

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет по рубежному контролю №2

Выполнил:
студент группы ИУ5-31Б
Абуховский Иван
Александрович

Подпись: _____

Дата: _____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись: _____

Дата: _____

Москва, 2021 г.

Рубежный контроль №2

Задание

- 1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
- 2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD – фреймворка (3 теста).

Текст программы

AbukhovskiyRK1IU5_31B.py

```
from operator import itemgetter

class Student:
    """Студент"""
    def __init__(self, id, surname, dolgi, group_id):
        self.id = id
        self.surname = surname
        self.dolgi = dolgi
        self.group_id = group_id
```

```
class Group:
    """Группа"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

```
class StudentGroup:
    """
    'Студенты группы' для реализации
    связи многие-ко-многим
    """
    def __init__(self, group_id, student_id):
        self.group_id = group_id
        self.student_id = student_id
```

```
# Группы
groups = [
    Group(1, 'ИУ5-31Б'),
    Group(2, 'ИУ6-12Б'),
    Group(3, 'ИУ7-53'),

    Group(11, 'МТ5-11'),
    Group(22, 'РК5-32Б'),
    Group(33, 'СМ5-13'),
]
```

```
# Студенты
students = [
    Student(1, 'Абуховский', 25, 1),
    Student(2, 'Черноморец', 47, 2),
    Student(3, 'Шагиахметов', 39, 3),
    Student(4, 'Стельмах', 81, 4),
    Student(5, 'Сыса', 17, 5),
]
```

```
students_groups = [
    StudentGroup(1,1),
```

```

StudentGroup(2,2),
StudentGroup(2,3),
StudentGroup(3,4),
StudentGroup(3,5),

StudentGroup(11,1),
StudentGroup(22,2),
StudentGroup(22,3),
StudentGroup(33,4),
StudentGroup(33,5),
]

def task_a1(one_to_many):
    one_to_many = one_to_many
    return [i for i in sorted(one_to_many, key=itemgetter(2))]

def task_a2(one_to_many):
    res_2_unsorted = []
    # Перебираем все группы
    for g in groups:
        # Список студентов группы
        g_students = list(filter(lambda i: i[2] == g.name, one_to_many))
        # Если класс не пустой
        if len(g_students) > 0:
            # Долги за обучение студентов группы
            g_dolgii = [dolgi for _, dolgi, _ in g_students]
            # Суммарный долг студентов группы
            g_dolgii_sum = sum(g_dolgii)
            res_2_unsorted.append((g.name, g_dolgii_sum))

    # Сортировка по суммарному долгу
    res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
    return res_2

def task_a3(many_to_many):
    res_3 = {}
    # Перебираем все группы
    for g in groups:
        if 'Б' in g.name:
            # Список студентов группы
            g_students = list(filter(lambda i: i[2] == g.name, many_to_many))
            # Только фамилии студентов
            g_students_surnames = [x for x, _, _ in g_students]
            # Добавляем результат в словарь
            # ключ - группа, значение - список фамилий
            res_3[g.name] = g_students_surnames
    return res_3

def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(s.surname, s.dolgi, g.name)
                    for g in groups
                    for s in students
                    if s.group_id==g.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(g.name, sg.group_id, sg.student_id)
                          for g in groups
                          for sg in students_groups
                          if g.id==sg.group_id]

```

```

many_to_many = [(s.surname, s.dolgi, group_name)
                 for group_name, group_id, student_id in many_to_many_temp
                 for s in students if s.id==student_id]

print('Задание A1')
print(task_a1(one_to_many))

print('\nЗадание A2')
print(task_a2(one_to_many))

print('\nЗадание A3')
print(task_a3(many_to_many))

if __name__ == '__main__':
    main()

```

test.py

```
from AbukhovskiyRK1IU5_31B import *
import unittest

class Test(unittest.TestCase):

    def setUp(self):
        self.groups = [
            Group(1, 'ИУ5-31Б'),
            Group(2, 'ИУ6-12Б'),
            Group(3, 'ИУ7-53'),

            Group(11, 'МТ5-11'),
            Group(22, 'РК5-32Б'),
            Group(33, 'СМ5-13'),
        ]
        self.students = [
            Student(1, 'Абуховский', 25, 1),
            Student(2, 'Черноморец', 47, 2),
            Student(3, 'Шагиахметов', 39, 3),
            Student(4, 'Стельмах', 81, 4),
            Student(5, 'Сыса', 17, 5),
        ]
        self.students_groups = [
            StudentGroup(1,1),
            StudentGroup(2,2),
            StudentGroup(2,3),
            StudentGroup(3,4),
            StudentGroup(3,5),

            StudentGroup(11,1),
            StudentGroup(22,2),
            StudentGroup(22,3),
            StudentGroup(33,4),
            StudentGroup(33,5),
        ]

        self.one_to_many = [(s.surname, s.dolgi, g.name)
                             for g in groups
                             for s in students
                             if s.group_id==g.id]

        self.many_to_many_temp = [(g.name, sg.group_id, sg.student_id)
                                   for g in groups
                                   for sg in students_groups
                                   if g.id==sg.group_id]

        self.many_to_many = [(s.surname, s.dolgi, group_name)
                              for group_name, group_id, student_id in many_to_many_temp
                              for s in students if s.id==student_id]

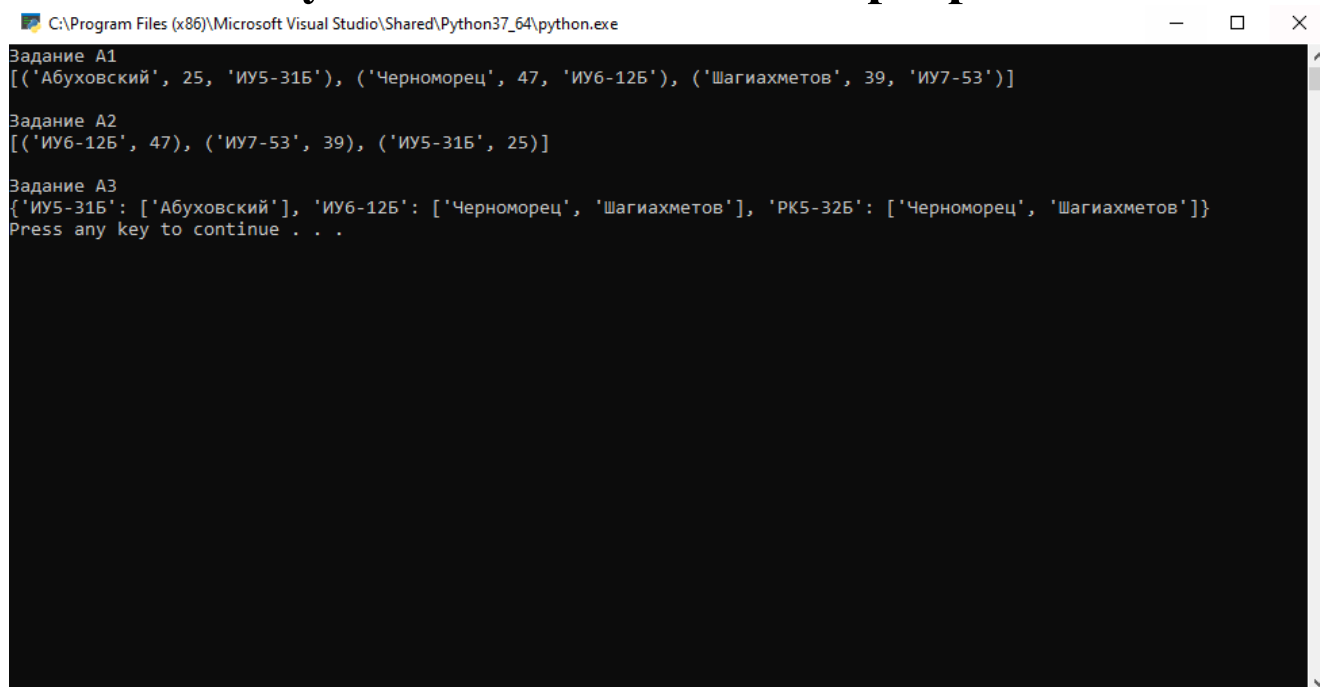
    def test_task_a1(self):
        prediction = [('Абуховский', 25, 'ИУ5-31Б'), ('Черноморец', 47, 'ИУ6-12Б'),
                      ('Шагиахметов', 39, 'ИУ7-53')]
        self.assertEqual(task_a1(self.one_to_many), prediction)

    def test_task_a2(self):
        prediction = [('ИУ6-12Б', 47), ('ИУ7-53', 39), ('ИУ5-31Б', 25)]
        self.assertEqual(task_a2(self.one_to_many), prediction)

    def test_task_a3(self):
```

```
        prediction = {'ИУ5-31Б': ['Абуховский'], 'ИУ6-12Б': ['Черноморец', 'Шагиахметов'],  
                      'РК5-32Б': ['Черноморец', 'Шагиахметов']}  
        self.assertEqual(task_a3(self.many_to_many), prediction)  
  
if __name__ == '__main__':  
    unittest.main()
```

Результаты выполнения программы



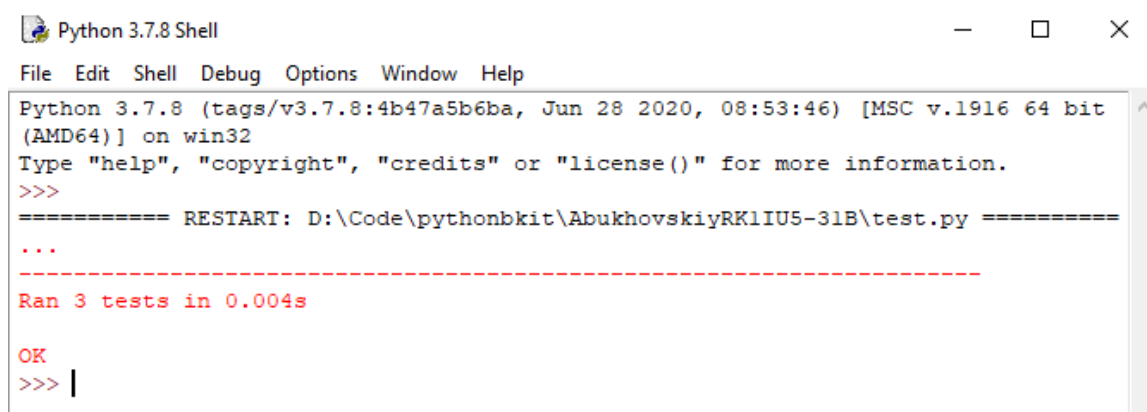
```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe

Задание A1
[('Абуховский', 25, 'ИУ5-31Б'), ('Черноморец', 47, 'ИУ6-12Б'), ('Шагиахметов', 39, 'ИУ7-53')]

Задание A2
[('ИУ6-12Б', 47), ('ИУ7-53', 39), ('ИУ5-31Б', 25)]

Задание A3
{'ИУ5-31Б': ['Абуховский'], 'ИУ6-12Б': ['Черноморец', 'Шагиахметов'], 'РК5-32Б': ['Черноморец', 'Шагиахметов']}
Press any key to continue . . .
```

Программа была отредактирована так, чтобы результаты ее выполнения не изменились, но ее можно было модульно её протестировать.



```
Python 3.7.8 Shell

File Edit Shell Debug Options Window Help

Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Code\pythonbkit\AbukhovskiyRK1IU5-31B\test.py =====
...
-----
Ran 3 tests in 0.004s

OK
>>> |
```

(Тесты запускал в Python Shell, а не в Visual Studio, для удобства просмотра результатов и редактирования).

Тестирование показывает, что программа работает исправно.

Если поменять ожидаемый результат в тестах, то результат тестирования укажет на несоответствие полученного результата ожидаемому.

```
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
===== RESTART: D:\Code\pythonbkit\AbukhovskiyRK1IU5-31B\test.py =====
FF.
=====
FAIL: test_task_a1 (__main__.Test)
-----
Traceback (most recent call last):
  File "D:\Code\pythonbkit\AbukhovskiyRK1IU5-31B\test.py", line 54, in test_task_a1
    self.assertEqual(task_a1(self.one_to_many), prediction)
AssertionError: Lists differ: [('Абуховский', 25, 'ИУ5-31Б'), ('Черноморец[44 chars]53')] != [('Шагиахметов', 25, 'ИУ5-31Б'), ('Черноморец[44 chars]53')]

First differing element 0:
('Абуховский', 25, 'ИУ5-31Б')
('Шагиахметов', 25, 'ИУ5-31Б')

- [('Абуховский', 25, 'ИУ5-31Б'),
?   ^^^  ----

+ [('Шагиахметов', 25, 'ИУ5-31Б'),
?   ^^^^^ +++

      ('Черноморец', 47, 'ИУ6-12Б'),
- ('Шагиахметов', 39, 'ИУ7-53')]
?   ^^^^^ ---

+ ('Абуховский', 39, 'ИУ7-53')]
?   ^^^  +++

=====
FAIL: test_task_a2 (__main__.Test)
-----
Traceback (most recent call last):
  File "D:\Code\pythonbkit\AbukhovskiyRK1IU5-31B\test.py", line 58, in test_task_a2
    self.assertEqual(task_a2(self.one_to_many), prediction)
AssertionError: Lists differ: [('ИУ6-12Б', 47), ('ИУ7-53', 39), ('ИУ5-31Б', 25)] != [('ИУ5-34Б', 47), ('ИУ7-53', 39), ('ИУ6-12Б', 25)]

First differing element 0:
('ИУ6-12Б', 47)
('ИУ5-34Б', 47)

- [('ИУ6-12Б', 47), ('ИУ7-53', 39), ('ИУ5-31Б', 25)]
+ [('ИУ5-34Б', 47), ('ИУ7-53', 39), ('ИУ6-12Б', 25)]

-----
Ran 3 tests in 0.005s

FAILED (failures=2)
>>> |
```

Ln: 58 Col: 4

