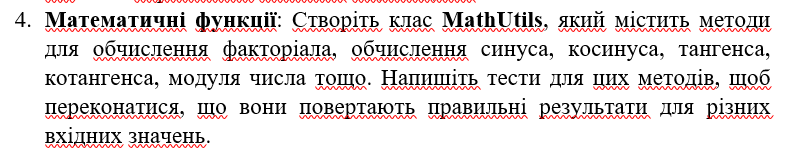
Частина 1  


Код  
import unittest

import math

class MathUtils:

def factorial(self, n):

if n == 0:

return 1

else:

return n \* self.factorial(n-1)

def sine(self, x):

return math.sin(x)

def cosine(self, x):

return math.cos(x)

def tangent(self, x):

return math.tan(x)

def cotangent(self, x):

return 1 / math.tan(x)

def absolute(self, x):

return abs(x)

class TestMathUtils(unittest.TestCase):

def setUp(self):

self.math\_utils = MathUtils()

def test\_factorial(self):

self.assertEqual(self.math\_utils.factorial(0), 1)

self.assertEqual(self.math\_utils.factorial(1), 1)

self.assertEqual(self.math\_utils.factorial(5), 120)

def test\_sine(self):

self.assertAlmostEqual(self.math\_utils.sine(0), 0.0)

self.assertAlmostEqual(self.math\_utils.sine(math.pi/2), 1.0)

def test\_cosine(self):

self.assertAlmostEqual(self.math\_utils.cosine(0), 1.0)

self.assertAlmostEqual(self.math\_utils.cosine(math.pi), -1.0)

def test\_tangent(self):

self.assertAlmostEqual(self.math\_utils.tangent(0), 0.0)

self.assertAlmostEqual(self.math\_utils.tangent(math.pi/4), 1.0)

def test\_cotangent(self):

self.assertAlmostEqual(self.math\_utils.cotangent(math.pi/4), 1.0)

self.assertAlmostEqual(self.math\_utils.cotangent(math.pi/2), 0.0)

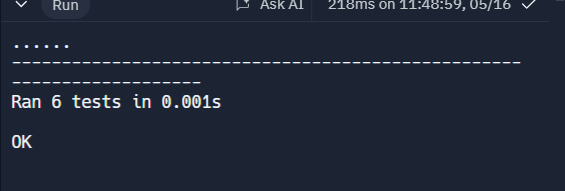
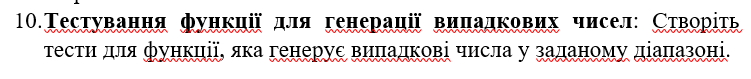
def test\_absolute(self):

self.assertEqual(self.math\_utils.absolute(5), 5)

self.assertEqual(self.math\_utils.absolute(-5), 5)

self.assertEqual(self.math\_utils.absolute(0), 0)

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

unittest.main()  
Результат  
  
Частина 2  
  
Код  
import pytest

import random

def generate\_random\_number(start, end):

return random.randint(start, end)

def test\_generate\_random\_number\_within\_range():

start = -1

end = 100

random\_number = generate\_random\_number(start, end)

assert start <= random\_number <= end

def test\_generate\_random\_number\_with\_negative\_range():

start = -100

end = -50

random\_number = generate\_random\_number(start, end)

assert start <= random\_number <= end

def test\_generate\_random\_number\_with\_same\_range():

start = 10

end = 10

random\_number = generate\_random\_number(start, end)

assert start <= random\_number <= end

def test\_generate\_random\_number\_with\_reversed\_range():

start = 100

end = 1

random\_number = generate\_random\_number(start, end)

assert start <= random\_number <= end

Результат  
