**Assignment No. 5- Creation and Use of Testing Tool**

**Student Name :** Ankit Kailas Nehul **MIS:** 111903018

**Student Name :** Ghanasham Rajaram Salunkhe **MIS:** 111903033

**Course Teacher :** Rohini Sarode Mam

**Subject:** Software Engineering Part 1

**TOPIC: Calculator and Its testing tools**

Files: 1) Calculator.py 2) Test.py

**Functions in Calculator.py:**

* ADD
* *Substract*
* *Multiply*
* *Divide*
* *Power*
* *Root*
* *square*
* Interger division
* factorial

**Functions in Test.py:**

* Test\_add
* Test\_substract
* Test\_multiply
* Test\_divide
* *Test\_Power*
* *Test\_square*
* *Test\_Root*
* Test\_Interger division
* Test\_factorial

**CODE FOR calculator.py**

class Calculator():

    def **add**(self,x, y):

        """Add Function"""

        return x + y

    def **subtract**(self,x, y):

        """Subtract Function"""

        return x - y

    def **multiply**(self,x, y):

        """Multiply Function"""

        return x \* y

    def **divide**(self,x, y):

        """Divide Function"""

        if y == 0:

            raise ValueError('Can not divide by zero!')

        return x / y

    def **integer\_divide**(self,x,y):

        """Interger Division Function"""

        if y == 0:

            raise ValueError('Can not divide by zero!')

        return x //y

    def **power**(self, x,y):

        """Power of number Function"""

        return x\*\*y

    def **root**(self, x):

        """square root Function"""

        if x<0:

            raise ValueError('Cant find square root of negative number')

        return x\*\*(0.5)

    def **square**(self,x):

        """square Function"""

        return x\*\*2

    def **factorial**(self,x):

        """Function to find factorial"""

        fact=1

        if(x==0 or x==1):

            return 1

        else:

            for i in range (x,1,-1):

                fact=fact\*i

            return fact

**CODE FOR test.py**

# CODE: Test.py

import unittest

from calculator import \*

def **setUpModule**():

**print**('Start the testing module')

def **tearDownModule**():

**print**('End the testing module')

class TestCalculator(unittest.TestCase):

   # Create an instance of the calculator that can be used in all tests

**@**classmethod

   def **setUpClass**(self):

**print**('Set up class')

      self.calc = Calculator()

   # Write test methods for subtract, multiply, and divide

   def **test\_add**(self):

      self.**assertEqual**(self.calc.**add**(2, 7), 9, msg=None)

      self.**assertEqual**(self.calc.**add**(-5, 5), 0, msg=None)

      self.**assertEqual**(self.calc.**add**(-7, -5), -12, msg=None)

   def **test\_subtract**(self):

      self.**assertEqual**(self.calc.**subtract**(2, 7), -5, msg=None)

      self.**assertEqual**(self.calc.**subtract**(-5, 6), -11, msg=None)

      self.**assertEqual**(self.calc.**subtract**(7, -5), 12, msg=None)

   def **test\_multiply**(self):

      self.**assertEqual**(self.calc.**multiply**(2, 7), 14, msg=None)

      self.**assertEqual**(self.calc.**multiply**(-5, 6), -30, msg=None)

      self.**assertEqual**(self.calc.**multiply**(-1, 1), -1, msg=None)

      self.**assertEqual**(self.calc.**multiply**(-1, -1), 1, msg=None)

   def **test\_divide**(self):

      self.**assertEqual**(self.calc.**divide**(10, 5), 2, msg=None)

      self.**assertEqual**(self.calc.**divide**(-1, 1), -1, msg=None)

      self.**assertEqual**(self.calc.**divide**(-1, -1), 1, msg=None)

      self.**assertEqual**(self.calc.**divide**(5, 2), 2.5, msg=None)

   def **test\_interger\_divide**(self):

      self.**assertEqual**(self.calc.**integer\_divide**(10, 5), 2, msg=None)

      self.**assertEqual**(self.calc.**integer\_divide**(-1, 1), -1, msg=None)

      self.**assertEqual**(self.calc.**integer\_divide**(-1, -1), 1, msg=None)

      self.**assertEqual**(self.calc.**integer\_divide**(5, 2), 2, msg=None)

   def **test\_power**(self):

      self.**assertEqual**(self.calc.**power**(10, 2), 100, msg=None)

      self.**assertEqual**(self.calc.**power**(-1, 1), -1, msg=None)

      self.**assertEqual**(self.calc.**power**(2, -2), 0.25, msg=None)

   def **test\_square**(self):

      self.**assertEqual**(self.calc.**square**(10), 100, msg=None)

      self.**assertEqual**(self.calc.**square**(-6), 36, msg=None)

   def **test\_root**(self):

      self.**assertEqual**(self.calc.**root**(16), 4, msg=None)

      self.**assertEqual**(self.calc.**root**(25), 5, msg=None)

      self.**assertEqual**(self.calc.**root**(625), 25, msg=None)

   def **test\_factorial**(self):

      self.**assertEqual**(self.calc.**factorial**(1), 1, msg=None)

      self.**assertEqual**(self.calc.**factorial**(0), 1, msg=None)

      self.**assertEqual**(self.calc.**factorial**(5), 120, msg=None)

if \_\_name\_\_ == '\_\_main\_\_':

   unittest.main()