Programming Fundamentals

Lecture 7 – Python Exception Handling

Iresh Bandara

ROBERT GORDON UNIVERSITY ABERDEEN
TEF Gold
INFORMATICS INSTITUTE OF TECHNOLOGY

Learning Outcomes

- This lecture addresses LO2 and LO4 for the module
- On completion of this lecture, students are expected to explain and apply
  - Try – except – else – finally
  - Raise exception
- Analyse program flows based on exception handling

Agenda

- Exception handling – Introduction
- Various try -except scenarios
- Try – except – else
- Try- except- finally
- Common exception types
- Raising and exception

Exception Handing - Introduction

- Event triggered during the execution time due to the disruption of the normal flow
- Python raises an exception when this happens and represents an erroneous situation
- Exception is a python object or a user defined object
- Developer needs to handle the exception.
- Otherwise program will crash

Exception Handing – try except

- Suspicious code need to be wrapped inside a try block
- When an exception raised, except block will be called
- Try except needs to follow the proper indentation

```
try:
    #suspicious code
except:
    #handle the exception
#code
```

Execute if the code inside raise an exception only

Reach here after handling the exception. If there was no exception handling and the code inside try raise an exception program could have crashed

Exception Handing – try except and else

```
try:
    #suspicious code
except exception_type1 :
    #handle the exception1
except exception_type2 :
    #handle the exception2
else:
    #if except was not called
#code after handling
```

Can have multiple exception blocks. Depends on the exception raised, relevant block will be called.

Else will be called when no exception was raised

Code reach here after handling exception

Few Exception types

Exception	**Reason for throwing**
ArithmeticError	all errors that occur for numeric calculation.
ZeroDivisionError	Errors when division or modulo by zero takes place
KeyboardInterrupt	when the user interrupts program execution, usually by pressing Ctrl+c.
AssertionError	in case of failure of the Assert statement.
SystemExit	when sys.exit() function was called.
OverflowError	when a calculation exceeds maximum limit for a numeric type
IOError	input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.
ValueError	When passed invalid values to a built-in function

More try except variations

```
try:
  #suspicious code
except:
  #handling for all types of
exceptions/ common block
```

```
try:
  #suspicious code
except(except_type1,except_type2….):
  #handling mentioned types/common
```

```
try:
  #suspicious code
except exception_type as Error:
  print(Error)
```

```
try:
  #suspicious code
except exception_type
   pass #catch it, but no action taken
```

Exception Handing – try except and finally

```
try:
    #suspicious code
except exception_type1 :
    #handle the exception1
finally:
    #Always execute
```

Can have multiple exception blocks. Depends on the exception raised, relevant block will be called.

Does not matter the exception was raised or not, this block will always execute

Raising an exception

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

Raise a TypeError if the type of x is not an integer

```
try:
     func(“test”)
except TypeError as e:
     print(e)
     #print only integers are allowed
```

Call the above function

TypeError is captured here

Challenge : Custom exceptions

- So far we discussed about predefined exception types
- It is possible the developer to create custom exceptions and raises
- Can create custom exception classes
- **Try to create a custom exception for a certain condition, trigger the exception and catch it**

Summary

- Try blocks are used to wrapped suspicious code snippets
- If the code inside try raises an exception except block will be called
- If the except is not called, Else will be called
- Finally block is triggered all the time
- Common exception types were discussed
- “Raise” was used to raise an exception under a certain circumstance
- Custom exceptions can be defined using Python clases