# Chapter 9

# *Exceptions*

# Chapter 9

# Exceptions

In This Chapter:

1. Exceptions
2. Incorrect Regex

## 9.1 Exceptions

Errors detected during execution are called *exceptions*.

https://docs.python.org/2/library/exceptions.html

***ZeroDivisionError***

This error is raised when the second argument of a division or modulo operation is zero.

```
>>> a = '1'
>>> b = '0'
>>> print int(a) / int(b)
>>> ZeroDivisionError: integer division or modulo by zero
```

***ValueError***

This error is raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.

```
>>> a = '1'
>>> b = '#'
>>> print int(a) / int(b)
>>> ValueError: invalid literal for int() with base 10: '#'
```

**Handling Exceptions**

The statements *try* and *except* can be used to handle selected exceptions. A *try* statement may have more than one except clause to specify handlers for different exceptions.

```
try:
   print 1/0
except ZeroDivisionError as e:
print "Error Code:",e
```

**Output:** Error Code: integer division or modulo by zero

You are given two values a and b.
Perform integer division and print a/b.

**Input Format**

The first line contains T, the number of test cases.
The next T lines each contain the space separated values of a and b.

**Constraints**

$0 < T < 10$

**Output Format**

Print the value of a/b.
In case of *ZeroDivisionError* or *ValueError*, print the error code.

**Sample Input**

```
3
1 0
2 $
3 1
```

**Sample Output**

```
Error Code: integer division or modulo by zero
Error Code: invalid literal for int() with base 10: '$'
3
```

**Code**

```
T = int(raw_input())
for _ in range(T):
    a, b = raw_input().split()
    try:
        print int(a) / int (b)
    except ZeroDivisionError as z:
        print "Error Code:",z
    except ValueError as v:
        print "Error Code:",v
```

Exception hierarchy

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StandardError
      |    +-- BufferError
      |    +-- ArithmeticError
      |    |    +-- FloatingPointError
      |    |    +-- OverflowError
      |    |    +-- ZeroDivisionError
      |    +-- AssertionError
      |    +-- AttributeError
      |    +-- EnvironmentError
      |    |    +-- IOError
      |    |    +-- OSError
      |    |         +-- WindowsError (Windows)
      |    |         +-- VMSError (VMS)
      |    +-- EOFError
      |    +-- ImportError
      |    +-- LookupError
      |    |    +-- IndexError
      |    |    +-- KeyError
      |    +-- MemoryError
      |    +-- NameError
      |    |    +-- UnboundLocalError
      |    +-- ReferenceError
      |    +-- RuntimeError
      |    |    +-- NotImplementedError
      |    +-- SyntaxError
      |    |    +-- IndentationError
      |    +-- SystemError
      |    +-- TypeError
      |    +-- ValueError
      |         +-- UnicodeError
      |              +-- UnicodeDecodeError
      |              +-- UnicodeEncodeError
      |              +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
```

## 9.2 Incorrect Regex

Given a string S, find out whether S is a valid **regex** or not.

**Input Format**

The first line contains integer T, the number of test cases.
The next T lines contains the string S.

**Constraints**

$0 < T < 100$

**Output Format**

Print "True" or "False" for each test case without quotes.

**Sample Input**

```
2
.*\+
.*+
```

**Sample Output**

```
True
False
```

**Explanation**

**.*\+** : Valid regex.

**.*+**: Has the error multiple repeat. Hence, it is invalid.

**Code**

```
import re

T = int(raw_input())

for _ in range(T):
    s = raw_input()
    try:
        prog = re.compile(s)
        print "True"
    except Exception as e:
        print "False"
```