



**Министерство науки и высшего образования Российской  
Федерации Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

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

**Рубежный контроль №2  
по дисциплине «Базовые компоненты интернет-технологий»**

**Выполнила:  
студентка группы ИУ5-33Б  
Пересыпкина М.А.**

**Проверил:  
Гапанюк Ю.Е.**

**2021 г.**

## Постановка задания:

Рубежный контроль представляет собой разработку тестов на языке Python.

- 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:

RK2.py

```
# используется для сортировки
from operator import itemgetter

class Prog:
    """Программа"""

    def __init__(self, id, nameProg, memory, comp_id):
        self.id = id
        self.nameProg = nameProg
        self.memory = memory
        self.comp_id = comp_id

class Comp:
    """Компьютер"""

    def __init__(self, id, name):
        self.id = id
        self.name = name

class ProgComp:
    """
    'Программы компьютера' для реализации
    связи многие-ко-многим
    """

    def __init__(self, comp_id, prog_id):
        self.comp_id = comp_id
        self.prog_id = prog_id

# Компьютеры
comps = [
    Comp(1, 'Asus'),
    Comp(2, 'Acer'),
    Comp(3, 'Xiaomi'),
    Comp(4, 'HP'),
    Comp(5, 'Apple'),
    Comp(6, 'Lenovo'),
]
```

```

# Программы
progs = [
    Prog(1, 'Adobe Premiere Pro', 1150, 1),
    Prog(2, 'Sony Vegas Pro', 970, 2),
    Prog(3, 'Discord', 200, 2),
    Prog(4, 'Figma', 450, 3),
    Prog(5, 'PyCharm Community', 580, 4),
]

# Программы и компьютеры для связи многие-ко-многим
progs_comps = [
    ProgComp(1, 1),
    ProgComp(2, 2),
    ProgComp(2, 3),
    ProgComp(3, 4),
    ProgComp(4, 5),

    ProgComp(5, 1),
    ProgComp(5, 2),
    ProgComp(5, 3),
    ProgComp(6, 4),
    ProgComp(6, 5),
]

def first_task(one_to_many):
    task1 = []
    for nameProg, memory, name in one_to_many:
        if nameProg[0] == "A":
            task1.append((nameProg, name))
    return task1

def second_task(one_to_many):
    task2_uns = []
    for c in comps:
        # все программы компьютера
        p_cls = list(filter(lambda i: i[2] == c.name, one_to_many))
        if len(p_cls) > 0:
            c_memory = [memory for _, memory, _ in p_cls]
            c_minMemory = min(c_memory)
            task2_uns.append((c.name, c_minMemory))
    task2 = sorted(task2_uns, key=itemgetter(1))
    return task2

def third_task(many_to_many):
    task3_uns = []
    for nameProg, memory, name in many_to_many:
        task3_uns.append((nameProg, name))

    task3 = list(sorted(task3_uns, key=itemgetter(0)))
    return task3

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

    # Соединение данных один-ко-многим
    one_to_many = [(p.nameProg, p.memory, c.name)
                    for c in comps
                    for p in progs
                    if p.comp_id == c.id]

```

```

# Соединение данных многие-ко-многим
many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
                      for c in comps
                      for pc in progs_comps
                      if c.id == pc.comp_id]

many_to_many = [(p.nameProg, p.memory, comp_name)
                 for comp_name, comp_id, prog_id in many_to_many_temp
                 for p in progs if p.id == prog_id]

print('\nЗадание 1 \n', first_task(one_to_many))
print('\nЗадание 2 \n', second_task(one_to_many))
print('\nЗадание 3 \n', third_task(many_to_many))

if __name__ == '__main__':
    main()

```

## test.py

```

import unittest
import sys, os

sys.path.append(os.getcwd())
from RK2 import *

class TddTest(unittest.TestCase):
    def test_first_task(self):
        one_to_many = [(p.nameProg, p.memory, c.name)
                       for c in comps
                       for p in progs
                       if p.comp_id == c.id]
        self.assertEqual(first_task(one_to_many), [('Adobe Premiere Pro',
'Asus')])

    def test_2(self):
        one_to_many = [(p.nameProg, p.memory, c.name)
                       for c in comps
                       for p in progs
                       if p.comp_id == c.id]
        self.assertEqual(second_task(one_to_many), [('Acer', 200), ('Xiaomi',
450), ('HP', 580), ('Asus', 1150)])

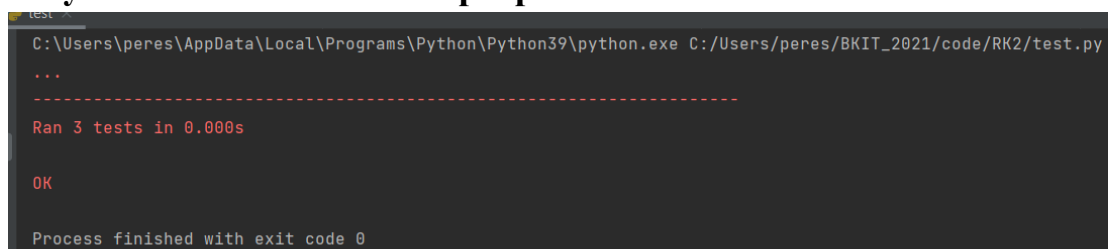
    def test_3(self):
        many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
                              for c in comps
                              for pc in progs_comps
                              if c.id == pc.comp_id]

        many_to_many = [(p.nameProg, p.memory, comp_name)
                         for comp_name, comp_id, prog_id in many_to_many_temp
                         for p in progs if p.id == prog_id]
        self.assertEqual(third_task(many_to_many),
                          [('Adobe Premiere Pro', 'Asus'), ('Adobe Premiere
Pro', 'Apple'), ('Discord', 'Acer'),
                          ('Discord', 'Apple'), ('Figma', 'Xiaomi'),
                          ('Figma', 'Lenovo'), ('PyCharm Community', 'HP'),
                          ('PyCharm Community', 'Lenovo'), ('Sony Vegas Pro',
'Acer'), ('Sony Vegas Pro', 'Apple')]
                          )

```

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

## Результаты выполнения программы:



The screenshot shows a terminal window with the following output:

```
test  
C:\Users\peres\AppData\Local\Programs\Python\Python39\python.exe C:/Users/peres/BKIT_2021/code/RK2/test.py  
...  
-----  
Ran 3 tests in 0.000s  
  
OK  
  
Process finished with exit code 0
```