Computational Thinking and Programming - 2

Exception Handling

XII

Exception Handling

Exception handling in Python involves the use of try and except block.

- The code that may result an exception is written in try block
- The code for handling the exception when the exception is raised is written in except block

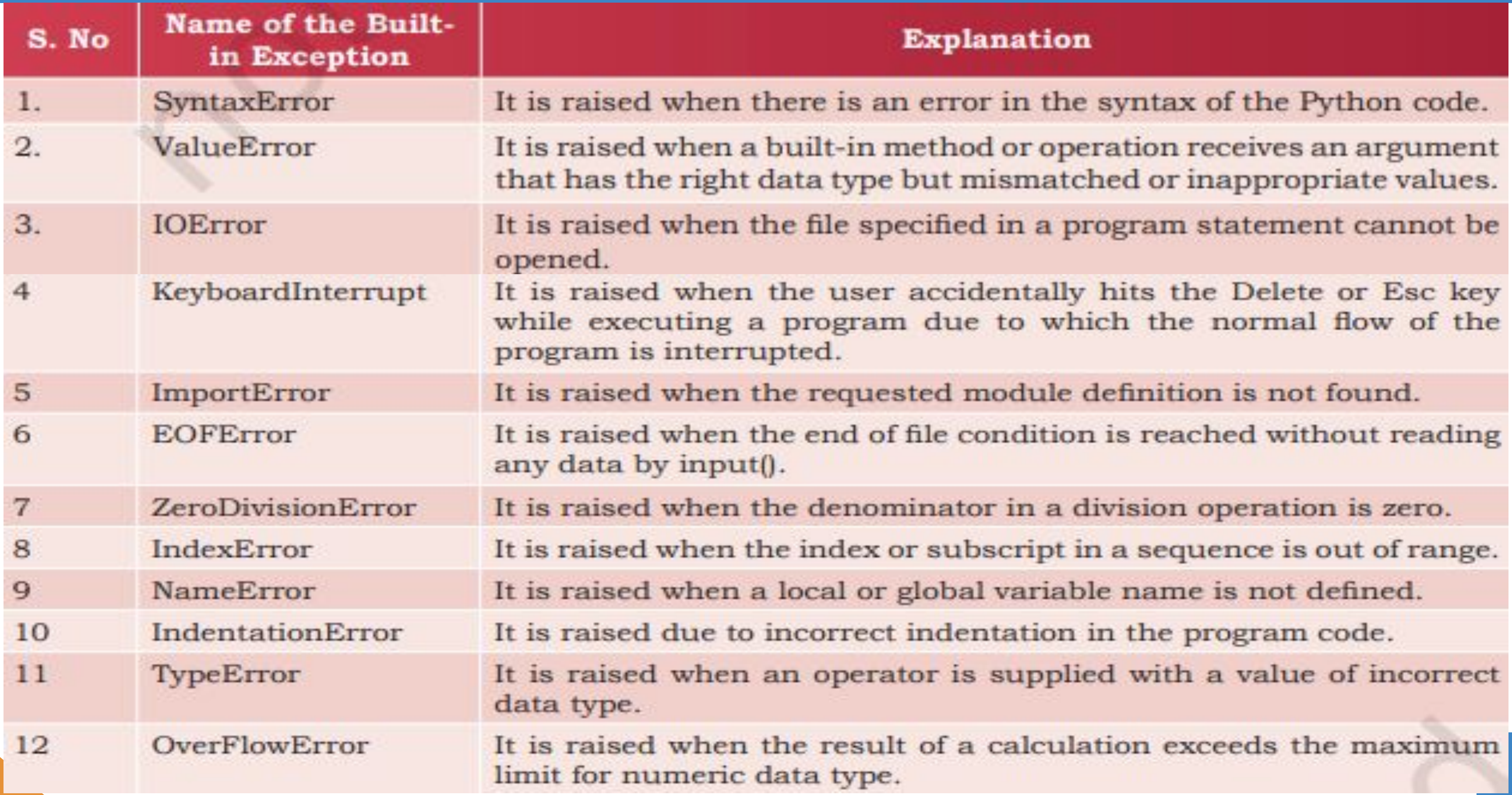
Syntax :
```
try :
        statements
except:
        statements
```

The above syntax means that a try statement must have a try clause and at least one except clause.

Exception Handling

```
Example :

try:

    n1 = int(input("Enter number 1- "))

    n2 = int(input("Enter number 2 -"))

    res = n1 / n2

    print("Result -",res)

except:

    print("Denominator must be non zero ")
```

Built In Exceptions



S. No	Name of the Built-in Exception	Explanation
1.	SyntaxError	It is raised when there is an error in the syntax of the Python code.
2.	ValueError	It is raised when a built-in method or operation receives an argument that has the right data type but mismatched or inappropriate values.
3.	IOError	It is raised when the file specified in a program statement cannot be opened.
4	KeyboardInterrupt	It is raised when the user accidentally hits the Delete or Esc key while executing a program due to which the normal flow of the program is interrupted.
5	ImportError	It is raised when the requested module definition is not found.
6	EOFError	It is raised when the end of file condition is reached without reading any data by input().
7	ZeroDivisionError	It is raised when the denominator in a division operation is zero.
8	IndexError	It is raised when the index or subscript in a sequence is out of range.
9	NameError	It is raised when a local or global variable name is not defined.
10	IndentationError	It is raised due to incorrect indentation in the program code.
11	TypeError	It is raised when an operator is supplied with a value of incorrect data type.
12	OverFlowError	It is raised when the result of a calculation exceeds the maximum limit for numeric data type.

finally block

The try statement in Python can also have an optional finally clause. The statements inside the finally block are always executed regardless of whether an exception has occurred in the try block or not. It is a common practice to use finally clause while working with files to ensure that the file object is closed. If used, finally should always be placed at the end of try clause, after all except blocks and the else block.

Syntax :

```
try:
        Statements
except:
        Statements
finally :
        statments
```

Exception Handling

```
print ("Handling exception using try...except...finally")
try:
        numerator=50
        denom=int(input("Enter the denominator: "))
        quotient=(numerator/denom)
        print ("Division performed successfully")
except ZeroDivisionError:
        print ("Denominator as ZERO is not allowed")
except ValueError:
        print ("Only INTEGERS should be entered")
finally:
        print ("OVER AND OUT")
```

```
Handling exception using try...except...else...finally
Enter the denominator: 4
Division performed successfully
OVER AND OUT
```

```
Handling exception using try...except...else...finally
Enter the denominator: p
Only INTEGERS should be entered
OVER AND OUT
```