BCA – 502: Python Programming

Rahul Kumar Singh

In today's Class we have discussed on Exception Handling in Python.

What is Exception?

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

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. Since the try block raises an error, the except block will be executed. Without the try block, the program will crash and raise an error.

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a **try**: block. After the **try**: block, include an **except**: statement, followed by a block of code which handles the problem as elegantly as possible.

Syntax:-

Here is simple syntax of try....except...else blocks −

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Here are few important points about the above-mentioned syntax −

- A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.

- You can also provide a generic except clause, which handles any exception.
- After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.
- The else-block is a good place for code that does not need the try: block's protection.

Example:- This example opens a file, writes content in the, file and comes out gracefully because there is no problem at all –

```
#!/usr/bin/python
try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
   fh.close()
```

Output:-

```
Written content in the file successfully
```

Example:- This example tries to open a file where you do not have write permission, so it raises an exception –

```
#!/usr/bin/python
try:
   fh = open("testfile", "r")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
```

Output:-

```
Error: can't find file or read data
```

The except Clause with No Exceptions:-

You can also use the except statement with no exceptions defined as follows –

```
try:
   You do your operations here;
   ......................
except:
   If there is any exception, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

This kind of a **try-except** statement catches all the exceptions that occur. Using this kind of try-except statement is not considered a good programming practice though, because it catches all exceptions but does not make the programmer identify the root cause of the problem that may occur.

The except Clause with Multiple Exceptions:-

You can also use the same except statement to handle multiple exceptions as follows –

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
   If there is any exception from the given exception list,
   then execute this block.
   ......................
```

```
else:
   If there is no exception then execute this block.
```

The try-finally Clause:-

You can use a finally: block along with a try: block. The finally block is a place to put any code that must execute, whether the try-block raised an exception or not. You cannot use else clause as well along with a finally clause.

The syntax of the try-finally statement is this –

```
try:
   You do your operations here;
   ......................
   Due to any exception, this may be skipped.
finally:
   This would always be executed.
   ......................
```

Example:-

```
#!/usr/bin/python
try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
finally:
   print "Error: can\'t find file or read data"
```

Output:-

If you do not have permission to open the file in writing mode, then this will produce the following result −

```
Error: can't find file or read data
```

Example:−2

```
#!/usr/bin/python
try:
   fh = open("testfile", "w")
   try:
      fh.write("This is my test file for exception handling!!")
   finally:
      print "Going to close the file"
      fh.close()
except IOError:
   print "Error: can\'t find file or read data"
```

Output:-

```
Going to close the file
```

When an exception is thrown in the try block, the execution immediately passes to the finally block. After all the statements in the finally block are executed, the exception is raised again and is handled in the except statements if present in the next higher layer of the try-except statement.