FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

Topic 5.2: Exception Handling

CSGE601020 - Dasar-Dasar Pemrograman 1

Lintang Matahari Hasani, S.Kom., M.Kom. | Dr.Eng. Lia Sadita, S.Kom., M.Eng.

Acknowledgement

This slide is an adapted version of **Exception Handling** slides used in DDP1 Course (2020/2021) by Hafizh Rafizal Adnan, M.Kom, and **Files and Exceptions I** slides by Punch and Enbody (2013)

Some additional contents, illustrations and visual design elements are provided by **Lintang Matahari Hasani, M.Kom.**

In this session, you will learn ...

Errors in Programming

Exception Overview

Error Names

Exception Handling Using `try` and `except`

Errors in Programming (1)

There are several types of error in programming:

Syntax errors
Errors where the program does not follow the rules of Python. For example: we forgot a colon, we did not provide an end parenthesis

Runtime errors
Errors during program execution. For example: dividing by 0, accessing a character past the end of a string

Semantic/logic errors:
The program is successfully executed.
Errors due to incorrect algorithms.

```
prins('DDP1')
```

...a

```
a = True
if a:
      print('amogus'))
```

...b

```
a = 'amogus'
print('among us <<<< {:^10s}'.format(a)
```

...c

```
      x = 0
if x:
print('amogus')
```

...d

Errors in Programming (2)

There are several types of error in programming:

Syntax errors
Errors where the program does not follow the rules of Python. For example: we forgot a colon, we did not provide an end parenthesis

Runtime errors
Errors during program execution. For example: dividing by 0, accessing a character past the end of a string

Semantic/logic errors:
The program is successfully executed.
Errors due to incorrect algorithms.

```
a = 10
b = 0
print(a / b)
```

...a

```
a = 'Mantappu Jiwa!'
print(a[100])
```

...b

```
print('This is {test!s:10s}'.format(sample = 123))
```

...c

Errors in Programming (3)

There are several types of error in programming:

Syntax errors
Errors where the program does not follow the rules of Python. For example: we forgot a colon, we did not provide an end parenthesis

Runtime errors
Errors during program execution. For example: dividing by 0, accessing a character past the end of a string

Semantic/logic errors:
The program is successfully executed.
Errors due to incorrect algorithms.

Suppose we want to reverse a string:

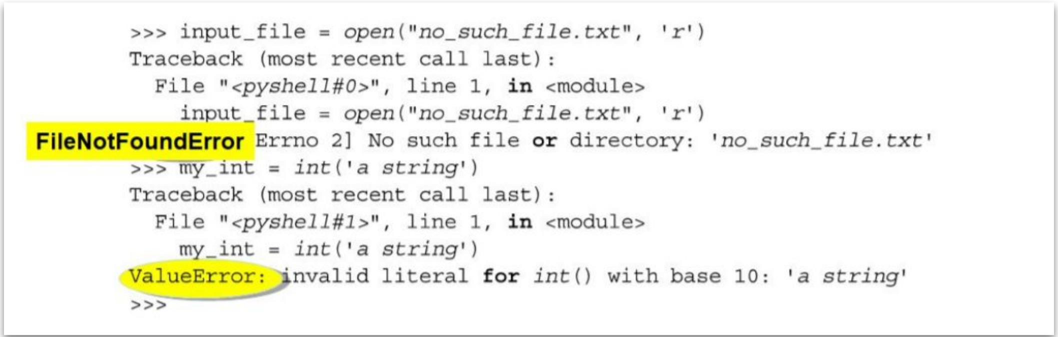
```
a = 'Mantappu Jiwa!'
new_str = ''
for x in range(0, len(a) - 1):
      new_str = new_str + a[x]

print(new_str)
```

Suppose we want to print a formatted string with:
30 spaces and align-center

```
new_str = input('Enter a word:')
print(f'Entered word: {new_str:<10s}')
```

Error Names (1)

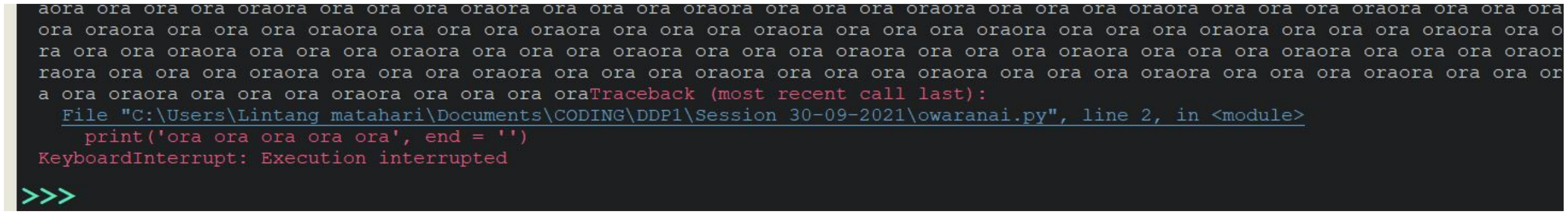
Errors have specific **names**, and Python shows them to us all the time.



```
>>> input_file = open("no_such_file.txt", 'r')
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    input_file = open("no_such_file.txt", 'r')
FileNotFoundError Errno 2] No such file or directory: 'no_such_file.txt'
>>> my_int = int('a string')
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    my_int = int('a string')
ValueError: invalid literal for int() with base 10: 'a string'
>>>
```

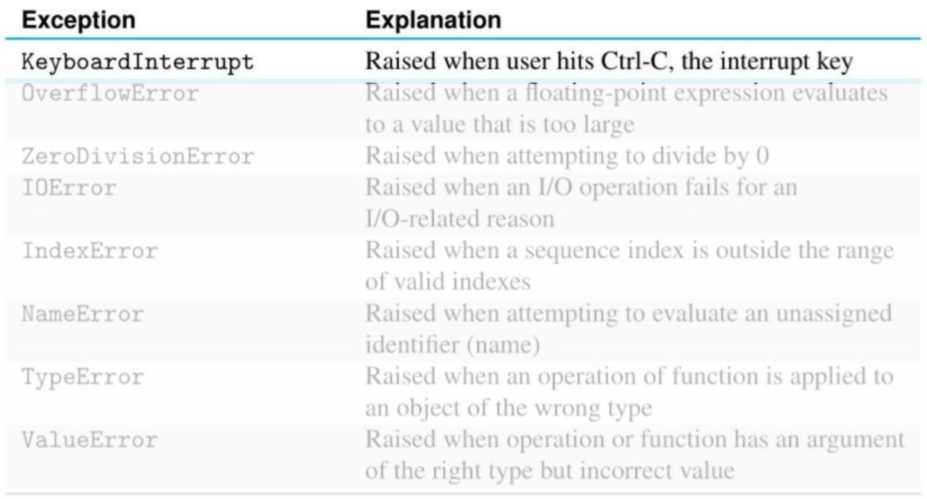
We can redefine the error message or action to handle the error

Error Names (2): `KeyboardInterrupt`



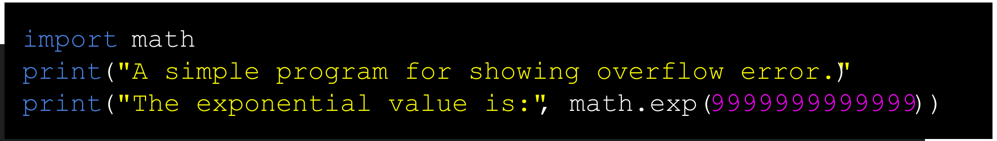
```
aora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora
ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora o
ra ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraor
raora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora ora oraora ora ora or
a ora oraora ora ora ora oraora ora ora ora oraTraceback (most recent call last):
  File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\owaranai.py", line 2, in <module>
    print('ora ora ora ora ora', end = '')
KeyboardInterrupt: Execution interrupted

>>>
```

We will get an `KeyboardInterupt` if the user pressed **Ctrl+C** when the program is running

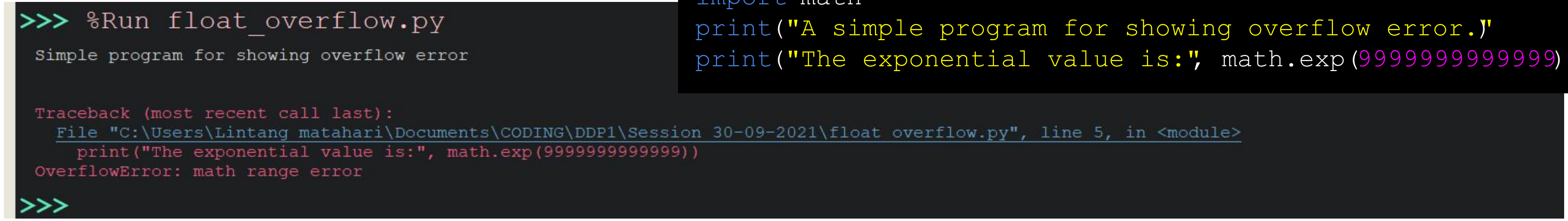


Exception	Explanation
`KeyboardInterrupt`	Raised when user hits Ctrl-C, the interrupt key
`OverflowError`	Raised when a floating-point expression evaluates to a value that is too large
`ZeroDivisionError`	Raised when attempting to divide by 0
`IOError`	Raised when an I/O operation fails for an I/O-related 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

Error Names (3): `OverflowError`

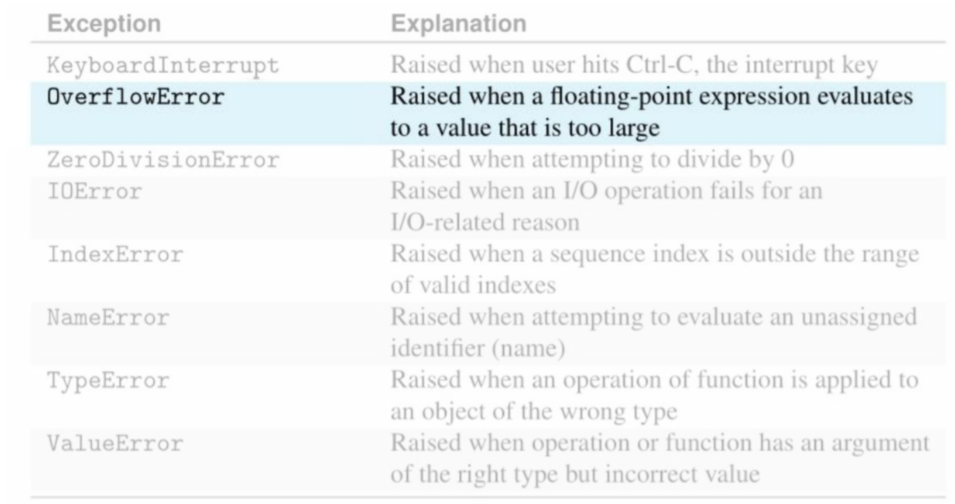


```
import math
print("A simple program for showing overflow error.)"
print("The exponential value is:", math.exp(9999999999999))
```



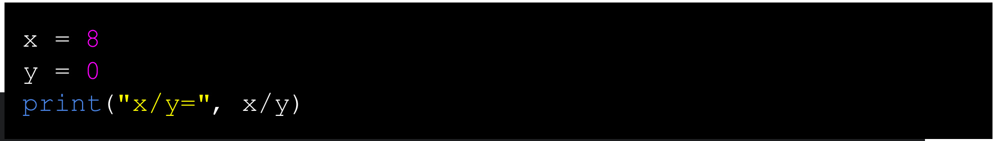
```
>>> %Run float_overflow.py
 Simple program for showing overflow error


 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\float_overflow.py", line 5, in <module>
     print("The exponential value is:", math.exp(9999999999999))
 OverflowError: math range error

>>>
```

We will get an `OverflowError` if the value of a given type **exceeds** the declared limit value

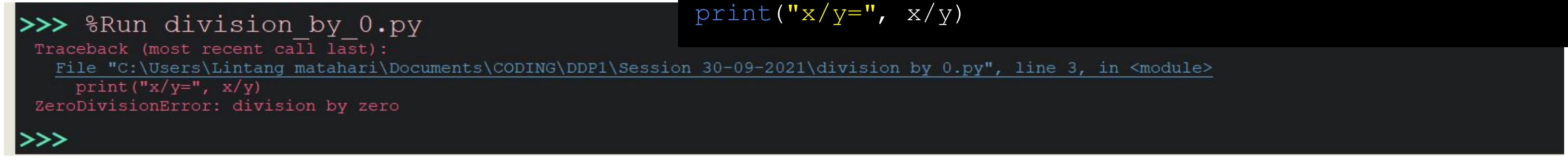


Exception	Explanation
KeyboardInterrupt	Raised when user hits Ctrl-C, the interrupt key
OverflowError	Raised when a floating-point expression evaluates to a value that is too large
ZeroDivisionError	Raised when attempting to divide by 0
IOError	Raised when an I/O operation fails for an I/O-related reason
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

Error Names (4): `ZeroDivisionError`

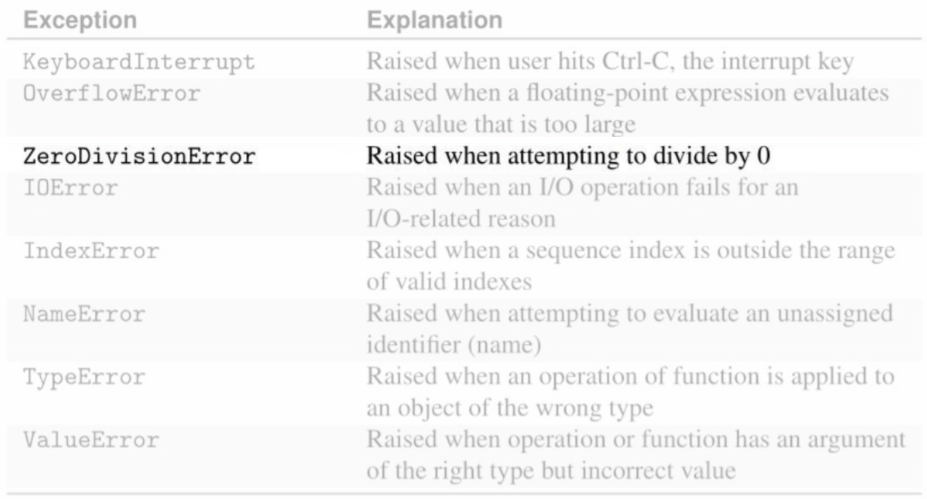


```
x = 8
y = 0
print("x/y=", x/y)
```



```
>>> %Run division_by_0.py
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\division by 0.py", line 3, in <module>
     print("x/y=", x/y)
 ZeroDivisionError: division by zero

>>>
```

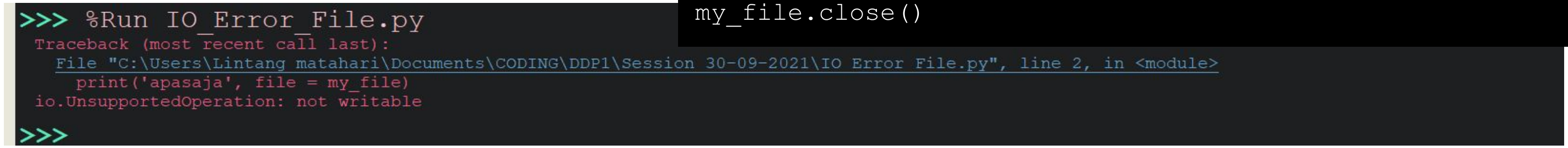
We will get an `ZeroDivisionError` if we attempt to **divide by 0**



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

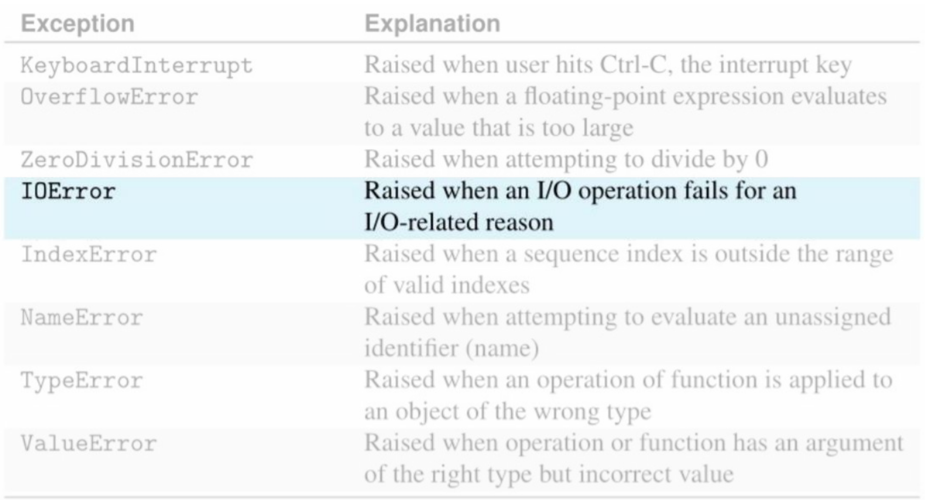
Error Names (5): `IOError` (UnsupportedOperation)

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



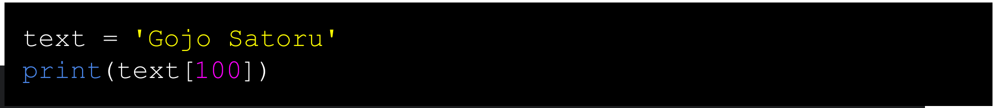
```
>>> %Run IO_Error_File.py
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\IO Error File.py", line 2, in <module>
     print('apasaja', file = my_file)
 io.UnsupportedOperation: not writable

>>>
```

In the above example, we get an `IOError` because we attempt to **write to a file that is not writable**

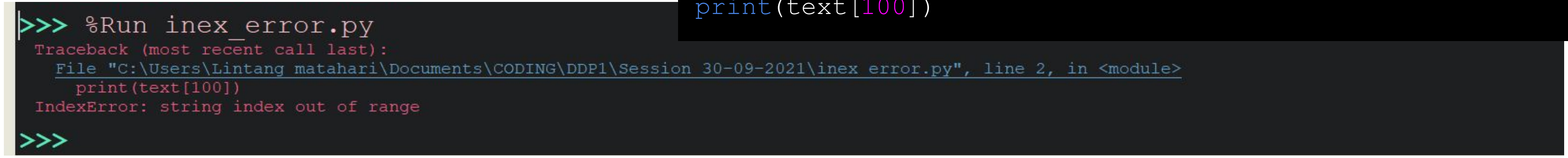


Exception	Explanation
KeyboardInterrupt	Raised when user hits Ctrl-C, the interrupt key
OverflowError	Raised when a floating-point expression evaluates to a value that is too large
ZeroDivisionError	Raised when attempting to divide by 0
IOError	Raised when an I/O operation fails for an I/O-related reason
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

Error Names (5): `IndexError`

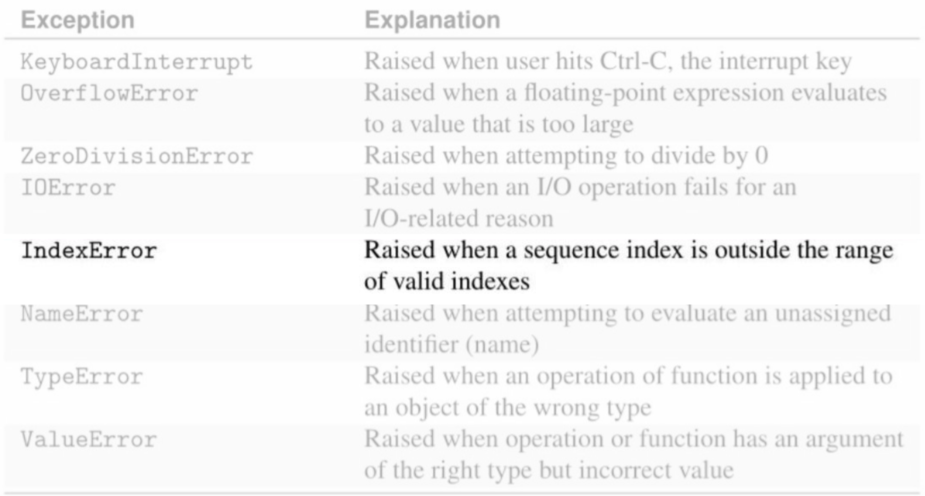


```
text = 'Gojo Satoru'
print(text[100])
```



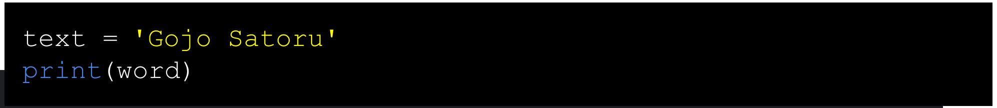
```
>>> %Run inex_error.py
Traceback (most recent call last):
  File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\inex error.py", line 2, in <module>
    print(text[100])
IndexError: string index out of range
>>>
```

In the above example, we get an `IndexError` because we attempt to access an element of a collection using sequence index that is **outside the valid range**

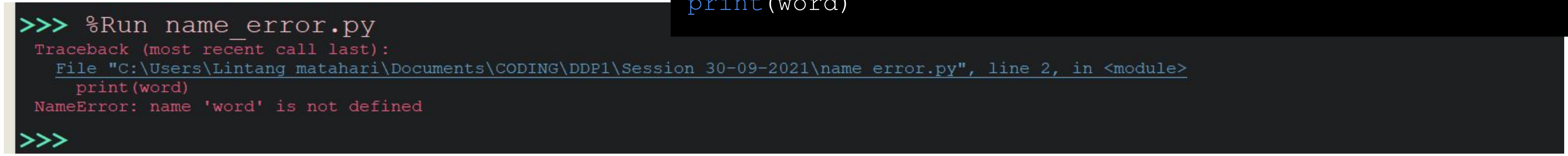


Exception	Explanation
KeyboardInterrupt	Raised when user hits Ctrl-C, the interrupt key
OverflowError	Raised when a floating-point expression evaluates to a value that is too large
ZeroDivisionError	Raised when attempting to divide by 0
IOError	Raised when an I/O operation fails for an I/O-related reason
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

Error Names (6): `NameError`

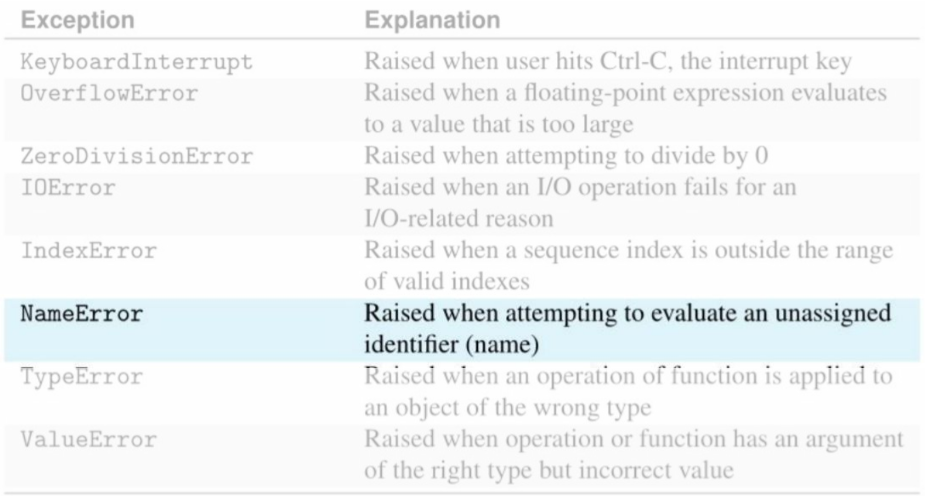


```python
text = 'Gojo Satoru'
print(word)
```



```
>>> %Run name_error.py
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\name_error.py", line 2, in <module>
     print(word)
 NameError: name 'word' is not defined
>>>
```

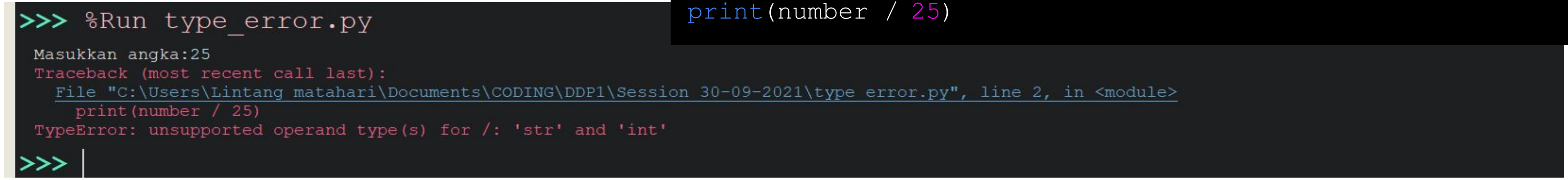
In the above example, we get an `NameError` because we attempt to evaluate an **undefined** variable



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

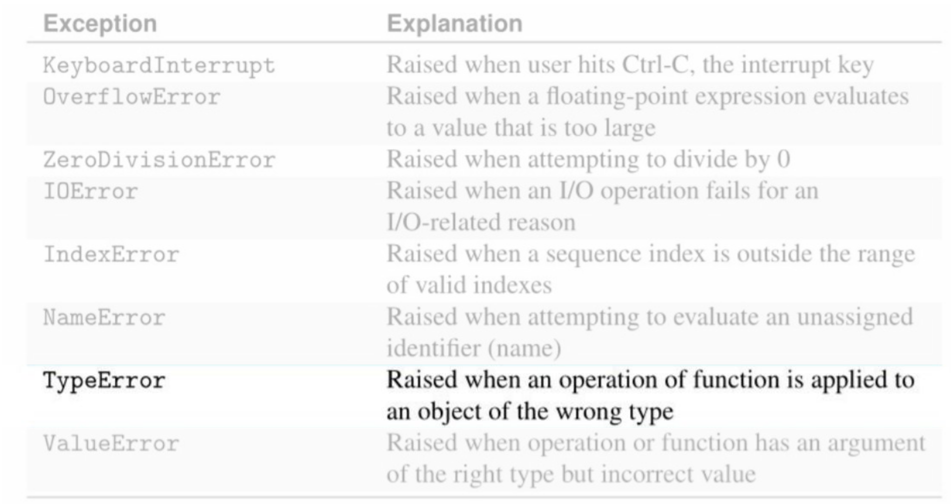
Error Names (7): `TypeError`

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



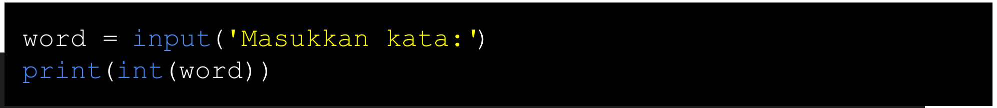
```
>>> %Run type_error.py
 Masukkan angka:25
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\type_error.py", line 2, in <module>
     print(number / 25)
 TypeError: unsupported operand type(s) for /: 'str' and 'int'
>>> 
```

In the above example, we get a `TypeError` because we attempt to use **unsupported operand type** for `str` and `int`

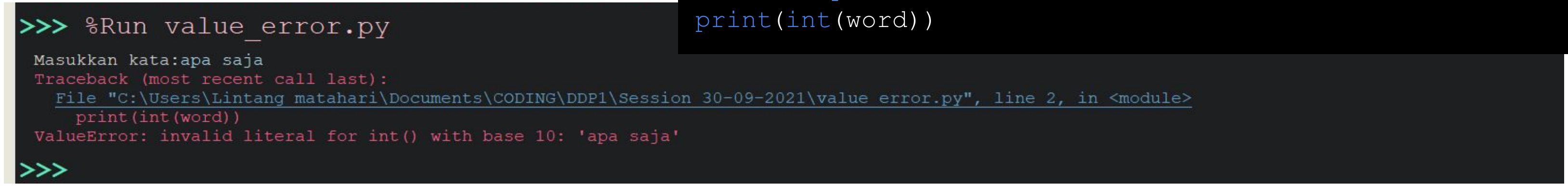


Exception	Explanation
`KeyboardInterrupt`	Raised when user hits Ctrl-C, the interrupt key
`OverflowError`	Raised when a floating-point expression evaluates to a value that is too large
`ZeroDivisionError`	Raised when attempting to divide by 0
`IOError`	Raised when an I/O operation fails for an I/O-related 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

Error Names (8): `ValueError`

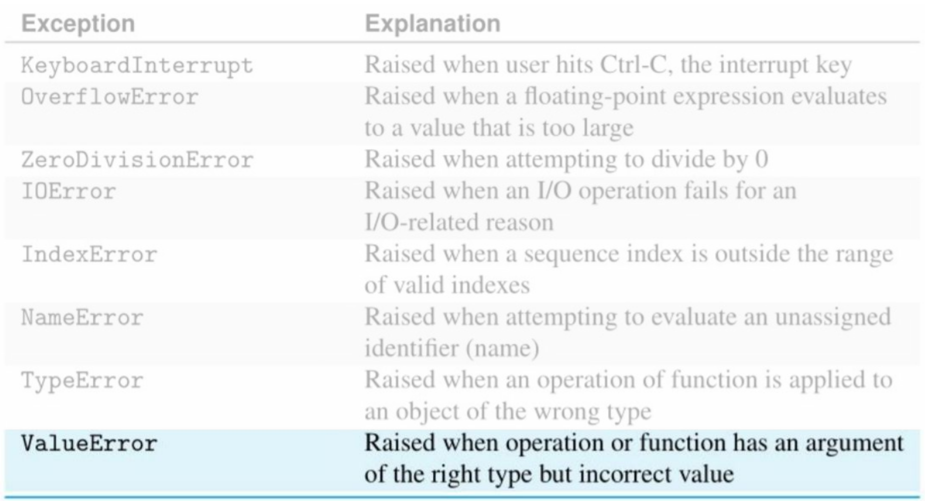


```
word = input('Masukkan kata:')
print(int(word))
```



```
>>> %Run value_error.py

 Masukkan kata:apa saja
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\value error.py", line 2, in <module>
     print(int(word))
 ValueError: invalid literal for int() with base 10: 'apa saja'
>>>
```

In the above example, we get a `ValueError` if we attempt to convert an **unconvertible** string value (e.g., ‘ABC’) to `int`



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

Triggering Question 1

What are the outputs?

a.
```
prins(a)
```

b.
```
a = 5/0
```

c.
```
my_file = open('file.txt','r')
```

d.
```
str1 = 'DDP1'
print(str1[12])
```

Will the program produces an error? What kind of error?

Write the answer in the **comment section**

Dealing with “Exceptional” Situations

> “An *exception* is an event that occurs during the execution of a program that disrupts the normal flow of instructions during the execution of a program.”
> *(iitk.ac.in)*

➔ Most modern languages provide methods to deal with ‘exceptional’ situations

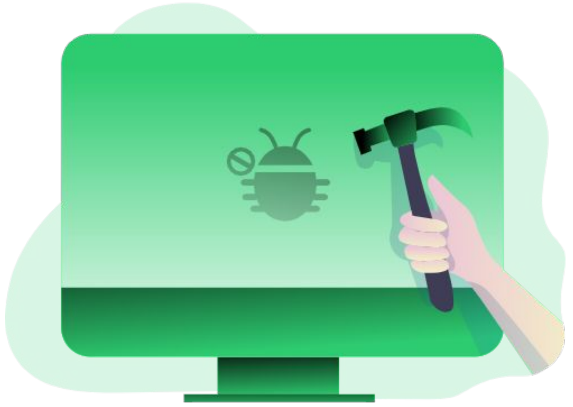
➔ Gives the programmer the option **to keep the user from having the program stop in case an “exceptional situation” occurs**

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)

```python
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")
```



Exception Handling General Form (1)

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

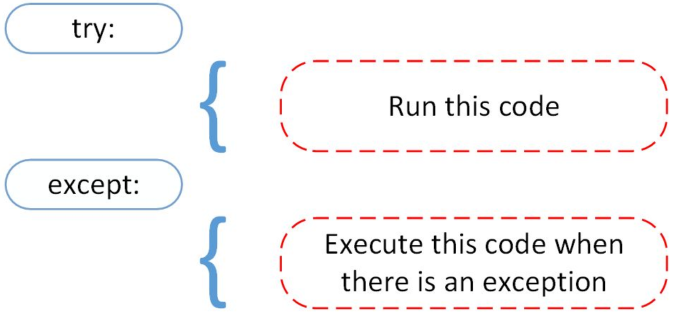
```
a = 6
b = 0
# simple use of try-except block for handling errors
try:
    g = a/b
except ZeroDivisionError:
    print("This is a DIVIDED BY ZERO error")

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

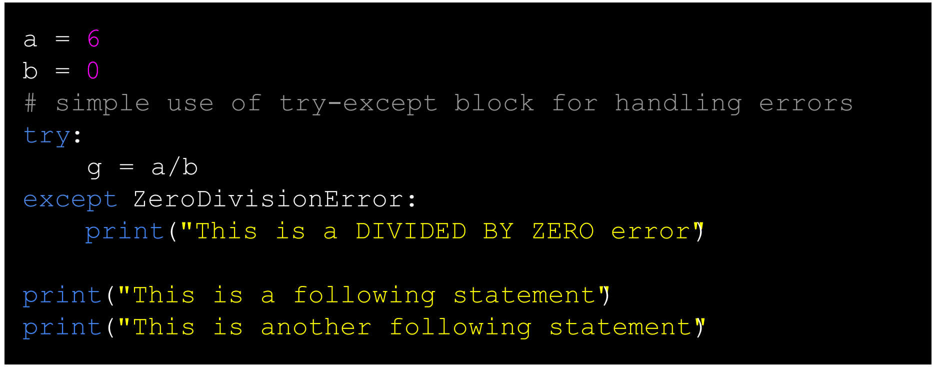
The try suite contains code that we want to **monitor for errors** during its execution.

➔ 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

Exception Handling General Form (2)





```
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")
```

https://realpython.com/python-exceptions/#the-try-and-except-block-handling-exceptions

`except` Suites

- ➔ 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**

```
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` and `except` group

- ➔ If no exception in the `try` suite, skip all the `try`/`except` to the next line of code
- ➔ If an error occurs in a `try` suite, **look for the right exception**
- ➔ **If found, run** that `except` suite and then skip past the `try`/`except` group to the next line of code
- ➔ If no exception handling found, give the error to Python

```python
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")
```

Triggering Question 2

Suppose we want to create a simple program to give the result from x / 25. X is intended to be an integer that is inputted by the user.

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

What will happen if the user does not give a number and write 'DDP1 mantappu jiwa' instead? ^^

Write the answer in the **comment section**

User Input: Potential Unhandled Error

This is a modified version of the previous program.

```python
is_prompt = True
print('Ini adalah program penghitung pembagian dengan 25.)'

while(is_prompt):
    number = int(input('Masukkan angka: '))
    print(number / 25)

    ask = input('Anda ingin melanjutkan? Y/N: ')
    if(ask == 'N'):
        is_prompt = False

print('Program selesai.')
```

- ➔ In general, we have assumed that the input we receive is correct (from a file, from the user).
- ➔ This is almost never true. There is always the chance that the input could be wrong
- ➔ Our programs should be able to handle this.

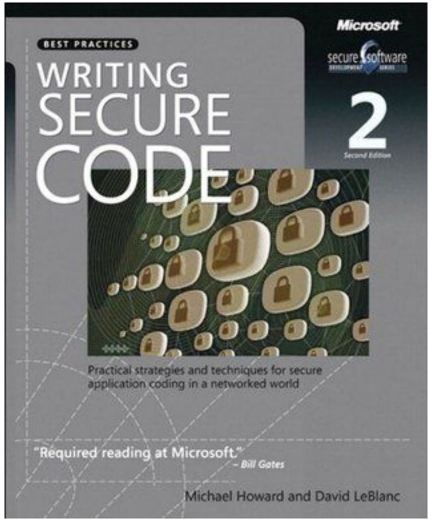
Worse yet, input is evil

In "Writing Secure Code”, by Howard and LeBlanc, there is an interesting quote:

“All input is evil until proven otherwise”

➔ Most security holes in programs are based on assumptions programmers make about input

➔ Secure programs protect themselves from evil input


BEST PRACTICES
WRITING SECURE CODE
Microsoft
secure software
2
Second Edition
Practical strategies and techniques for secure application coding in a networked world
"Required reading at Microsoft." – Bill Gates
Michael Howard and David LeBlanc

Code Example: Handling Value Error from User Input

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

Code Example: Error Handling Using `except ValueError:`

```python
var_a = ["satu", 2, "3"]
for x in var_a:
     try:
          b = int(x)
          print("Berhasil memproses", x)
     except ValueError:
          print("ValueError saat memproses", x)
```

Code Example: Error Handling Using `except:`

```
var_a = ["satu", 2, "3"]
for x in var_a:
     try:
          b = int(x)
          print("Berhasil memproses", x)
     except:
          print("Error saat memproses", x)
```

Pros: - Can catch any kind of errors
Cons: - Unclear error, hard to trace

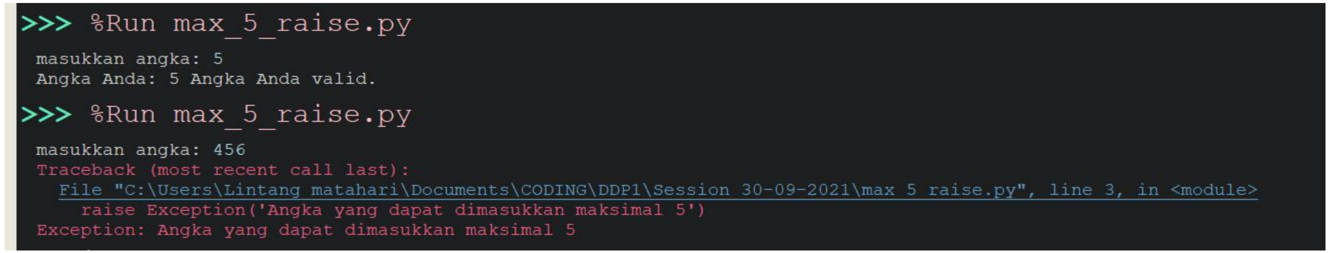
Raising an Exception

We can **forcefully** raise an exception in Python using `raise`

- ➔ We may want to use this if we want to raise an error when certain condition is met.
- ➔ Maybe we have certain rules that must be complied in certain operations

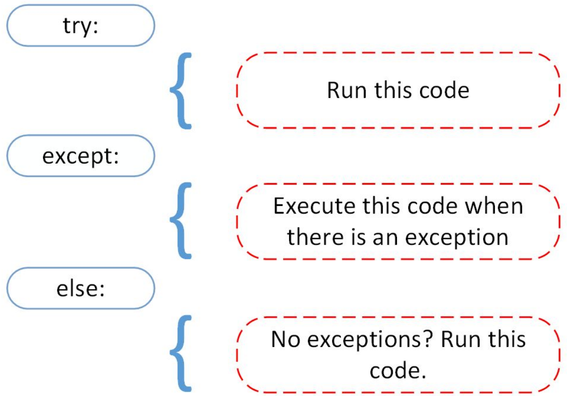
Example:
Suppose we want to limit the user inputted integer to the following program up to 5.

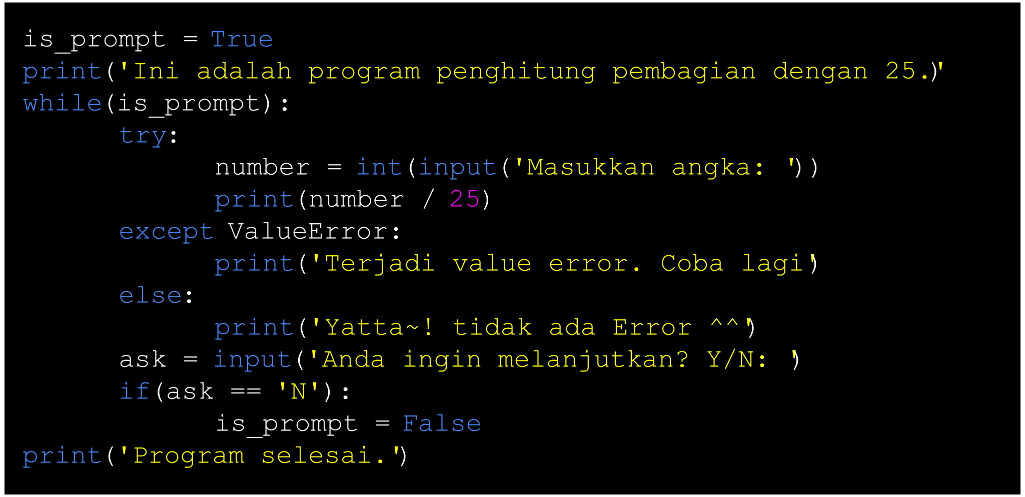
```python
number = int(input('masukkan angka: '))
if number > 5:
      raise Exception('Angka yang dapat dimasukkan maksimal 5')

print('Angka Anda:', number, 'Angka Anda valid.')
```



```
>>> %Run max_5_raise.py

 masukkan angka: 5
 Angka Anda: 5 Angka Anda valid.

>>> %Run max_5_raise.py

 masukkan angka: 456
 Traceback (most recent call last):
   File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 30-09-2021\max_5_raise.py", line 3, in <module>
     raise Exception('Angka yang dapat dimasukkan maksimal 5')
 Exception: Angka yang dapat dimasukkan maksimal 5
```

`try except else` Suites


try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.

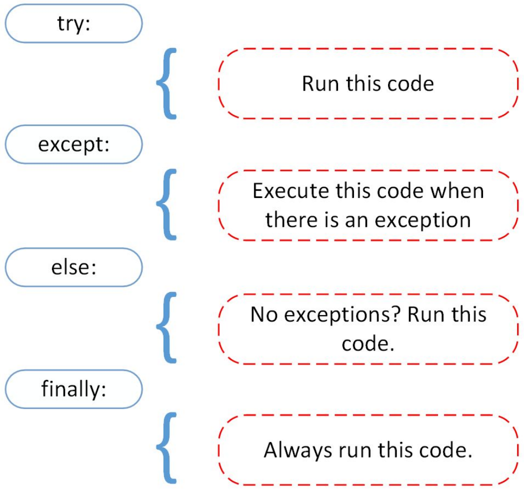


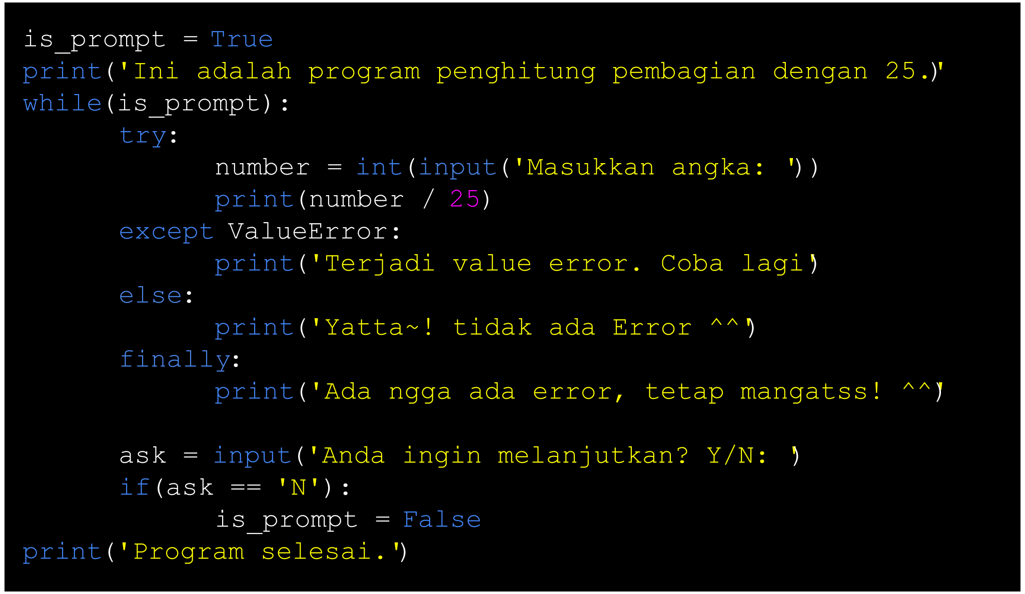
```
is_prompt = True
print('Ini adalah program penghitung pembagian dengan 25.')
while(is_prompt):
        try:
                number = int(input('Masukkan angka: '))
                print(number / 25)
        except ValueError:
                print('Terjadi value error. Coba lagi')
        else:
                print('Yatta~! tidak ada Error ^^')
        ask = input('Anda ingin melanjutkan? Y/N: ')
        if(ask == 'N'):
                is_prompt = False
print('Program selesai.')
```

The `else` suite will only be executed **if no Exception occurs**

https://realpython.com/python-exceptions/#the-else-clause

`try except else finally` Suites


try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.
finally:
Always run this code.



```
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 mangatss! ^^')

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

The `finally` suite will always be executed **even when Exception occurs**

https://realpython.com/python-exceptions/#the-else-clause

Triggering Question 3

What is the output?

Suppose the user inputted ‘25+25’

```
try:
      number = int(input('Masukkan angka:'))
      print(number/100)
except ValueError:
      print('Terjadi Value Error. Gagal memroses.')
except NameError:
      print('Terjadi Name Error. Gagal memroses.')
except TypeError:
      print('Terjadi Type Error. Gagal memroses.')
else:
      print('Proses dijalankan.')
finally:
      print('Program telah berjalan.')
```

Write the answer in the **comment section**

Review Questions

Explain the definition of syntax, runtime, and logic errors.

What is an Exception?

Explain how `try`, `except`, `else`, and `finally` suites are executed.

When do we need to use `try` and `except` Exception handling?
Give one case.

FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

Q&A Session