```
29
30  import unittest
31
32
33  """ def powerOf(x, y) -- return x to the power of y."""
34  class Test(unittest.TestCase):
35
36      """ The following is testing empty input e.g. either x or y, or
37      both x and y are empty, the result should not be defined."""
38      def testEmptyInput(self):
39          self.assertEquals(powerOf("", 2), None)
40          self.assertEquals(powerOf(10, ""), None)
41          self.assertEquals(powerOf("", ""), None)
42
43
44      """ The following is testing bad input e.g. input strings instead of numbers,
45          the result should be not defined. """
46      def testBadInput(self):
47          self.assertEquals(powerOf("hello", 2), None)
48          self.assertEquals(powerOf(10, "a"), None)
49          self.assertEquals(powerOf("hello", "bye"), None)
50
51
52      """ The following is testing integers only: both a and b are integers
53      1st -- test 2 to the power of 3 is 8
54      2nd -- test 10 to the power of -2 is 0.01
55      3rd -- test -10 to the power of 2 is 100
56      4th -- test -10 to the power of -2 is 0.01. """
57      def testIntegersOnly(self):
58          self.assertEqual(powerOf(2, 3), 8)
59          self.assertEquals(powerOf(10, -2), 0.01)
60          self.assertEquals(powerOf(-10, 2), 100)
61          self.assertEquals(powerOf(-10, -2), 0.01)
62
```

```
64      """ The following is testing the cases of 0s. """
65      def testZero(self):
66
67          """ test a positive number (including integer and float) to the power of 0 is 1. """
68          self.assertEquals(powerOf(10, 0), 1)
69          self.assertEquals(powerOf(3234.23432, 0), 1)
70
71          """ testing a negative number (including integer and float) to the power of 0 is 1. """
72          self.assertEquals(powerOf(-8, 0), 1)
73          self.assertEquals(powerOf(-33.234, 0), 1)
74
75          """ testing 0 to the power of a positive number (including integer and float) is 0. """
76          self.assertEqual(powerOf(0, 123), 0)
77          self.assertEquals(powerOf(0, 99.999), 0)
78
79          """ testing 0 to the power of 0 is 1. """
80          self.assertEquals(powerOf(0, 0), 1)
81
82          """ testing 0 to the power of a negative number (including integer and float) is not defined,
83          as a number cannot be divided by 0. """
84          self.assertEquals(powerOf(0, -3), None)
85          self.assertEquals(powerOf(0, -2.45), None)
86
```

```
    """ The following is testing float numbers:
    either a or b, or both a and b are floats. """
    def testFloat(self):

        """ testing a positive integer to the power of (positive and negative) float. """
        self.assertEquals(powerOf(100, 0.5), 10)
        self.assertEquals(powerOf(4, -0.5), 0.5)
        self.assertEquals(round(powerOf(2, 1.5), 3), 2.828)
        self.assertEquals(round(powerOf(2, -1.5), 3), 0.354)


        """ testing if a negative integer to the power of (positive and negative) float is not defined
        as a number cannot be divided by 0"""
        self.assertEquals(powerOf(-100, 0.5), None)
        self.assertEquals(powerOf(-100, -0.5), None)
        self.assertEquals(powerOf(-2, 1.5), None)
        self.assertEquals(powerOf(-2, -1.5), None)


        """ testing a positive float to the power of (positive and negative) float"""
        self.assertEquals(round(powerOf(1.5, -1.5), 3), 0.544)
        self.assertEquals(round(powerOf(1.5, 1.5), 3), 1.837)
        self.assertEquals(round(powerOf(4.8, 0.5), 3), 2.191)
        self.assertEquals(round(powerOf(1.5, -0.5), 3), 0.816)


        """ testing a negative float to the power of (positive and negative) float is not defined"""
        self.assertEquals(powerOf(-4.8, 0.5), None)
        self.assertEquals(powerOf(-1.5, -0.5), None)
        self.assertEquals(powerOf(-2.5, 1.5), None)
        self.assertEquals(powerOf(-1.5, -2.5), None)
```

```
    """ The following is testing a negative number to the power of a (positive and negative) fraction
    1st -- if the numerator of the fractions is an even number, result is defined
    2nd -- if the numerator of the fractions is an odd number, result is not defined.
    3rd -- if the denominator of the fractions is 0, result is not defined. """
    def testFraction(self):
        self.assertEquals(round(powerOf(-2, 2/3), 3), 1.587)
        self.assertEquals(powerOf(-2, 3/5), None)
        self.assertEquals(powerOf(-2, 5/0), None)


if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```