

**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 №2.22 по дисциплине основы программной  
инженерии**

Выполнил:  
Шальнев Владимир Сергеевич,  
2 курс, группа ПИЖ-б-о-20-1,

Проверил:  
Доцент кафедры  
прикладной математики и  
компьютерной безопасности,  
Воронкин Р.А.

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2022 г.

## Выполнение:

### Пример:

```
import calc

def test_add():
    if calc.add(1, 2) == 3:
        print("Test add(a, b) is OK")
    else:
        print("Test add(a, b) is Fail")

def test_sub():
    if calc.sub(4, 2) == 2:
        print("Test sub(a, b) is OK")
    else:
        print("Test sub(a, b) is Fail")

def test_mul():
    if calc.mul(2, 5) == 10:
        print("Test mul(a, b) is OK")
    else:
        print("Test mul(a, b) is Fail")

def test_div():
    if calc.div(8, 4) == 2:
        print("Test div(a, b) is OK")
    else:
        print("Test div(a, b) is Fail")

if __name__ == '__main__':
    test_add()
    test_sub()
    test_mul()
    test_div()
```

### Работа примера:

```
Test add(a, b) is OK
Test sub(a, b) is OK
Test mul(a, b) is OK
Test div(a, b) is OK

Process finished with exit code 0
```

### Пример:

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

import unittest
import calc
```

```

class CalcTest(unittest.TestCase):
    """Calc tests"""

    @classmethod
    def setUpClass(cls):
        """Set up for class"""
        print("setUpClass")
        print("=====")

    @classmethod
    def tearDownClass(cls):
        """Tear down for class"""
        print("=====")
        print("tearDownClass")

    def setUp(self):
        """Set up for test"""
        print("Set up for [" + self.shortDescription() + "]")

    def tearDown(self):
        """Tear down for test"""
        print("Tear down for [" + self.shortDescription() + "]")
        print("")

    def test_add(self):
        """Add operation test"""
        print("id: " + self.id())
        self.assertEqual(calc.add(1, 2), 3)

    def test_sub(self):
        """Sub operation test"""
        print("id: " + self.id())
        self.assertEqual(calc.sub(4, 2), 2)

    def test_mul(self):
        """Mul operation test"""
        print("id: " + self.id())
        self.assertEqual(calc.mul(2, 5), 10)

    def test_div(self):
        """Div operation test"""
        print("id: " + self.id())
        self.assertEqual(calc.div(8, 4), 2)

if __name__ == '__main__':
    unittest.main()

```

Работа примера:

Ran 4 tests in 0.008s

OK

Пример:

```

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

```

```

import unittest
import calc_tests

testLoad = unittest.TestLoader()
suites = testLoad.loadTestsFromModule(calc_tests)
testResult = unittest.TestResult()
runner = unittest.TextTestRunner(verbosity=1)
testResult = runner.run(suites)
print("errors")
print(len(testResult.errors))
print("failures")
print(len(testResult.failures))
print("skipped")
print(len(testResult.skipped))
print("testsRun")
print(testResult.testsRun)

```

Работа примера:

```

....ss
-----
Ran 6 tests in 0.000s

OK (skipped=2)
errors
0
failures
0
skipped
2
testsRun
6

Process finished with exit code 0

```

Индивидуальное задание:

```

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

import sys
from datetime import datetime
import sqlite3

DB_PATH = "./data.db"

def create_db():
    con = sqlite3.connect(DB_PATH)
    cur = con.cursor()

```

```

cur.execute(
    '''
    CREATE TABLE IF NOT EXISTS types(
        type_id INTEGER PRIMARY KEY AUTOINCREMENT,
        train_type TEXT NOT NULL
    )
    '''
)

cur.execute(
    '''
    CREATE TABLE IF NOT EXISTS departures(
        departure_id INTEGER PRIMARY KEY AUTOINCREMENT,
        train_number INTEGER NOT NULL,
        destination TEXT NOT NULL,
        type_id INTEGER NOT NULL,
        time DATE,
        FOREIGN KEY(type_id) REFERENCES types(type_id)
    )
    '''
)

con.commit()
con.close()

def select_all(db_path=DB_PATH):
    pass
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()
    cursor.execute(
        """
        SELECT
            departures.train_number,
            types.train_type,
            departures.destination,
            departures.time
        FROM departures
        INNER JOIN types ON types.type_id = departures.type_id
        """
    )
    rows = cursor.fetchall()
    conn.commit()
    conn.close()

    return [
        {
            'destination': row[2],
            'number': row[0],
            'train_type': row[1],
            'time': datetime.strptime(row[3], '%H:%M'),
        }
        for row in rows
    ]

def add():
    destination = input("Название пункта назначения? ")
    number = int(input("Номер поезда? "))
    time = input("Время отправления ЧЧ:ММ? ")
    train_type = input("Тип поезда? ")
    time = datetime.strptime(time, '%H:%M')

    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()

```

```

        cursor.execute(
            """
            SELECT type_id FROM types WHERE train_type = ?
            """,
            (train_type,)
        )
        row = cursor.fetchone()

        if row is None:
            cursor.execute(
                """
                INSERT INTO types (train_type) VALUES (?)
                """,
                (train_type,)
            )
            type_id = cursor.lastrowid
        else:
            type_id = row[0]

        cursor.execute(
            """
            INSERT INTO departures (train_number, destination, type_id, time)
            VALUES (?, ?, ?, ?)
            """,
            (number, destination, type_id, time.strftime("%H:%M"))
        )

        conn.commit()
        conn.close()

def select(command, db_path=DB_PATH):
    count = 0
    parts = command.split(' ', maxsplit=1)
    time = datetime.strptime(parts[1], '%H:%M')

    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()

    cursor.execute(
        """
        SELECT
            departures.train_number,
            types.train_type,
            departures.destination,
            departures.time
        FROM departures
        INNER JOIN types ON types.type_id = departures.type_id
        WHERE departures.time >= ?
        """,
        (time.strftime("%H:%M"),)
    )
    rows = cursor.fetchall()
    conn.close()

    trains = [
        {
            'destination': row[2],
            'number': row[0],
            'train_type': row[1],
            'time': time,
        }
        for row in rows
    ]

```

```

]

if len(trains) == 0:
    print("Отправлений позже этого времени нет.")
else:
    print_list(trains)
return trains

def main():
    create_db()
    while True:
        command = get_command()
        if command == 'exit':
            break

        elif command == 'add':
            add()

        elif command == 'list':
            print_list(select_all())

        elif command.startswith('select'):
            select(command)

        elif command == 'help':
            print_help()

        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)

def get_command():
    return input(">>> ").lower()

def print_list(trains):
    line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 28,
        '-' * 14,
        '-' * 14,
        '-' * 19
    )
    print(line)
    print(
        '| {:^4} | {:^28} | {:^14} | {:^14} | {:^19} |'.format(
            "No",
            "Название пункта назначения",
            "Номер поезда",
            "Тип поезда",
            "Время отправления"
        )
    )
    print(line)
    for idx, train in enumerate(trains, 1):
        print(
            '| {:>4} | {:<28} | {:^14} | {:<14} | {:>19} |'.format(
                idx,
                train.get('destination', ''),
                train.get('number', ''),
                train.get('train_type', ''),
                train.get('time', 0).strftime("%H:%M")
            )
        )

```

```

    )
    print(line)

def print_help():
    print("Список команд:\n")
    print("add - добавить отправление;")
    print("list - вывести список отправлений;")
    print("select <ЧЧ:ММ> - вывод на экран информации о "
          "поездах, отправляющихся после этого времени;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

if __name__ == '__main__':
    main()

```

```

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

import unittest
import main
from datetime import datetime
import sqlite3
import os
from random import choice, randint
from string import ascii_lowercase, digits, ascii_uppercase

class TrainTests(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        """Set up for class"""
        print("Настройка класса")
        print("=====")

    @classmethod
    def tearDownClass(cls):
        """Tear down for class"""
        print("=====")
        print("Снос класса")

    def setUp(self):
        """Set up for test"""
        print("Настройка для [" + self.shortDescription() + "]")
        print("Создание тестовой базы данных...")

    def tearDown(self):
        """Tear down for test"""
        print("Снос для [" + self.shortDescription() + "]")
        os.remove('test.db')
        print("Тестовая база данных была удалена")

    def test_select_all(self):
        """The whole selection test"""
        con = sqlite3.connect('test.db')
        cur = con.cursor()

        cur.execute(
            '''

```



```

        CREATE TABLE IF NOT EXISTS types(
            type_id INTEGER PRIMARY KEY AUTOINCREMENT,
            train_type TEXT NOT NULL
        )
        '''
    )

    cur.execute(
        '''
        CREATE TABLE IF NOT EXISTS departures(
            departure_id INTEGER PRIMARY KEY AUTOINCREMENT,
            train_number INTEGER NOT NULL,
            destination TEXT NOT NULL,
            type_id INTEGER NOT NULL,
            time DATE,
            FOREIGN KEY(type_id) REFERENCES types(type_id)
        )
        '''
    )

    db_list = []
    num_of_records = randint(2, 10)

    print(f"Количество записей: {num_of_records}")

    for _ in range(num_of_records):
        train_types = ('Cargo', 'Passenger', 'Construction')

        #####
        letters = ascii_lowercase
        length = randint(1, 15)

        destination = ''.join(choice(letters) for i in range(length))

        number = randint(1, 100)

        train_type = choice(train_types)

        time = ''.join(str(randint(10, 23)) +
                        ":" + str(randint(10, 59)))

        ans = {
            'destination': destination,
            'number': number,
            'train_type': train_type,
            'time': datetime.strptime(time, '%H:%M'),
        }

        print(ans)
        db_list.append(ans)

    cursor = con.cursor()

    cursor.execute(
        """
        SELECT type_id FROM types WHERE train_type = ?
        """,
        (train_type,)
    )
    row = cursor.fetchone()

    if row is None:
        cursor.execute(
            """
            INSERT INTO types (train_type) VALUES (?)
            """

```

```

        """
        (train_type,)
    )
    type_id = cursor.lastrowid
else:
    type_id = row[0]

    cursor.execute(
        """
        INSERT INTO departures (train_number, destination, type_id,
time)
        VALUES (?, ?, ?, ?)
        """
        (number, destination, type_id, time)
    )

    con.commit()

self.assertEqual(main.select_all('test.db'), db_list)
con.close()

def test_select_by_type(self):
    """Selection test"""
    conn = sqlite3.connect('test.db')
    cur = conn.cursor()

    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS types(
            type_id INTEGER PRIMARY KEY AUTOINCREMENT,
            train_type TEXT NOT NULL
        )
        """
    )

    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS departures(
            departure_id INTEGER PRIMARY KEY AUTOINCREMENT,
            train_number INTEGER NOT NULL,
            destination TEXT NOT NULL,
            type_id INTEGER NOT NULL,
            time DATE,
            FOREIGN KEY(type_id) REFERENCES types(type_id)
        )
        """
    )

    train_types = ('Cargo', 'Passenger', 'Construction')
    #####
    letters = ascii_lowercase
    length = randint(1, 15)

    destination = ''.join(choice(letters) for i in range(length))

    number = randint(1, 100)

    train_type = choice(train_types)

    time = ''.join(str(randint(10, 23)) +
                    ":" + str(randint(10, 59)))

    cursor = conn.cursor()

```

```

        cursor.execute(
            """
            SELECT type_id FROM types WHERE train_type = ?
            """,
            (train_type,)
        )
        row = cursor.fetchone()

        if row is None:
            cursor.execute(
                """
                INSERT INTO types (train_type) VALUES (?)
                """,
                (train_type,)
            )
            type_id = cursor.lastrowid
        else:
            type_id = row[0]

        cursor.execute(
            """
            INSERT INTO departures (train_number, destination, type_id, time)
            VALUES (?, ?, ?, ?)
            """,
            (number, destination, type_id, time)
        )
        conn.commit()

        ans = [
            {
                'destination': destination,
                'number': number,
                'train_type': train_type,
                'time': datetime.strptime(time, '%H:%M'),
            }
        ]
        print(ans)
        test_data = 'select ' + ' '.join(str(time))
        self.assertEqual(
            main.select(test_data, 'test.db'), ans
        )
        conn.close()

```

```

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

import unittest
import main_test

testLoad = unittest.TestLoader()
suites = testLoad.loadTestsFromModule(main_test)
testResult = unittest.TestResult()
runner = unittest.TextTestRunner(verbosity=1)
testResult = runner.run(suites)

print("errors")
print(len(testResult.errors))
print("failures")
print(len(testResult.failures))
print("skipped")

```

```
print(len(testResult.skipped))
print("testsRun")
print(testResult.testsRun)
```

Пример работы индивидуального задания:

```
✓ Tests passed: 2 of 2 tests - 1sec 12ms
Launching unittests with arguments python -m unittest F:/pythonProject/lab_2_10/task/main_test.py in F:/pythonProject/lab_2_10/task

Настройка класса
=====
=====
Снос класса

Process finished with exit code 0
Настройка для [The whole selection test]
Создание тестовой базы данных...
Количество записей: 9
{'destination': 'yxmhdgurel', 'number': 21, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 17, 57)}
{'destination': 'bmvrkucmmooyfu', 'number': 92, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 18, 28)}
{'destination': 'llrglbyqfeey', 'number': 24, 'train_type': 'Construction', 'time': datetime.datetime(1900, 1, 1, 17, 10)}
{'destination': 'rhbotsuxv', 'number': 76, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 16, 59)}
{'destination': 'ombrhrhrgaim', 'number': 18, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 22, 56)}
{'destination': 'lgwdlhgmrvvxhyrm', 'number': 59, 'train_type': 'Construction', 'time': datetime.datetime(1900, 1, 1, 10, 23)}
{'destination': 'wjapfvhn', 'number': 70, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 20, 47)}
{'destination': 'hsufyma', 'number': 19, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 21, 21)}
{'destination': 'orazlyu', 'number': 49, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 17, 16)}
Снос для [The whole selection test]
Тестовая база данных была удалена
Настройка для [Selection test]
Создание тестовой базы данных...

Ran 2 tests in 1.013s

OK
[{'destination': 'pgarovvysumeb', 'number': 84, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 16, 13)}]
+-----+
| No | Название пункта назначения | Номер поезда | Тип поезда | Время отправления |
+-----+
| 1 | pgarovvysumeb | 84 | Cargo | 16:13 |
+-----+
Снос для [Selection test]
Тестовая база данных была удалена
```

Ответы на вопросы:

### 1. Для чего используется автономное тестирование?

Для тестирования функций, классов, методов и т.д. с целью выявления ошибок в работе в этих отдельных единицах общей программы.

### 2. Какие фреймворки Python получили наибольшее распространение для решения задач автономного тестирования?

- unittest
- nose
- pytest

### 3. Какие существуют основные структурные единицы модуля unittest?

Test fixture – обеспечивает подготовку окружения для выполнения тестов, а также организацию мероприятий по их корректному завершению (например, очистка ресурсов).

Test case – это элементарная единица тестирования, в рамках которой проверяется работа компонента тестируемой программы (метод, класс, поведение и т. п.).

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).

Test runner – это компонент, которые оркестрирует (координирует взаимодействие) запуск тестов и предоставляет пользователю результат их выполнения.

#### 4. Какие существуют способы запуска тестов unittest?

Запуск тестов можно сделать как из командной строки, так и с помощью графического интерфейса пользователя.

#### 5. Каково назначение класса 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.

tearDown() Метод вызывается после завершения работы теста.

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

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

Метод	Описание
<code>assertRaises(exc, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> вызывает исключение <code>exc</code>
<code>assertRaisesRegex(exc, r, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> вызывает исключение <code>exc</code> , сообщение которого совпадает с регулярным выражением <code>r</code>
<code>assertWarns(warn, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> выдает сообщение <code>warn</code>
<code>assertWarnsRegex(warn, r, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> выдает сообщение <code>warn</code> и оно совпадает с регулярным выражением <code>r</code>

Метод	Описание
<code>assertAlmostEqual(a, b)</code>	<code>round(a-b, 7) == 0</code>
<code>assertNotAlmostEqual(a, b)</code>	<code>round(a-b, 7) != 0</code>
<code>assertGreater(a, b)</code>	<code>a &gt; b</code>
<code>assertGreaterEqual(a, b)</code>	<code>a &gt;= b</code>
<code>assertLess(a, b)</code>	<code>a &lt; b</code>
<code>assertLessEqual(a, b)</code>	<code>a &lt;= b</code>
<code>assertRegex(s, r)</code>	<code>r.search(s)</code>
<code>assertNotRegex(s, r)</code>	<code>not r.search(s)</code>
<code>assertCountEqual(a, b)</code>	a и b имеют одинаковые элементы (порядок неважен)

Метод	Описание
<code>assertMultiLineEqual(a, b)</code>	строки (strings)
<code>assertSequenceEqual(a, b)</code>	последовательности (sequences)
<code>assertListEqual(a, b)</code>	списки (lists)
<code>assertTupleEqual(a, b)</code>	кортежи (tuples)
<code>assertSetEqual(a, b)</code>	множества или неизменяемые множества (frozensets)
<code>assertDictEqual(a, b)</code>	словари (dicts)

## 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.

`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?

Во избежание ошибок тестирования, так как некоторые тесты могут давать заведомо неправильный результат в зависимости от какого-либо условия. Для этого такие тесты необходимо пропускать.

## 12. Как выполняется безусловный и условных пропуск тестов? Как выполнить пропуск класса тестов?

Безусловный пропуск: `@unittest.skip(reason)` записывается перед объявлением теста.

Условный пропуск:

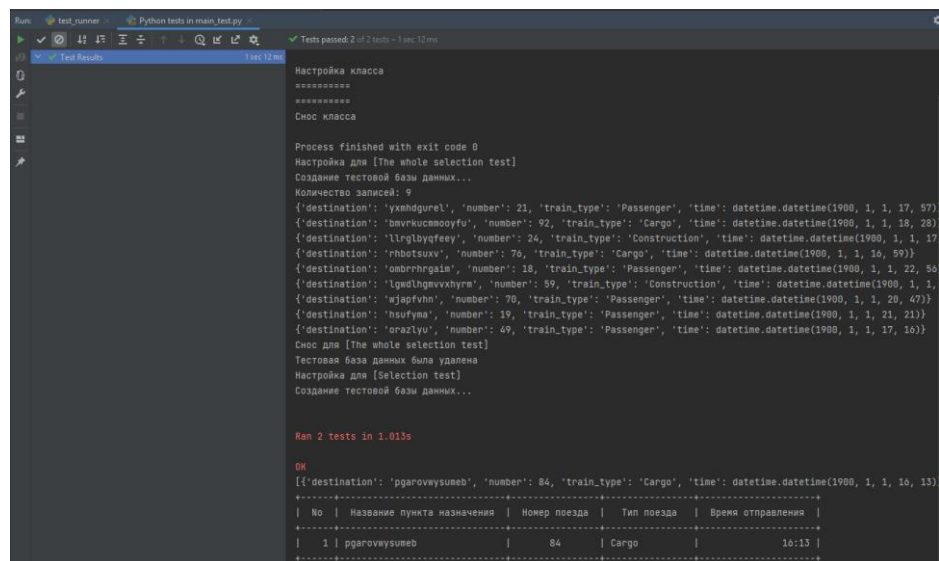
1) `@unittest.skipIf(condition, reason)` – Тест будет пропущен, если условие (`condition`) истинно.

2) @unittest.skipUnless(condition, reason) – Тест будет пропущен если, условие (condition) не истинно.

Пропуск класса тестов: @unittest.skip(reason) записывается перед объявлением класса.

**13. Самостоятельно изучить средства по поддержке тестов unittest в PyCharm. Приведите обобщенный алгоритм проведения тестирования с помощью PyCharm.**

1. Создать класс для тестирования
2. Написать код тестов
3. Запустить тест



```
Run: test_runner Python tests in main_test.py Tests passed: 2 of 2 tests - 1 sec 12 ms
Test Results 1 sec 12 ms

Настройка класса
=====
Снос класса

Process finished with exit code 0
Настройка для [The whole selection test]
Создание тестовой базы данных...
Количество записей: 9
{'destination': 'yxmhdgurel', 'number': 21, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 17, 57)}
{'destination': 'bmvrkucsmooyfu', 'number': 92, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 18, 28)}
{'destination': 'llegloyfeey', 'number': 24, 'train_type': 'Construction', 'time': datetime.datetime(1900, 1, 1, 17, 59)}
{'destination': 'nmbotuuu', 'number': 78, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 18, 59)}
{'destination': 'ombrhrvgain', 'number': 15, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 22, 54)}
{'destination': 'lqsdhgwvkhyme', 'number': 89, 'train_type': 'Construction', 'time': datetime.datetime(1900, 1, 1, 1, 1)}
{'destination': 'xjaprvhn', 'number': 70, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 20, 47)}
{'destination': 'hsufyma', 'number': 39, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 21, 21)}
{'destination': 'orazlyu', 'number': 49, 'train_type': 'Passenger', 'time': datetime.datetime(1900, 1, 1, 17, 16)}
Снос для [The whole selection test]
Тестовая база данных была удалена
Настройка для [Selection test]
Создание тестовой базы данных...

Ran 2 tests in 1.013s

OK
[{'destination': 'pgarovysuseb', 'number': 84, 'train_type': 'Cargo', 'time': datetime.datetime(1900, 1, 1, 16, 13)}]
+-----+
| No | Название пункта назначения | Номер поезда | Тип поезда | Время отправления |
+-----+
| 1 | pgarovysuseb | 84 | Cargo | 16:13 |
+-----+
```

Окно тестирования