Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ Кафедра ИУ5

Курс «Основы информатики» Отчет по Рубежному контролю №2

Выполнил студент группы ИУ5-33Б: Емельянов А.К. Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е. Подпись и дата:

Москва, 2024 г

Code refactored

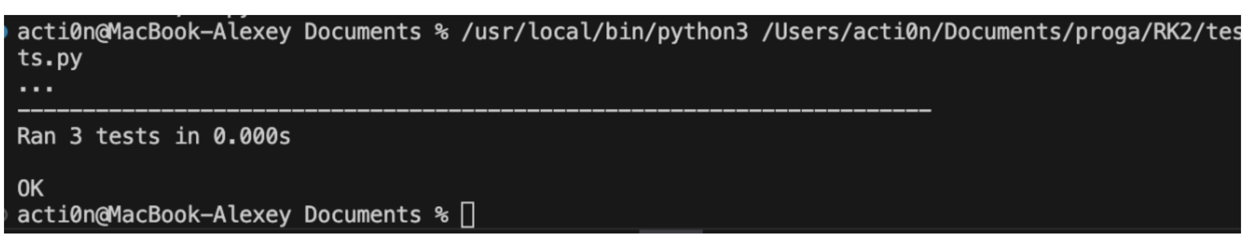
```
class Student:
    def __init__(self, student_id, surname, scholarship, group_id):
        self.student_id = student_id
        self.surname = surname
        self.scholarship = scholarship
        self.group_id = group_id

class Group:
    def __init__(self, group_id, group_name):
        self.group_id = group_id
        self.group_name = group_name

class GroupStudent:
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id

# Данные
students = [
    Student(student_id=1, surname="Андреев", scholarship=1500, group_id=1),
    Student(student_id=2, surname="Борисов", scholarship=1200, group_id=1),
    Student(student_id=3, surname="Алексеева", scholarship=1800, group_id=2),
    Student(student_id=4, surname="Аркадьев", scholarship=1600, group_id=3),
    Student(student_id=5, surname="Васильева", scholarship=1100, group_id=2)
]

groups = [
    Group(group_id=1, group_name="Группа А"),
    Group(group_id=2, group_name="Группа Б"),
    Group(group_id=3, group_name="Группа В")
]

group_students = [
    GroupStudent(student_id=1, group_id=1),
    GroupStudent(student_id=2, group_id=1),
    GroupStudent(student_id=3, group_id=2),
    GroupStudent(student_id=4, group_id=3),
    GroupStudent(student_id=5, group_id=2)
]

# Функции
def task_1(students, groups):

    return [
        (student.surname, next(group.group_name for group in groups if group.group_id == student.group_id))
        for student in students if student.surname.startswith("А")
    ]

def task_2(students, groups):
       group_min_scholarships = {
        group.group_name: min(student.scholarship for student in students if student.group_id == group.group_id)
        for group in groups
    }
    return sorted(group_min_scholarships.items(), key=lambda x: x[1])

def task_3(students, groups, group_students):

    return sorted([
        (next(student.surname for student in students if student.student_id == gs.student_id),
         next(group.group_name for group in groups if group.group_id == gs.group_id))
        for gs in group_students
    ], key=lambda x: x[0])

# Основной код
if __name__ == "__main__":
    print("Task#1")
    for a in task_1(students, groups):
        print(a)

    print("Task#2")
    for a in task_2(students, groups):
        print(a)


    print("Task#3")
    for a in task_3(students, groups, group_students):
        print(a)
```

Testing

```
import unittest
# from refactored_main import *
# from refactored_main import task_1, task_2, task_3, students, groups, group_students
class Student:
    def __init__(self, student_id, surname, scholarship, group_id):
        self.student_id = student_id
        self.surname = surname
        self.scholarship = scholarship
        self.group_id = group_id


class Group:
    def __init__(self, group_id, group_name):
        self.group_id = group_id
        self.group_name = group_name


class GroupStudent:
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id
```

```python
students = [
    Student(student_id=1, surname="Андреев", scholarship=1500, group_id=1),
    Student(student_id=2, surname="Борисов", scholarship=1200, group_id=1),
    Student(student_id=3, surname="Алексеева", scholarship=1800, group_id=2),
    Student(student_id=4, surname="Аркадьев", scholarship=1600, group_id=3),
    Student(student_id=5, surname="Васильева", scholarship=1100, group_id=2)
]

groups = [
    Group(group_id=1, group_name="Группа А"),
    Group(group_id=2, group_name="Группа Б"),
    Group(group_id=3, group_name="Группа В")
]

group_students = [
    GroupStudent(student_id=1, group_id=1),
    GroupStudent(student_id=2, group_id=1),
    GroupStudent(student_id=3, group_id=2),
    GroupStudent(student_id=4, group_id=3),
    GroupStudent(student_id=5, group_id=2)
]

def task_1(students, groups):

    return [
        (student.surname, next(group.group_name for group in groups if group.group_id == student.group_id))
        for student in students if student.surname.startswith("А")
    ]

def task_2(students, groups):

    group_min_scholarships = {
        group.group_name: min(student.scholarship for student in students if student.group_id == group.group_id)
        for group in groups
    }
    return sorted(group_min_scholarships.items(), key=lambda x: x[1])

def task_3(students, groups, group_students):
        return sorted([
        (next(student.surname for student in students if student.student_id == gs.student_id),
         next(group.group_name for group in groups if group.group_id == gs.group_id))
        for gs in group_students
    ], key=lambda x: x[0])

class TestTasks(unittest.TestCase):

    def test_task_1(self):
        expected = [("Андреев", "Группа А"), ("Аркадьев", "Группа В"), ("Алексеева", "Группа Б")]
        self.assertEqual(task_1(students, groups), expected)

    def test_task_2(self):
        expected = [("Группа Б", 1100), ("Группа А", 1200), ("Группа В", 1600)]
        self.assertEqual(task_2(students, groups), expected)

    def test_task_3(self):
        """Тест для функции task_3"""
        expected = [
            ("Алексеева", "Группа Б"),
            ("Андреев", "Группа А"),
            ("Аркадьев", "Группа В"),
            ("Борисов", "Группа А"),
            ("Васильева", "Группа Б")
        ]
        self.assertEqual(task_3(students, groups, group_students), expected)

if __name__ == "__main__":
    unittest.main()
```

Result



```
acti0n@MacBook-Alexey Documents % /usr/local/bin/python3 /Users/acti0n/Documents/proga/RK2/tes
ts.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
acti0n@MacBook-Alexey Documents % 
```