**EXCEPTION HANDLING**

Exception can be said to be any abnormal condition in a program resulting to the disruption in the flow of the program.

Whenever an exception occurs the program halts the execution and thus further code is not executed. Thus exception is that error which python script is unable to tackle with.

Exception in a code can also be handled. In case it is not handled, then the code is not executed further and hence execution stops when exception occurs.

**Hierarchy Of Exception:**

1. ZeroDivisionError: Occurs when a number is divided by zero.
2. NameError: It occurs when a name is not found. It may be local or global.
3. IndentationError: If incorrect indentation is given.
4. IOError: It occurs when Input Output operation fails.
5. EOFError: It occurs when end of file is reached and yet operations are being performed

etc..

**Exception Handling:**

The suspicious code can be handled by using the try block. Enclose the code which raises an exception inside the try block. The try block is followed except statement. It is then further followed by statements which are executed during exception and in case if exception does not occur.

**Syntax:**

1. try:
2. malicious code
3. except Exception1:
4. execute code
5. except Exception2:
6. execute code
7. ....
8. ....
9. except ExceptionN:
10. execute code
11. else:
12. In case of no exception, execute the else block code.

**eg:**

1. try:
2. a=10/0
3. print a
4. except ArithmeticError:
5. print "This statement is raising an exception"
6. else:
7. print "Welcome"

**Output:**

1. >>>
2. This statement is raising an exception
3. >>>

**Explanation:**

1. The malicious code (code having exception) is enclosed in the try block.
2. Try block is followed by except statement. There can be multiple except statement with a single try block.
3. Except statement specifies the exception which occurred. In case that exception is occurred, the corresponding statement will be executed.
4. At the last you can provide else statement. It is executed when no exception is occurred.

**Except with no Exception:**

Except statement can also be used without specifying Exception.

**Syntax:**

1. try:
2. code
3. except:
4. code to be executed in case exception occurs.
5. else:
6. code to be executed in case exception does not occur.

**eg:**

1. try:
2. a=10/0;
3. except:
4. print "Arithmetic Exception"
5. else:
6. print "Successfully Done"

**Output:**

1. >>>
2. Arithmetic Exception
3. >>>

**Declaring Multiple Exception**

Multiple Exceptions can be declared using the same except statement:

**Syntax:**

1. try:
2. code
3. except Exception1,Exception2,Exception3,..,ExceptionN
4. execute this code in case any Exception of these occur.
5. else:
6. execute code in case no exception occurred.

**eg:**

1. try:
2. a=10/0;
3. except ArithmeticError,StandardError:
4. print "Arithmetic Exception"
5. else:
6. print "Successfully Done"

**Output:**

1. >>>
2. Arithmetic Exception
3. >>>

**Finally Block:**

**In case if there is any code which the user want to be executed, whether exception occurs or not then that code can be placed inside the finally block. Finally block will always be executed irrespective of the exception.**

**Syntax:**

1. try:
2. Code
3. finally:
4. code which is must to be executed.

**eg:**

1. try:
2. a=10/0;
3. print "Exception occurred"
4. finally:
5. print "Code to be executed"

**Output:**

1. >>>
2. Code to be executed
3. Traceback (most recent call last):
4. File "C:/Python27/noexception.py", line 2, in <module>
5. a=10/0;
6. ZeroDivisionError: integer division or modulo by zero
7. >>>

In the above example finally block is executed. Since exception is not handled therefore exception occurred and execution is stopped.

**Raise an Exception:**

You can explicitly throw an exception in Python using ?raise? statement. raise will cause an exception to occur and thus execution control will stop in case it is not handled.

**Syntax:**

1. raise Exception\_class,<value>

**eg:**

1. try:
2. a=10
3. print a
4. raise NameError("Hello")
5. except NameError as e:
6. print "An exception occurred"
7. print e

**Output:**

1. >>>
2. 10
3. An exception occurred
4. Hello
5. >>>

**Explanation:**

i) To raise an exception, raise statement is used. It is followed by exception class name.

ii) Exception can be provided with a value that can be given in the parenthesis. (here, Hello)

iii) To access the value "as" keyword is used. "e" is used as a reference variable which stores the value of the exception.

**Custom Exception:**

**Refer to this section after visiting Class and Object section:**

Creating your own Exception class or User Defined Exceptions are known as Custom Exception.

**eg:**

1. class ErrorInCode(Exception):
2. def \_\_init\_\_(self, data):
3. self.data = data
4. def \_\_str\_\_(self):
5. return repr(self.data)
7. try:
8. raise ErrorInCode(2000)
9. except ErrorInCode as ae:
10. print "Received error:", ae.data

**Output:**

1. >>>
2. Received error : 2000
3. >>>