Exception

Dasar-Dasar Pemrograman 1

Raja Oktovin P. Damanik

Motivation

- Able to use simple exception

Errors

- Syntax Error: the code cannot be interpreted; cannot be executed
- Runtime Error: the execution fails
- Logic Error: can be run and produces output, but a false one

When to use exception

- Some runtime errors cannot be handled using if [or when handled using if can be extremely tedious].
- Example:
  - User wants to read a file with file name that does not exist causing **FileNotFoundError**.
  - You already perform some type conversion such as int("2.1") but it still fails, causing **ValueError**.
- Some runtime errors should be handled with simple if-else. Some can even be handled without "manual check". Analyse how the program executes and prove that the condition that might cause error can never happen!
  - Before performing division x/y, check first if y ==0. Failing to do so might result in **DivisionByZeroError**.
  - Before accessing an index i using x[i] that is out of range, check first if is in the correct range. Failing to do so might cause **IndexError**.

When to use exception

- If the runtime error occurs, the program does not terminate gracefully.
- Using exception, we can:
  - Modify the error message to our user (e.g. to be more informative or specific).
  - Continue and control the program flow (ask user to input again).
  - Terminate the program gracefully.

More types of error

Exception	Explanation
KeyboardInterrupt	Raised when user hits Ctrl-C, the interrupt key
OverflowError	Raised when a floating-point expression evaluates to a value that is too large
ZeroDivisionError	Raised when attempting to divide by 0
IOError	Raised when an I/O operation fails for an I/O-related reason
IndexError	Raised when a sequence index is outside the range of valid indexes
NameError	Raised when attempting to evaluate an unassigned identifier (name)
TypeError	Raised when an operation of function is applied to an object of the wrong type
ValueError	Raised when operation or function has an argument of the right type but incorrect value

Examples

- Coba jalankan masing-masing potongan kode berikut. Anda seharusnya pernah menemukan kesalahan sejenis saat menulis program.

```
my_file = open('my_file.txt', 'r')
print('apasaja', file = my_file)
my_file.close()
```

```
text = 'Gojo Satoru'
print(word)
```

```
import math
print("A simple program for showing overflow error.")
print("The exponential value is:", math.exp(9999999999999))
```

```
number = input('Masukkan angka:')
print(number / 25)
```

```
text = 'Gojo Satoru'
print(text[100])
```

```
x = 8
y = 0
print("x/y=", x/y)
```

```
my_file = open('my_file.txt', 'r')
print('apasaja', file = my_file)
my_file.close()
```

```
word = input('Masukkan kata:')
convert_word = int(word)
```

Exception Handling

Basic idea:

- ➔ Keep watch on a particular section of code
- ➔ If we get an exception, raise/throw that exception (**let it be known**)
- ➔ Look for a catcher that can handle that kind of exception
- ➔ If found, handle it, otherwise let Python handle it (which usually halts the program)

```
a = 6
b = 0
# simple use of try-except block for handling errors
try:
    g = a/b
except ZeroDivisionError:
    print("This is a DIVIDED BY ZERO error")

print("This is a following statement")
print("This is another following statement")
```

```
try:
      # suite
except a_particular_error:
      # suite
```

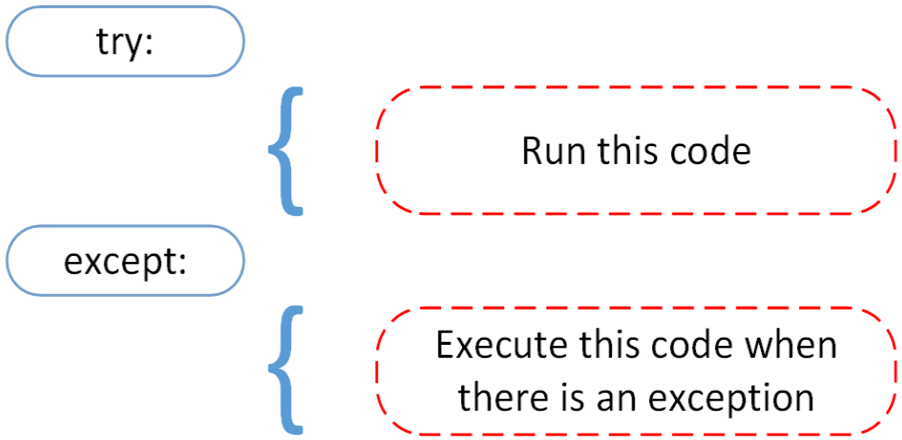
```
a = 6
b = 0
# simple use of try-except block for handling errors
try:
    g = a/b
except ZeroDivisionError:
    print("This is a DIVIDED BY ZERO error")

print("This is a following statement")
print("This is another following statement")
```

The try suite contains code that we want to **monitor for errors** during its execution.

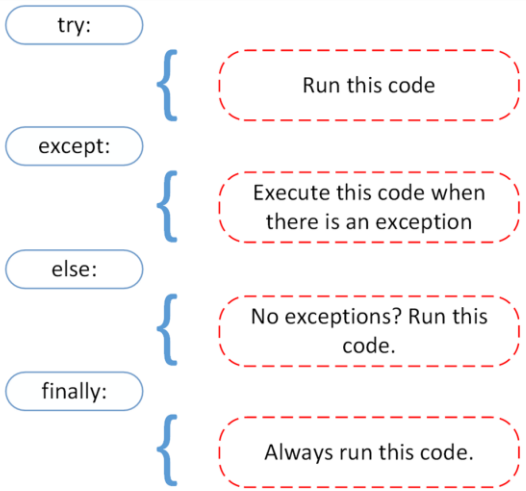
- ➔ If an error occurs anywhere in that try suite, Python **looks for a handler** that can deal with the error.
- ➔ If no special handler exists, Python handles it, meaning the program **halts** and with an error message as we have seen so many time

- ➔ An `except` suite (perhaps multiple `except` suites) is associated with a `try` suite.
- ➔ Each exception names a **type of exception** it is monitoring for.
- ➔ If the error that occurs in the `try` suite **matches the type of exception**, then that `except` suite is **activated**

General Exception



Using try-except-else-finally




```
is_prompt = True
print('Ini adalah program penghitung pembagian dengan 25.')
while(is_prompt):
      try:
            number = int(input('Masukkan angka: '))
            print(number / 25)
      except ValueError:
            print('Terjadi value error. Coba lagi')
      else:
            print('Yatta~! tidak ada Error ^^')
      finally:
            print('Ada ngga ada error, tetap dijalanin’)

      ask = input('Anda ingin melanjutkan? Y/N: ')
      if(ask == 'N'):
            is_prompt = False
print('Program selesai.')
```

Example 1: Computing Average Arbitrarily Many Positive Integers

Write a program that asks the user to input several **positive integers**; The number of integers that the user input can be arbitrary. When the user inputs -1, the program stops and compute their average. Handles the exception that might occur!

Example 2: File Not Found

Write a program that asks the user to input the name of a text file in the same directory of the program. The program then searches the whole program

Latihan

Buat sebuah program yang menerima masukan dua kali dari user berupa nama file teks yang ingin dibaca. Diasumsikan file teks berada pada direktori yang sama dengan program Anda. Program kemudian mencetak file teks baru dengan format nama [namafileteks1]_[namafileteks2]_merged.txt yang berisi isi file teks 1 diikuti isi file teks 2. Tangani errornya sehingga jika pengguna tidak memasukkan nama file yang benar, program memberi informasi bahwa "Maaf, file [nama file masukan user] tidak dapat ditemukan." lalu minta masukan lagi ke user. Ini berlaku untuk nama file pertama maupun kedua.

Contoh:

Masukkan nama file: bagu.txt

Maaf, file bagu.txt tidak dapat ditemukan.

Masukkan nama file: bagus.txt

Masukkan nama file: keren

Maaf, file keren tidak dapat ditemukan.

Masukkan nama file: keren.txt

File bagus_hebat_merged.txt sudah berhasil dibuat.