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:**

**try**:

    malicious code

**except** Exception1:

    execute code

**except** Exception2:

    execute code

....

....

**except** ExceptionN:

    execute code

**else**:

    In case of no exception, execute the **else** block code.

**eg:**

**try**:

    a=10/0

**print** a

**except** ArithmeticError:

**print** "This statement is raising an exception"

**else**:

**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:**

**try**:

        code

**Except** :

        code to be executed **in** case exception occurs.

**else**:

        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:**

**try**:

    a=10/0;

**except** ArithmeticError,StandardError:

**print** "Arithmetic Exception"

**else**:

**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:**

**try**:

    Code

**finally**:

    code which **is** must to be executed.

**eg:**

**try**:

    a=10/0;

**print** "Exception occurred"

**finally**:

**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.