, y0), dan kanan atas (x1, y1).
    # untuk memastikan koordinat input sudah sesuai asumsi
    assert x0 < x1 and y0 < y1, "koordinat tidak valid"

    deltax = x1 - x0
    deltay = y1 - y0
    return deltax * deltay

segi4_1 = luas_segiempat(1, 1, 3, 3)
segi4_2 = luas_segiempat(2, 4, 3, 3)
print(segi4_1 + segi4_2)
```