МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №10
по дисциплине основы программной инженерии

Выполнил: Духно Михаил Александрович, 2 курс, группа ПИЖ-б-о-20-1, Проверил: Доцент кафедры инфокоммуникаций, Воронкин Р.А.

Ставрополь, 2022 г

Программный код программы расписания поездов:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import sqlite3




def add_element(name, num, tm, conn):
    tm = tm.split(' ')
    tm = f'{tm[0]}-{tm[1]}-{tm[2]}'
    cur = conn.cursor()
    query = f"""
    INSERT INTO citys (id, city)
    VALUES
    ({num}, '{name}');
    """
    cur.execute(query)
    conn.commit()


    query = f"""
        INSERT INTO train (train_id, time, city)
        VALUES
        ({num}, '{tm}', {num})
        """
    cur.execute(query)
    conn.commit()
    cur.close()




def find_train(num, conn):
    cur = conn.cursor()
    query = f"""SELECT *
    FROM train
    JOIN citys ON train.city=citys.id
    WHERE train.train_id = {num}"""
    cur.execute(query)
    res = cur.fetchone()
    cur.close()
```

```
    if len(res) > 0:
        print(
            f'Конечный пункт: {res[4]} \n'
            f'Номер поезда: {res[0]} \n'
            f'Время отправления: {res[1]}'
        )
        return len(res)
    else:
        print('Поезда с таким номером нет')
        return len(res)




if __name__ == '__main__':
    print('LOADING...')
    conn = sqlite3.connect('data.db')
    cur = conn.cursor()
    query = """
    CREATE TABLE IF NOT EXISTS citys(
    id INT PRIMARY KEY,
    city VARCHAR(50));"""


    cur.execute(query)
    conn.commit()


    query = """
        CREATE TABLE IF NOT EXISTS train(
        train_id INT PRIMARY KEY,
        time DATATIME,
        city INT,
        FOREIGN KEY (city) REFERENCES
citys(id));"""


    cur.execute(query)
    conn.commit()
    cur.close()
    print('Hello!')


    flag = True
    while flag:
```

```
        print('1. Добавить новый поезд')
        print('2. Вывести информацию о поезде')
        print('3.Выход из программы')
        com = int(input('введите номер команды: '))
        if com == 1:
            name = input('city: ')
            num = input('num: ')
            tm = input('tm: ')
            add_element(name, num, tm, conn)
        elif com == 2:
            train_num = input('Введите номер
поезда: ')
            length = find_train(train_num, conn)
        elif com == 3:
            flag = False
    conn.close()
```

Код программы тестирования основного приложения:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import program
import sqlite3
import datetime




def add_train(conn):
    program.add_element('Moscow', '25', '2022 6 3',
conn)
    cur = conn.cursor()
    query = """SELECT *
                FROM train
                WHERE train_id = 25"""
    cur.execute(query)
    res = cur.fetchall()
    return len(res) > 0


def find_train(conn):
    length = program.find_train('25', conn)
    return length > 0
```

```
if __name__ == '__main__':
    conn = sqlite3.connect('data.db')
    res = add_train(conn)
    print(f'add train: {res}')
    res = find_train(conn)
    print(f'find train: {res}')
```

Контрольные вопросы:
1. Для чего используется автономное тестирование?
Для того чтобы проверить различные условия программы всеми возможными значениями
2. Какие фреймворки Python получили наибольшее распространение для решения задач
автономного тестирования?
unittest, pytest, nose
3. Какие существуют основные структурные единицы модуля unittest?
Test fixture
Test fixture – обеспечивает подготовку окружения для выполнения тестов, а также организацию
мероприятий по их корректному завершению (например очистка ресурсов). Подготовка окружения может включать в себя создание баз данных, запуск необходим серверов и т.п.
Test case
Test case – это элементарная единица тестирования, в рамках которой проверяется работа
компонента тестируемой программы (метод, класс, поведение и т. п.). Для реализации этой
сущности используется класс TestCase.
Test suite
Test suite – это коллекция тестов, которая может в себя включать как отдельные test case’ы так и
целые коллекции (т.е. можно создавать коллекции коллекций). Коллекции используются с целью
объединения тестов для совместного запуска.
Test runner
Test runner – это компонент, которые оркестрирует (координирует взаимодействие) запуск тестов
и предоставляет пользователю результат их выполнения. Test runner может иметь графический
интерфейс, текстовый интерфейс или возвращать какое-то заранее заданное значение, которое
будет описывать результат прохождения тестов.

4. Какие существуют способы запуска тестов unittest?
Интерфейс командной строки (CLI)

Графический интерфейс пользователя (GUI)

5. Каково назначение класса TestCase?

Как уже было сказано – основным строительным элементом при написании тестов с использованием unittest является TestCase. Он представляет собой класс, который должен
являться базовым для всех остальных классов, методы которых будут тестировать те или иные
автономные единицы исходной программы.

6. Какие методы класса TestCase выполняются при запуске и завершении работы тестов?

setUp()

Метод вызывается перед запуском теста. Как правило, используется для подготовки окружения
для теста.

tearDown()

Метод вызывается после завершения работы теста. Используется для “приборки” за тестом.

setUpClass()

Метод действует на уровне класса, т.е. выполняется перед запуском тестов класса. При этом
синтаксис требует наличие декоратора @classmethod.

tearDownClass()

Запускается после выполнения всех методов класса, требует наличия декоратора @classmethod.

7. Какие методы класса TestCase используются для проверки условий и генерации ошибок?

Метод	Описание
`assertEqual(a, b)`	a == b
`assertNotEqual(a, b)`	a != b
`assertTrue(x)`	bool(x) is True
`assertFalse(x)`	bool(x) is False
`assertIs(a, b)`	a is b
`assertIsNot(a, b)`	a is not b
`assertIsNone(x)`	x is None
`assertIsNotNone(x)`	x is not None
`assertIn(a, b)`	a in b
`assertNotIn(a, b)`	a not in b
`assertIsInstance(a, b)`	isinstance(a, b)
`assertNotIsInstance(a, b)`	not isinstance(a, b)

8. Какие методы класса TestCase позволяют собирать информацию о самом тесте?
countTestCases()
Возвращает количество тестов в объекте класса-наследника от TestCase.
id()
Возвращает строковый идентификатор теста. Как правило это полное имя метода, включающее
имя модуля и имя класса.
shortDescription()
Возвращает описание теста, которое представляет собой первую строку docstring'а метода, если
его нет, то возвращает None.

9. Каково назначение класса TestSuite? Как осуществляется загрузка тестов?
Класс TestSuite используется для объединения тестов в группы, которые могут включать в себя
как отдельные тесты так и заранее созданные группы. Помимо этого, TestSuite предоставляет
интерфейс, позволяющий TestRunner'у, запускать тесты.

10. Каково назначение класса TestResult?
Класс TestResult используется для сбора информации о результатах прохождения тестов.
11. Для чего может понадобиться пропуск отдельных тестов?
Для того чтобы ускорить тестирование и не тестировать уже протестированные части

12. Как выполняется безусловный и условных пропуск тестов? Как выполнить пропуск класса
тестов?
@unittest.skipIf(condition, reason)
@unittest.skip(reason)
13. Самостоятельно изучить средства по поддержке тестов unittest в PyCharm. Приведите
обобщенный алгоритм проведения тестирования с помощью PyCharm.