UnitTest Framework - Exceptions Test

Python testing framework provides the following assertion methods to check that exceptions are raised.

assertRaises(exception, callable, *args, **kwds)

Test that an exception (first argument) is raised when a function is called with any positional or keyword arguments. The test passes if the expected exception is raised, is an error if another exception is raised, or fails if no exception is raised. To catch any of a group of exceptions, a tuple containing the exception classes may be passed as exception.

In the example below, a test function is defined to check whether ZeroDivisionError is raised.

```
import unittest

def div(a,b):
   return a/b
class raiseTest(unittest.TestCase):
   def testraise(self):
      self.assertRaises(ZeroDivisionError, div, 1,0)

if __name__ == '__main__':
   unittest.main()
```

The testraise() function uses assertRaises() function to see if division by zero occurs when div() function is called. The above code will raise an exception. But changes arguments to div() function as follows −

```
self.assertRaises(ZeroDivisionError, div, 1,1)
```

When a code is run with these changes, the test fails as ZeroDivisionError doesn't occur.

```
F
======================================================================
FAIL: testraise (__main__.raiseTest)
----------------------------------------------------------------------
Traceback (most recent call last):
   File "raisetest.py", line 7, in testraise
```

```
      self.assertRaises(ZeroDivisionError, div, 1,1)
AssertionError: ZeroDivisionError not raised

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (failures = 1)
```

assertRaisesRegexp(exception, regexp, callable, *args, **kwds)

Tests that *regexp* matches on the string representation of the raised exception. regexp may be a regular expression object or a string containing a regular expression suitable for use by re.search().

The following example shows how assertRaisesRegexp() is used −

```
import unittest
import re

class raiseTest(unittest.TestCase):
   def testraiseRegex(self):
      self.assertRaisesRegexp(TypeError, "invalid", reg,"Point","TutorialsPoint")

if __name__ == '__main__':
   unittest.main()
```

Here, testraseRegex() test doesn't fail as first argument. "Point" is found in the second argument string.

```
======================================================================
FAIL: testraiseRegex (__main__.raiseTest)
----------------------------------------------------------------------
Traceback (most recent call last):
   File "C:/Python27/raiseTest.py", line 11, in testraiseRegex
      self.assertRaisesRegexp(TypeError, "invalid", reg,"Point","TutorialsPoint")
AssertionError: TypeError not raised
----------------------------------------------------------------------
```

However, the change is as shown below −

```
self.assertRaisesRegexp(TypeError, "invalid", reg,123,"TutorialsPoint")
```

TypeError exception will be thrown. Hence, the following result will be displayed −

```
======================================================================
FAIL: testraiseRegex (__main__.raiseTest)
----------------------------------------------------------------------
Traceback (most recent call last):
   File "raisetest.py", line 11, in testraiseRegex
      self.assertRaisesRegexp(TypeError, "invalid", reg,123,"TutorialsPoint")
AssertionError: "invalid" does not match
   "first argument must be string or compiled pattern"
----------------------------------------------------------------------
```