Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

**Звіт**

з лабораторної роботи №5

предмета «Технологія розроблення програмних систем»

Виконав:

Грищенко Б.А.

Група ПП-41

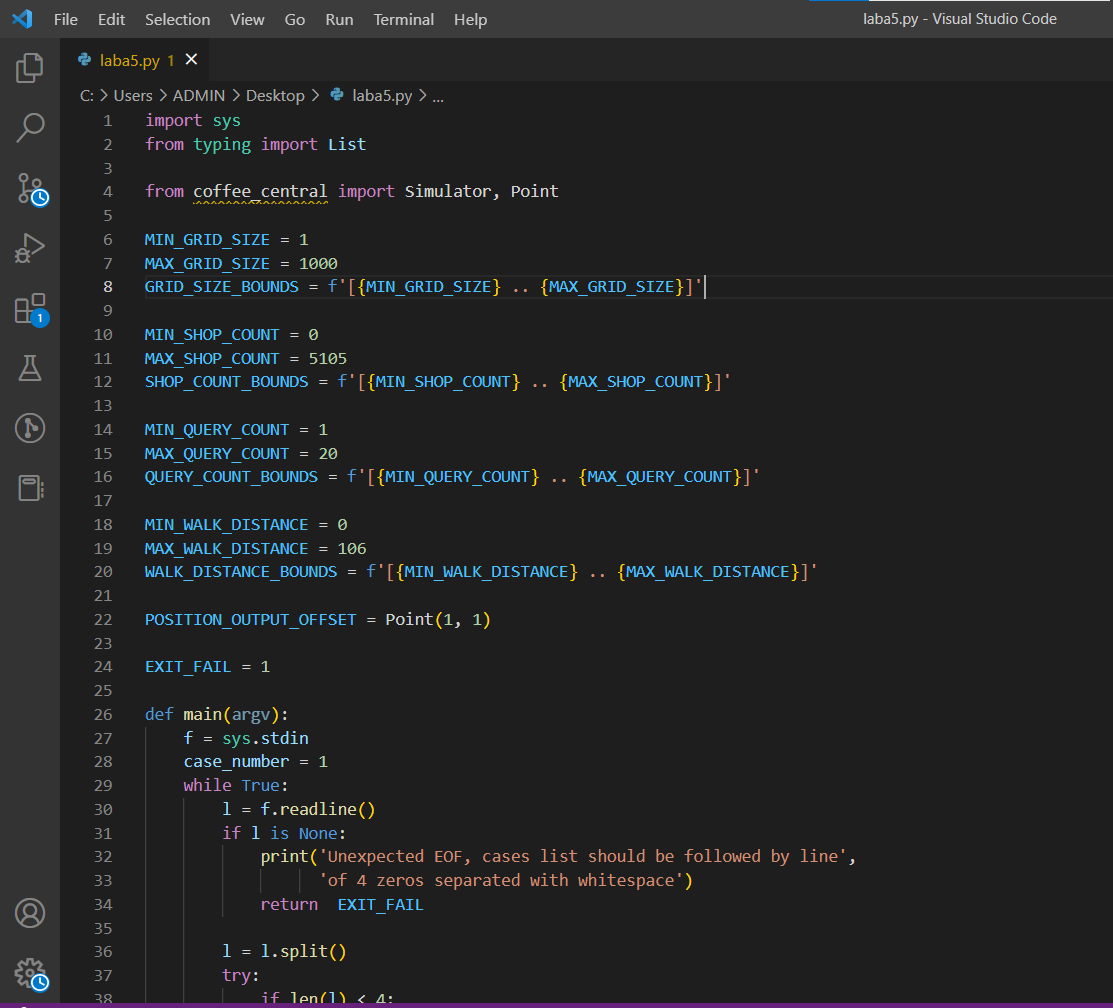
Перевірив:

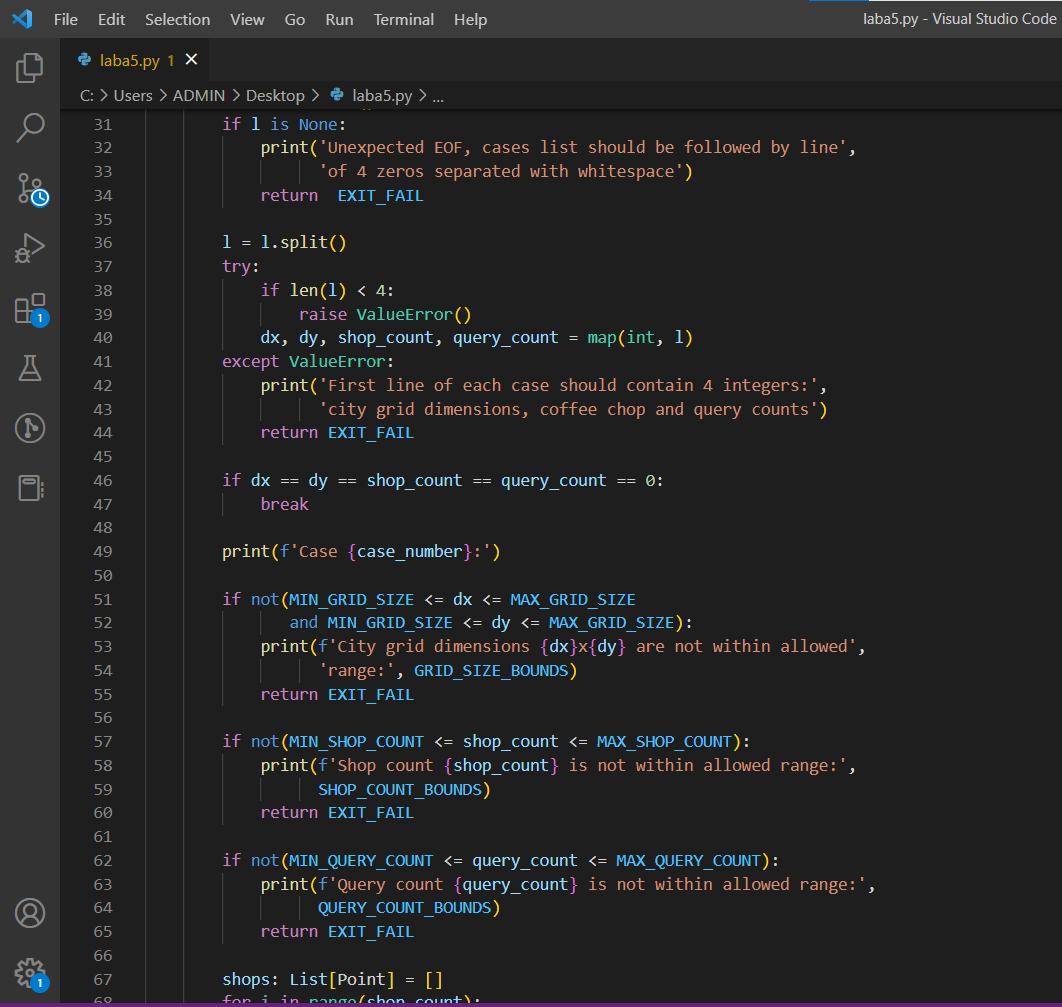
Білий Р.О.

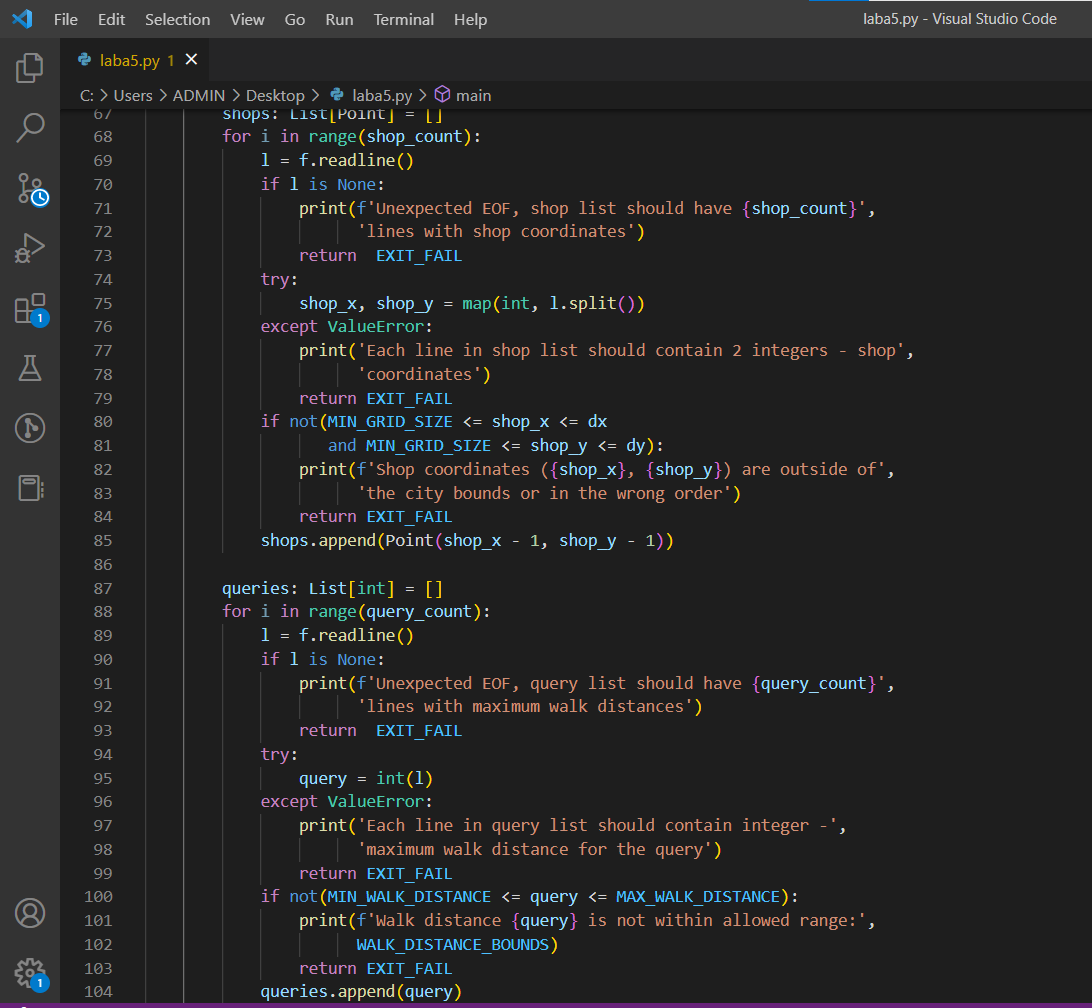
Київ 2022

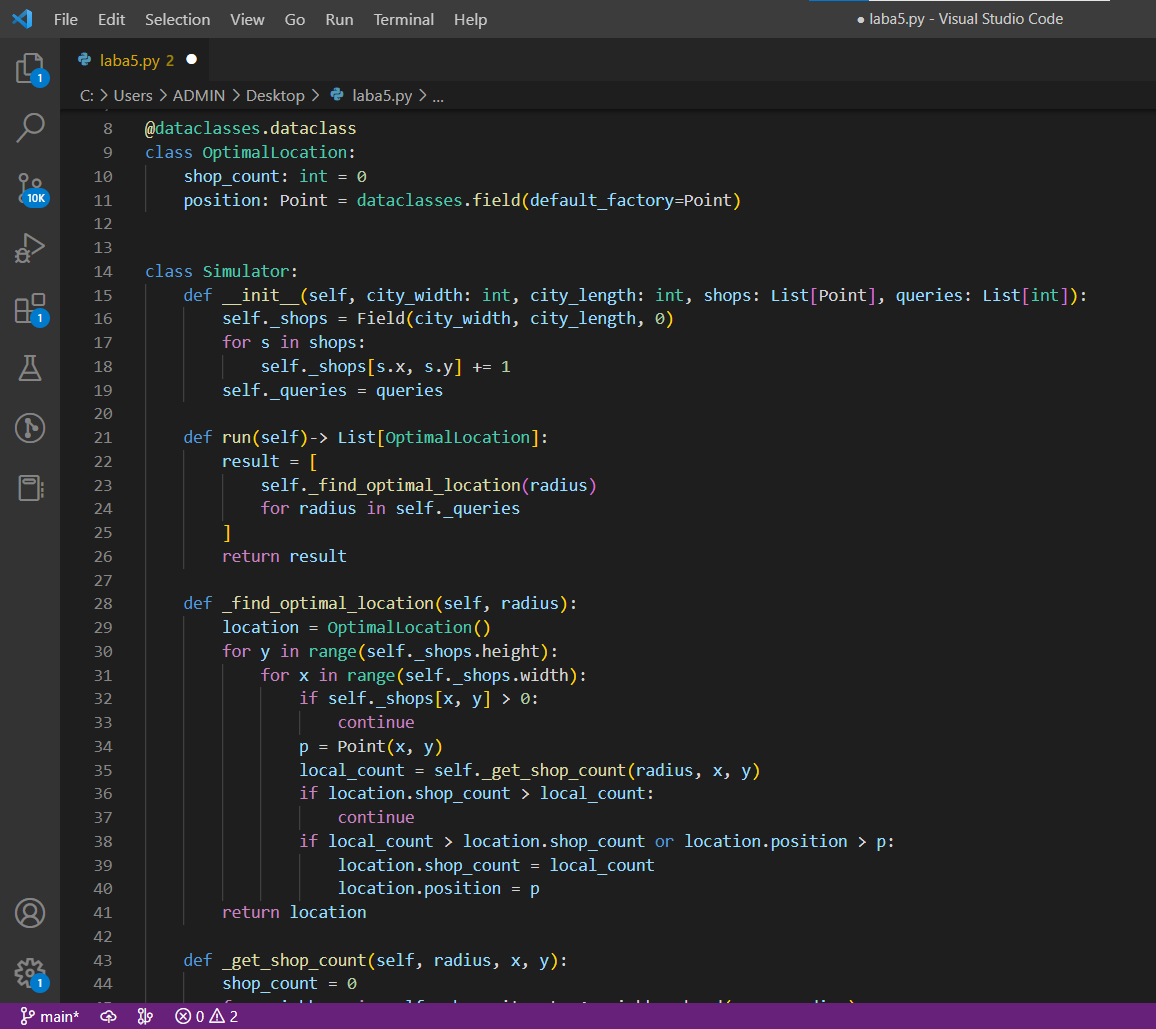
**Тема:** Модульні тести

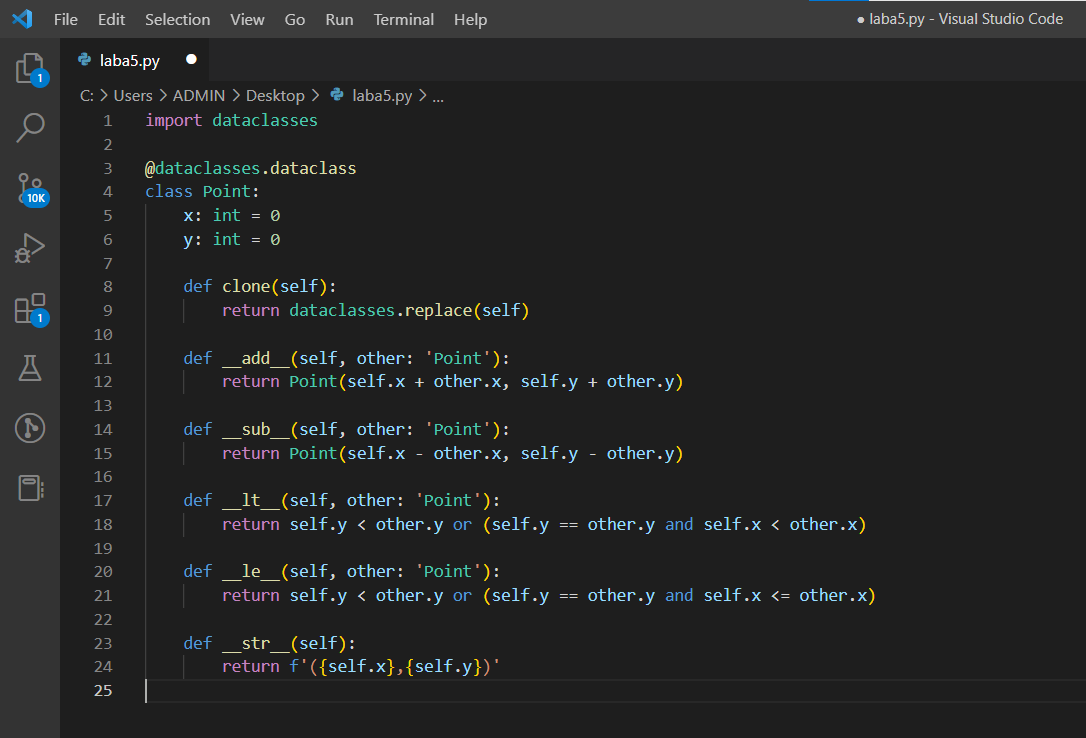
Код для Coffe Central.

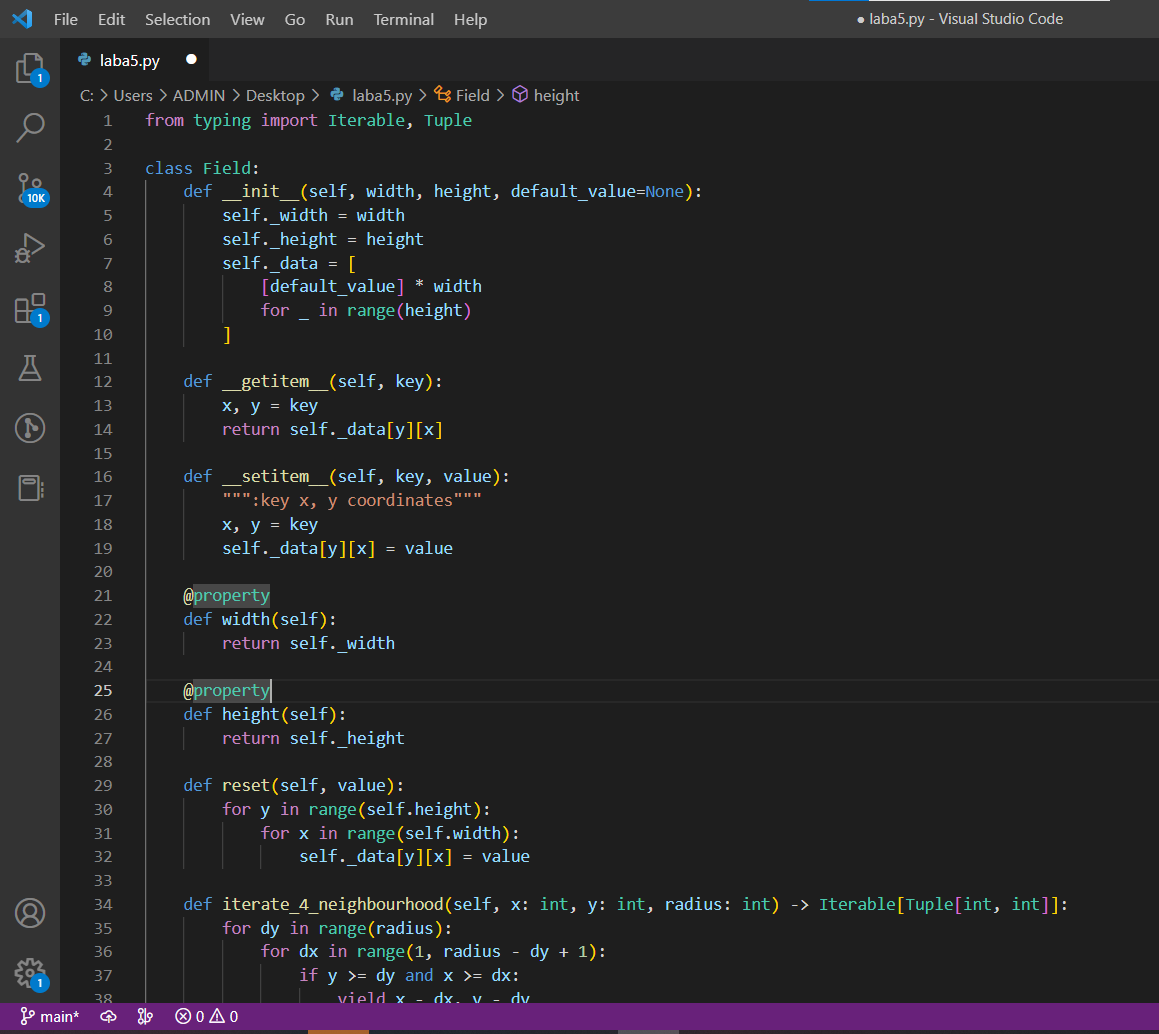












Код тестів:

test.py

import unittest  
  
from coffee\_central import Simulator, Point  
from coffee\_central.field import Field  
  
class TestPoint(unittest.TestCase):  
  
 def setUp(self):  
 self.point\_1 = Point(4,5)  
 self.point\_2 = Point(1,-2)  
 self.point\_3 = Point(4,6)  
 self.point\_4 = Point(5,5)  
 self.point\_5 = Point(4,-5)  
 self.point\_6 = Point(4,5)  
  
 def test\_add(self):  
 test\_point = self.point\_1 + self.point\_2  
 self.assertEqual(test\_point.x, 5)  
 self.assertEqual(test\_point.y, 3)  
   
 def test\_sub(self):  
 test\_point = self.point\_1 - self.point\_2  
 self.assertEqual(test\_point.x, 3)  
 self.assertEqual(test\_point.y, 7)  
   
 def test\_str(self):  
 self.assertEqual(str(self.point\_1),"(4,5)")  
   
 def test\_cmp(self):  
 try:  
 self.assertTrue(self.point\_1 <= self.point\_1)  
 except NotImplementedError:  
 None  
 try:  
 self.assertFalse(self.point\_1 < self.point\_1)  
 self.assertFalse(self.point\_1 < self.point\_2)  
 self.assertTrue(self.point\_1 < self.point\_3)  
 self.assertTrue(self.point\_1 < self.point\_4)  
 except NotImplementedError:  
 None  
   
 try:  
 self.assertTrue(self.point\_1 >= self.point\_1)  
 except NotImplementedError:  
 None  
   
 try:  
 self.assertFalse(self.point\_1 > self.point\_1)  
 self.assertFalse(self.point\_2 > self.point\_1)  
 self.assertTrue(self.point\_3 > self.point\_1)  
 self.assertTrue(self.point\_4 > self.point\_1)  
 except NotImplementedError:  
 None  
   
 try:  
 self.assertFalse(self.point\_1 == self.point\_2)  
 self.assertFalse(self.point\_1 == self.point\_5)  
 self.assertTrue(self.point\_1 == self.point\_6)  
 except NotImplementedError:  
 None  
   
class TestField(unittest.TestCase):  
  
 def setUp(self):  
 self.field = Field(100,400,0)  
  
 def test\_constructor(self):  
 f = Field(0,0,0)  
 self.assertEqual(len(f.\_data), 0)  
   
 f = Field(1,0,0)  
 self.assertEqual(len(f.\_data), 0)  
   
 f = Field(0,1,0)  
 self.assertEqual(len(f.\_data), 1)  
 self.assertEqual(len(f.\_data[0]), 0)  
   
 f = Field(3,8,4)  
 self.assertEqual(f.width, 3)  
 self.assertEqual(f.height, 8)  
 for i in range(3):  
 for k in range(8):  
 self.assertEqual(f[i,k], 4)  
   
 f = Field(8,8)  
 for i in range(8):  
 for k in range(8):  
 self.assertEqual(f[i,k], None)  
   
 def test\_getitem\_setitem(self):  
 self.field[3,5] = 123456  
 self.assertEqual(self.field[3,5], 123456)  
   
 def test\_reset(self):  
 self.field.reset(5)  
 for i in range(self.field.width):  
 for k in range(self.field.height):  
 self.assertEqual(self.field[i,k], 5)  
   
 self.field.reset(16)  
 for i in range(self.field.width):  
 for k in range(self.field.height):  
 self.assertEqual(self.field[i,k], 16)  
   
 def test\_iterate\_4\_neighbourhood(self):  
 result = list(self.field.iterate\_4\_neighbourhood(0,0,0))  
 self.assertEqual(len(result), 0)  
   
 result = list(self.field.iterate\_4\_neighbourhood(0,0,1))  
 expected\_result = [(1,0),(0,1)]  
 self.assertEqual(len(result), len(expected\_result))  
 for pair in expected\_result:  
 self.assertTrue(pair in result)  
   
 result = list(self.field.iterate\_4\_neighbourhood(self.field.width - 1,self.field.height - 1,1))  
 expected\_result = [  
 (self.field.width - 2,self.field.height - 1),  
 (self.field.width - 1,self.field.height - 2)  
 ]  
 self.assertEqual(len(result), len(expected\_result))  
 for pair in expected\_result:  
 self.assertTrue(pair in result)  
   
 result = list(self.field.iterate\_4\_neighbourhood(0,0,2))  
 expected\_result = [(1,0),(0,1),(2,0),(0,2),(1,1)]  
 self.assertEqual(len(result), len(expected\_result))  
 for pair in expected\_result:  
 self.assertTrue(pair in result)  
   
class TestSimulator(unittest.TestCase):  
 def test\_simulation(self):  
 s = Simulator(4,4,[Point(0,0),Point(0,1),Point(2,2),Point(3,3),Point(1,3)],[1,2,4])  
 best\_locations = s.run()  
 self.assertEqual(len(best\_locations), 3)  
 self.assertEqual(best\_locations[0].shop\_count, 3)  
 self.assertEqual(best\_locations[0].position, Point(2,3))  
 self.assertEqual(best\_locations[1].shop\_count, 4)  
 self.assertEqual(best\_locations[1].position, Point(1,1))  
 self.assertEqual(best\_locations[2].shop\_count, 5)  
 self.assertEqual(best\_locations[2].position, Point(2,0))  
   
 s = Simulator(4,4,[Point(0,0),Point(0,1),Point(2,2),Point(3,3),Point(1,3)],[])  
 best\_locations = s.run()  
 self.assertEqual(len(best\_locations), 0)  
   
 s = Simulator(0,0,[],[])  
 best\_locations = s.run()  
 self.assertEqual(len(best\_locations), 0)  
   
if (\_\_name\_\_ == '\_\_main\_\_'):  
 unittest.main()

Всі тести відпрацювали правильно, отже програма створена коректно.

**Висновки:**

У цій лабораторній я навчився розробляти модульні тести до різних елементів програми