University of Tripoli
Faculty of Information Technology
Department of Software Engineering

ITSE305 مواضيع مختارة
Python Programming
S2025

Lecture (8): Python Try Except

Python Try Except

- The try block lets you test a block of code for errors.
- The except block lets you handle the error.
- The else block lets you execute code when there is no error.
- The finally block lets you execute code, regardless of the result of the try- and except blocks.

by: Fatima Ben Lashihar

Exception Handling

- When an error occurs, or exception as we call it, Python will normally stop and generate an error message.
- These exceptions can be handled using the try statement that block raises an error, thus the except block will be executed.

```
try:
  print(x)
except:
  print("An exception occurred")
```

```
An exception occurred
```

- Without the try block, the program will crash and raise an error:

```
print(x)
```

```
Traceback (most recent call last):
  File "demo_try_except_error.py", line 3, in <module>
    print(x)
NameError: name 'x' is not defined
```


Many Exceptions

- You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error:

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

```
Variable x is not defined
```

Python Built-in Exceptions

- There are many Python Built-in Exceptions such as:
  - Exception Base class for all exceptions
  - IndentationError Raised when indentation is not correct
  - KeyError Raised when a key does not exist in a dictionary
  - NameError Raised when a variable does not exist
  - SyntaxError Raised when a syntax error occurs
  - ValueError Raised when there is a wrong value in a specified data type
  - ZeroDivisionError Raised when the second operator in a division is zero

Else

- You can use the else keyword to define a block of code to be executed if no errors were raised

```
try:
  print("Hello")
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
```

```
Hello
Nothing went wrong
```

Finally

- The finally block, if specified, will be executed regardless if the try block raises an error or not.

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

```
Something went wrong
The 'try except' is finished
```


Finally

```
try:
  f = open("demofile.txt")
  try:
    f.write("Lorum Ipsum")
  except:
    print("Something went wrong when writing to the
file")
  finally:
    f.close()
except:
  print("Something went wrong when opening the file")
```

```
Something went wrong when writing to the file
```

Raise an exception

- As a Python developer you can choose to throw an exception if a condition occurs.
- To throw (or raise) an exception, use the raise keyword.
- The raise keyword is used to raise an exception.
- You can define what kind of error to raise, and the text to print to the user.

```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

```
Traceback (most recent call last):
  File "demo_ref_keyword_raise.py", line 4, in <module>
    raise Exception("Sorry, no numbers below zero")
Exception: Sorry, no numbers below zero
```


Raise an exception

```
x = "hello"

if not type(x) is int:
  raise TypeError("Only integers are allowed")
```

```
Traceback (most recent call last):
  File "demo_ref_keyword_raise2.py", line 4, in <module>
    raise TypeError("Only integers are allowed")
TypeError: Only integers are allowed
```

Python Try Except - Example(1)

```
try:
    n = 0
    res = 100 / n

except ZeroDivisionError:
    print("You can't divide by zero!")

except ValueError:
    print("Enter a valid number!")

else:
    print("Result is", res)

finally:
    print("Execution complete.")
```

```
You can't divide by zero!
Execution complete.
```


Python Try Except - Example(2)

```
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 2)
divide(3, 0)
```

```
Yeah ! Your answer is : 1
Sorry ! You are dividing by zero
```

Python Try Except - Example(3)

```
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except Exception as e:
       # By this way we can know about the type of error occurring
        print("The error is: ",e)


divide(3, "GFG")
divide(3,0)
```

```
The error is:  unsupported operand type(s) for //: 'int' and 'str'
The error is:  integer division or modulo by zero
```


The END