Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Верификация программного обеспечения

ОТЧЁТ

к лабораторной работе №3

на тему

UNIT-ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Выполнил студент гр. 253505 Сенько Н.С.

Проверил ассистент кафедры проектирования информационно-компьютерных систем Сморгун Е.С.

Минск 2024

1 ПРАКТИЧЕСКАЯ ЧАСТЬ

## **1.1 Задача 1**

Напишите программу, выводящую в консоль надпись

Hello, world!

And hi again!

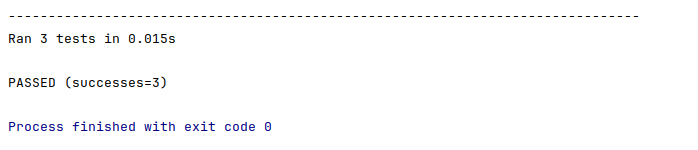
!!!!!!!!!!

Число восклицательных знаков в третьей строке должно быть случайным в диапазоне 5-50.

Таблица 1.1 – Unit-тестирование задачи 1.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка корректного вывода первой строки | Первой строкой в консоль должно выводиться «Hello, world!» | Тест пройден |
| Проверка корректного вывода второй строки | Второй строкой в консоль должно выводиться «And hi again!» | Тест пройден |
| Проверка корректного количества выведенных знаков в третьей строке | В третьей строке должно быть от 5 до 50 восклицательных знаков | Тест пройден |

class TestHellowWorld(unittest.TestCase):  
 def test\_print\_hello\_world\_hello\_word\_in\_out(self):  
 # Перехватываем вывод в stdout  
 import io  
 from unittest.mock import patch  
 with patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout:  
 print\_hello\_world()  
 output = mock\_stdout.getvalue()  
 # Проверяем, что в выводе есть текст "Hello, world!"  
 assert "Hello, world!" in output  
  
 def test\_print\_hello\_world\_and\_hi\_again\_in\_out(self):  
 """  
 Тестирует функцию print\_hello\_world()  
 Проверяет, что функция печатает текст с правильным количеством восклицательных знаков  
 """  
 # Перехватываем вывод в stdout  
 import io  
 from unittest.mock import patch  
 with patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout:  
 print\_hello\_world()  
 output = mock\_stdout.getvalue()  
  
 # Проверяем, что в выводе есть текст "And hi again!"  
 assert "And hi again!" in output  
  
  
 def test\_print\_hello\_count\_znak(self):  
  
 # Перехватываем вывод в stdout  
 import io  
 from unittest.mock import patch  
 with patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout:  
 print\_hello\_world()  
 output = mock\_stdout.getvalue()  
  
 # Проверяем, что количество восклицательных знаков в третьей строке находится в диапазоне 5-50  
 lines = output.splitlines()  
 assert 5 <= len(lines[2]) <= 50  
 assert lines[2].strip() == "!" \* len(lines[2])



## **1.2 Задача 2**

Напишите программу, которая получает с клавиатуры имя, фамилию и возраст людей. После

того, как ввод завершён, программа выводит список всех людей в формате

*Фамилия Имя Возраст*

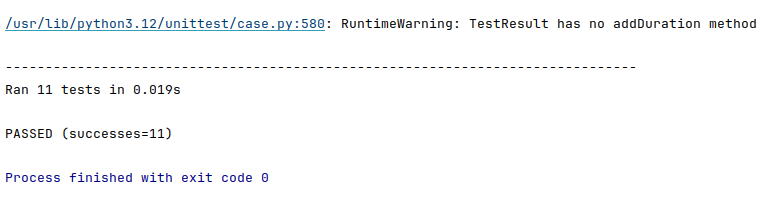
и считает (и выводит последней строкой): самый малый возраст, самый большой возраст, средний

возраст (с точностью до сотых года).

Таблица 1.2 – Unit-тестирование задачи 2.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка работы добавления валидных данных людей | Программа не выводит сообщения об ошибке | Тест пройден |
| Проверка работы добавления валидных данных людей с пробелами | Программа не выводит сообщения об ошибке | Тест пройден |
| Проверка работы программы при невалидном значении возраста | Программа должна вызывать исключение при каждом невалидном значении | Тест пройден |
| Проверка работы программы при отсутствии данных о людях | Программа должна выводить корректно работать и выводить «Нет данных для статистики» | Тест пройден |
| Проверка расчёта статистики при правильных входных данных | Программа выводит корректные данные статистики | Тест пройден |
| Проверка расчёта статистики при правильных входных данных нескольких людей | Программа выводит корректные данные статистики | Тест пройден |
| Проверка вывода | Программа выводит корректные данные | Тест пройден |

class TestPeopleData(unittest.TestCase):  
 def test\_parse\_person\_data\_valid(self):  
 self.assertEqual(parse\_person\_data('Иван Иванов 25'), ('Иванов', 'Иван', 25))  
  
 def test\_parse\_person\_data\_valid2(self):  
 self.assertEqual(parse\_person\_data('Иван Иванов 25'), ('Иванов', 'Иван', 25))  
  
 def test\_parse\_person\_data\_valid3(self):  
 self.assertEqual(parse\_person\_data('Иван \nИванов \n25'), ('Иванов', 'Иван', 25))  
  
 def test\_parse\_person\_data\_invalid(self):  
 with self.assertRaises(ValueError):  
 parse\_person\_data('Иван Иванов')  
  
 def test\_calculate\_age\_statistics(self):  
 people = [('Иванов', 'Иван', 25), ('Петров', 'Петр', 30), ('Сидоров', 'Сидор', 20)]  
 min\_age, max\_age, avg\_age = calculate\_age\_statistics(people)  
 self.assertEqual(min\_age, 20)  
 self.assertEqual(max\_age, 30)  
 self.assertAlmostEqual(avg\_age, 25.00, places=2)  
  
 def test\_zero\_person(self):  
 people = []  
 min\_age, max\_age, avg\_age = calculate\_age\_statistics(people)  
 self.assertEqual(min\_age, None)  
 self.assertEqual(max\_age, None)  
 self.assertAlmostEqual(avg\_age, None)  
  
 def test\_single\_person(self):  
 people = [('Смирнов', 'Алексей', 40)]  
 min\_age, max\_age, avg\_age = calculate\_age\_statistics(people)  
 self.assertEqual(min\_age, 40)  
 self.assertEqual(max\_age, 40)  
 self.assertAlmostEqual(avg\_age, 40.00, places=2)  
  
 def test\_same\_ages(self):  
 people = [('Иванов', 'Иван', 30), ('Петров', 'Петр', 30), ('Сидоров', 'Сидор', 30)]  
 min\_age, max\_age, avg\_age = calculate\_age\_statistics(people)  
 self.assertEqual(min\_age, 30)  
 self.assertEqual(max\_age, 30)  
 self.assertAlmostEqual(avg\_age, 30.00, places=2)  
  
 def test\_display\_people\_data(self):  
 people = [('Иванов', 'Иван', 25), ('Петров', 'Петр', 30)]  
 expected\_output = "Иванов Иван 25\nПетров Петр 30\n"  
  
 captured\_output = io.StringIO()  
 sys.stdout = captured\_output  
 display\_people\_data(people)  
 sys.stdout = sys.\_\_stdout\_\_  
  
 self.assertEqual(captured\_output.getvalue(), expected\_output)  
  
 @patch('builtins.input', side\_effect=['Иван Иванов 12', 'stop'])  
 def test\_input(self, mock):  
 people = get\_people\_data()  
 person = [('Иванов', 'Иван', 12)]  
 self.assertEqual(person, people)  
  
 @patch('builtins.input', side\_effect=['Иван Иванов 10',  
 'Иван Иванов 42',  
 'stop'])  
 def test\_input2(self, mock):  
 people = get\_people\_data()  
 person = [('Иванов', 'Иван', 10), ('Иванов', 'Иван', 42)]  
 self.assertEqual(person, people)



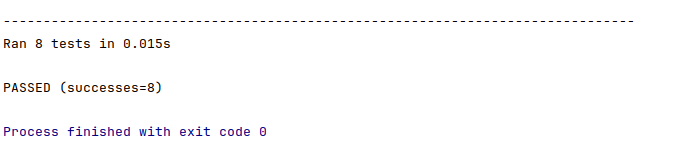
## **1.3 Задача 3**

Напишите программу, получающую из командной строки два числа – длину и ширину прямоугольника, а затем вычисляющую его площадь. Для тех, кто не знал или забыл: площадь прямоугольника вычисляется по формуле S = a\*b. Помните о надёжности, уделите ей максимум внимания.

Таблица 1.3 – Unit-тестирование задачи 3.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка работы программы с корректными входными данными | Программа должна корректно посчитать площадь прямоугольника с заданными длиной и шириной | Тест пройден |
| Проверка нулевой длины | Программа должна корректно посчитать площадь прямоугольника с заданными длиной и шириной | Тест пройден |
| Проверка орицательной длины | Функция кидает исключение | Тест пройден |
| Проверка орицательной ширины | Функция кидает исключение | Тест пройден |
| Проверка некорректного ввода | Функция кидает исключение | Тест пройден |

class TestRectangleArea(unittest.TestCase):  
 def test\_positive\_values(self):  
 self.assertEqual(calculate\_rectangle\_area(4, 5), 20)  
  
 def test\_zero\_length(self):  
 self.assertEqual(calculate\_rectangle\_area(0, 5), 0)  
  
 def test\_zero\_width(self):  
 self.assertEqual(calculate\_rectangle\_area(4, 0), 0)  
  
 def test\_negative\_length(self):  
 with self.assertRaises(ValueError):  
 calculate\_rectangle\_area(-4, 5)  
  
 def test\_negative\_width(self):  
 with self.assertRaises(ValueError):  
 calculate\_rectangle\_area(4, -5)  
  
 def test\_non\_numeric\_length(self):  
 with self.assertRaises(TypeError):  
 calculate\_rectangle\_area("abc", 5)  
  
 def test\_non\_numeric\_width(self):  
 with self.assertRaises(TypeError):  
 calculate\_rectangle\_area(4, "def")  
  
 @patch('builtins.input', side\_effect=['10', '20'])  
 def test\_input\_value(self, mock):  
 length = int(input("Введите длину: "))  
 width = int(input("Введите ширину: "))  
 area = calculate\_rectangle\_area(length, width)  
 self.assertEqual(area, 200)



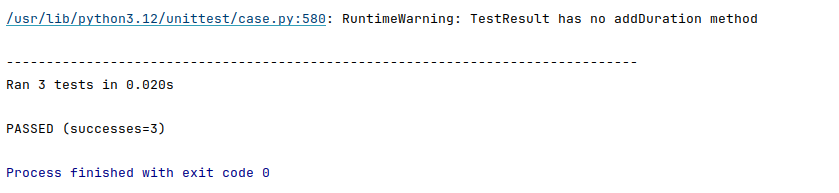
## **1.4 Задача 4**

Напишите программу, генерирующую HTML-файл таблицей, в которой фон строк постепенно меняется от белого к чёрному (с минимально возможным шагом). См. пример на картинке. Все параметры таблицы, её содержимое и т.п. – на ваше усмотрение. Главное, чтобы было максимально похоже на картинку.

Таблица 1.4 – Unit-тестирование задачи 4.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка на корректно составленный HTML-файл | В файле должна записаться корректная HTML-разметка | Тест пройден |
| Проверка наличия каждой строки таблицы | Программа должна генерировать файл с определенным количеством строк в таблице | Тест пройден |
| Проверка заданного оттенка фона каждой строки таблицы | Программа должна генерировать файл с правильным оттенком для каждой строки таблицы | Тест пройден |

class TestHtmlTableGeneration(unittest.TestCase):  
 def test\_html\_file\_creation(self):  
 generate\_html\_table()  
 self.assertTrue(os.path.exists("gradient\_table.html"))  
  
 def test\_html\_content(self):  
 generate\_html\_table(rows=3, columns=2)  
 with open("gradient\_table.html", "r", encoding="utf-8") as file:  
 content = file.read()  
 expected\_start = """<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">"""  
 self.assertTrue(content.startswith(expected\_start))  
 self.assertIn('<tr style="background-color: rgb(255, 255, 255);">', content)  
 self.assertIn('<tr style="background-color: rgb(128, 128, 128);">', content)  
 self.assertIn('<tr style="background-color: rgb(1, 1, 1);">', content)  
 self.assertIn('<td></td><td></td>', content)  
 self.assertNotIn('<td></td><td></td><td></td>', content)  
  
 def test\_html\_content2(self):  
 generate\_html\_table(rows=255, columns=255)  
 with open("gradient\_table.html", "r", encoding="utf-8") as file:  
 content = file.read()  
 expected\_start = """<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">"""  
 self.assertTrue(content.startswith(expected\_start))  
 self.assertIn('<tr style="background-color: rgb(255, 255, 255);">', content)  
 self.assertIn('<tr style="background-color: rgb(128, 128, 128);">', content)  
 self.assertIn('<tr style="background-color: rgb(1, 1, 1);">', content)  
 self.assertIn('<td></td><td></td>', content)  
 self.assertIn('<td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td>', content)  
  
 def test\_html\_content2(self):  
 generate\_html\_table(rows=255, columns=255)  
 with open("gradient\_table.html", "r", encoding="utf-8") as file:  
 content = file.read()  
 expected\_start = """<!DOCTYPE html>  
 <html lang="ru">  
 <head>  
 <meta charset="UTF-8">"""  
 self.assertTrue(content.startswith(expected\_start))  
 self.assertIn('<tr style="background-color: rgb(255, 255, 255);">', content)  
 self.assertIn('<tr style="background-color: rgb(128, 128, 128);">', content)  
 self.assertIn('<tr style="background-color: rgb(1, 1, 1);">', content)  
 self.assertIn('<td></td><td></td>', content)  
 self.assertIn(  
 '<td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td>',  
 content)  
  
 def test\_html\_content3(self):  
 with self.assertRaises(ValueError):  
 generate\_html\_table(rows=-10, columns=-10)  
   
  
 def tearDown(self):  
 if os.path.exists("gradient\_table.html"):  
 os.remove("gradient\_table.html")



## **1.5 Задача 5**

Напишите программу, ищущую в указанной папке (и всех подпапках) файлы с указанным расширением (расширение указывается как параметр командной строки). Результат поиска выводите в консоль в виде:

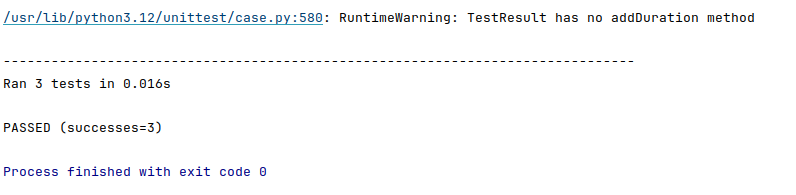
c:/dir1/dir2/file1.ext

c:/dir1/dir2/file2.ext

Таблица 1.5 – Unit-тестирование задачи 5.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка работы программы при поиске файлов с расширением .txt в правильной директории | Программа должна найти необходимые файлы | Тест пройден |
| Проверка файлов в пустой | Программа не должна найти файлы | Тест пройден |
| Проверка файлов во вложенных директорияъ | Программа должна найти определенное количество файлов | Тест пройден |
| Проверка работы программы при поиске файлов с расширением .py в правильной директории | Программа должна найти необходимые файлы | Тест пройден |

class TestSearchFiles(unittest.TestCase):  
  
 def setUp(self):  
 self.base\_dir = f'{os.path.dirname(os.path.abspath(\_\_file\_\_))}/test\_data\_5\_task'  
 os.makedirs(os.path.join(self.base\_dir, "empty\_directory"), exist\_ok=True)  
 os.makedirs(os.path.join(self.base\_dir, "test\_dir"), exist\_ok=True)  
 os.makedirs(os.path.join(self.base\_dir, "test\_dir", "subdir1"), exist\_ok=True)  
  
 with open(os.path.join(self.base\_dir, "test\_dir", "file1.txt"), "w") as f:  
 f.write("Content of file1.txt")  
  
 with open(os.path.join(self.base\_dir, "test\_dir", "file2.txt"), "w") as f:  
 f.write("Content of file2.txt")  
  
 with open(os.path.join(self.base\_dir, "test\_dir", "file.py"), "w") as f:  
 f.write("Content of file.py")  
  
 with open(os.path.join(self.base\_dir, "test\_dir", "subdir1", "file3.txt"), "w") as f:  
 f.write("Content of file3.txt")  
  
 def tearDown(self):  
 shutil.rmtree(self.base\_dir)  
  
 def test\_empty\_directory(self):  
 with (patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout):  
 search\_files(f"{self.base\_dir}/empty\_directory", "txt")  
 output = mock\_stdout.getvalue()  
 lines = output.splitlines()  
 self.assertEqual(len(lines), 0)  
  
  
  
 def test\_subdirectories(self):  
 with (patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout):  
 search\_files(f"{self.base\_dir}/test\_dir", "txt")  
 output = mock\_stdout.getvalue()  
 lines = output.splitlines()  
 result =[  
 f"{self.base\_dir}/test\_dir/file2.txt",  
 f"{self.base\_dir}/test\_dir/file1.txt",  
 f"{self.base\_dir}/test\_dir/subdir1/file3.txt"  
 ]  
 print(lines)  
 self.assertEqual(len(lines), 3)  
 for i in range(3):  
 self.assertEqual(lines[0], result[0])  
  
 def test\_another\_extension(self):  
 with (patch('sys.stdout', new\_callable=io.StringIO) as mock\_stdout):  
 search\_files(f"{self.base\_dir}/test\_dir", "py")  
 output = mock\_stdout.getvalue()  
 lines = output.splitlines()  
 result =[  
 f"{self.base\_dir}/test\_dir/file.py",  
 ]  
 self.assertEqual(len(lines), 1)  
 for i in range(1):  
 self.assertEqual(lines[0], result[0])



## **1.6 Задача 6**

Напишите программу, извлекающую документ с указанного URL и сохраняющую в файл на локальном диске. URL и папка, в которую нужно сохранять файл, передаются как параметры командной строки.

Таблица 1.6 – Unit-тестирование задачи 6.

|  |  |  |
| --- | --- | --- |
| Описание тестового случая | Ожидаемый результат | Полученный результат |
| Проверка работы программы при валидном значении пути в папке | Папка создается | Тест пройден |
| Проверка работы при валидном URL для скачивания файла | Файл создается | Тест пройден |
| Проверка ошибки сохранения | Создается исключение «General Error» | Тест пройден |
| Проверка работы при невалидном URL для скачивания файла | Создается исключение | Тест пройден |

class TestDownloadFile(unittest.TestCase):  
  
 @patch('os.makedirs')  
 @patch('os.path.exists')  
 @patch('requests.get')  
 @patch('builtins.open', new\_callable=mock\_open)  
 def test\_download\_file\_success(self, mock\_open\_fn, mock\_requests\_get, mock\_path\_exists, mock\_makedirs):  
 # Mock the response object returned by requests.get  
 mock\_response = mock.Mock()  
 mock\_response.status\_code = 200  
 mock\_response.content = b'file content'  
 mock\_requests\_get.return\_value = mock\_response  
  
 # Simulate that the directory already exists  
 mock\_path\_exists.return\_value = True  
  
 # Call the function  
 url = "http://example.com/file.txt"  
 save\_dir = "/fake/dir"  
 download\_file(url, save\_dir)  
  
 # Ensure requests.get was called with the URL  
 mock\_requests\_get.assert\_called\_once\_with(url)  
  
 # Ensure open was called correctly to save the file as 'filename'  
 file\_path = os.path.join(save\_dir, 'filename')  
 mock\_open\_fn.assert\_called\_once\_with(file\_path, 'wb')  
  
 # Ensure the content was written to the file  
 mock\_open\_fn().write.assert\_called\_once\_with(b'file content')  
  
 # Ensure os.makedirs was not called since the directory exists  
 mock\_makedirs.assert\_not\_called()  
  
 @patch('os.makedirs')  
 @patch('os.path.exists')  
 @patch('requests.get')  
 @patch('builtins.open', new\_callable=mock\_open)  
 def test\_download\_file\_directory\_creation(self, mock\_open\_fn, mock\_requests\_get, mock\_path\_exists, mock\_makedirs):  
 # Mock the response object returned by requests.get  
 mock\_response = mock.Mock()  
 mock\_response.status\_code = 200  
 mock\_response.content = b'some content'  
 mock\_requests\_get.return\_value = mock\_response  
  
 # Simulate that the directory does not exist  
 mock\_path\_exists.return\_value = False  
  
 # Call the function  
 url = "http://example.com/anotherfile.txt"  
 save\_dir = "/another/fake/dir"  
 download\_file(url, save\_dir)  
  
 # Ensure os.makedirs was called to create the directory  
 mock\_makedirs.assert\_called\_once\_with(save\_dir)  
  
 # Ensure open was called correctly to save the file as 'filename'  
 file\_path = os.path.join(save\_dir, 'filename')  
 mock\_open\_fn.assert\_called\_once\_with(file\_path, 'wb')  
  
 # Ensure the content was written to the file  
 mock\_open\_fn().write.assert\_called\_once\_with(b'some content')  
  
 @patch('os.makedirs')  
 @patch('os.path.exists')  
 @patch('requests.get')  
 def test\_download\_file\_http\_error(self, mock\_requests\_get, mock\_path\_exists, mock\_makedirs):  
 # Mock an HTTP error response  
 mock\_requests\_get.side\_effect = requests.exceptions.HTTPError("HTTP Error")  
  
 # Simulate that the directory exists  
 mock\_path\_exists.return\_value = True  
  
 # Use a mock for the print function to capture output  
 with patch('builtins.print') as mock\_print:  
 url = "http://example.com/errorfile.txt"  
 save\_dir = "/error/fake/dir"  
 download\_file(url, save\_dir)  
  
 # Ensure the error message was printed  
 mock\_print.assert\_called\_once\_with("Произошла ошибка при загрузке: HTTP Error")  
  
 @patch('os.makedirs')  
 @patch('os.path.exists')  
 @patch('requests.get')  
 def test\_download\_file\_general\_exception(self, mock\_requests\_get, mock\_path\_exists, mock\_makedirs):  
 # Mock a general exception during the file write process  
 mock\_requests\_get.side\_effect = Exception("General Error")  
  
 # Simulate that the directory exists  
 mock\_path\_exists.return\_value = True  
  
 # Use a mock for the print function to capture output  
 with patch('builtins.print') as mock\_print:  
 url = "http://example.com/generalerror.txt"  
 save\_dir = "/generalerror/fake/dir"  
 download\_file(url, save\_dir)  
  
 # Ensure the error message was printed  
 mock\_print.assert\_called\_once\_with("Произошла ошибка: General Error")

