



**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 «ГУИМЦ»

КАФЕДРА ИУ5 «Системы обработки информации и управления»

Дисциплина «Базовые компоненты ИТ»

ОТЧЁТ

Рубежный контроль № 2

Студент: Абрючнов Е.С. , группа ИУ5Ц-53Б

Преподаватель: Гапанюк Ю.Е.

2021 г.

Условия рубежного контроля №2 по курсу БКИТ

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).

main.py

```
from operator import itemgetter

class Program:
    """Язык программирования"""

    def __init__(self, id, named, popularity, dep_id):
        self.id = id
        self.named = named
        self.popularity = popularity
        self.dep_id = dep_id

class Syntaksis:
    """Синтаксисы"""

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

class ProgramSyntaksis:
    """
    'Синтаксисы' для реализации
    связи многие-ко-многим
    """

    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id

# Синтаксисы
deps = [
    Syntaksis(1, 'Конструкции'),
    Syntaksis(2, 'Функции'),
    Syntaksis(3, 'Операции'),
    Syntaksis(11, 'Конструкции (задания)'),
    Syntaksis(22, 'Функции (задания)'),
    Syntaksis(33, 'Операции (задания)'),
]

# Языки программирования
emps = [
    Program(1, 'Pascal', 0.3, 1),
```

```

        Program(2, 'C++', 4.5, 2),
        Program(3, 'Python', 12.1, 3),
        Program(4, 'JavaScript', 18.1, 3),
        Program(5, 'C#', 14.3, 3),
    ]

    emps_deps = [
        ProgramSyntaksis(1, 1),
        ProgramSyntaksis(2, 2),
        ProgramSyntaksis(3, 3),
        ProgramSyntaksis(3, 4),
        ProgramSyntaksis(3, 5),
        ProgramSyntaksis(11, 1),
        ProgramSyntaksis(22, 2),
        ProgramSyntaksis(33, 3),
        ProgramSyntaksis(33, 4),
    ]

    def sort_one(emps):
        return sorted(emps, key=itemgetter(2))

    def sort_two(emps, deps):
        res_12_unsorted = []
        for d in deps:
            # Список ЯП
            d_emps = list(filter(lambda i: i[2] == d.name, emps))
            # Если отдел не пустой
            if len(d_emps) > 0:
                # Популярность каждого ЯП
                d_popularity = [popular for _, popular, _ in d_emps]
                # Суммарная популярность ЯП
                d_popularity_sum = sum(d_popularity)
                res_12_unsorted.append((d.name, d_popularity_sum))
        return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)

    def sort_three(emps, deps):
        res_13 = {}
        for d in deps:
            if ' ' in d.name:
                # Список языков программирования
                d_emps = list(filter(lambda i: i[2] == d.name, emps))

                # Только названия ЯП
                d_emps_names = [x for x, _, _ in d_emps]
                # Добавляем результат в словарь

                # ключ - синтаксисы, значение - название ЯП
                res_13[d.name] = d_emps_names
        print(res_13)

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

        # Соединение данных один-ко-многим

        one_to_many = [(e.named, e.popularity, d.name)
                        for d in deps
                        for e in emps
                        if e.dep_id == d.id]

```

```

# Соединение данных многие-ко-многим

many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
                      for d in deps
                      for ed in emps_deps
                      if d.id == ed.dep_id]

many_to_many = [(e.named, e.popularity, dep_name)
                 for dep_name, dep_id, emp_id in many_to_many_temp
                 for e in emps if e.id == emp_id]

print('Задание B1\n', sort_one(one_to_many))
print('Задание B2\n', sort_two(one_to_many, deps))
print('Задание B3\n', sort_three(many_to_many, deps))

if __name__ == '__main__':
    main()

```

test.py

```

import unittest
import sys, os

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

one_to_many = [(e.named, e.popularity, d.name)
                for d in deps
                for e in emps
                if e.dep_id == d.id]

# Соединение данных многие-ко-многим

many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
                      for d in deps
                      for ed in emps_deps
                      if d.id == ed.dep_id]

many_to_many = [(e.named, e.popularity, dep_name)
                 for dep_name, dep_id, emp_id in many_to_many_temp
                 for e in emps if e.id == emp_id]

class Test(unittest.TestCase):
    def test_sort_one(self):
        self.assertEqual(sort_one(one_to_many), emps, {('Pascal', 0.3,
'Конструкции'), ('Python',
12.1, 'Операции'),
('JavaScript', 18.1,
'Операции'),
('C#', 14.3,
'Операции'),
('C++', 4.5,
'Функции')})

    def test_sort_two(self):
        self.assertEqual(sort_two(one_to_many), deps, deps), [('Операции',
44.5), ('Функции', 4.5),
('Конструкции', 0.3)]

    def test_sort_three(self):

```

```
        self.assertEqual(sort_three(many_to_many), deps, deps),
{'Конструкции': ['Pascal'], 'Функции':
['C++'],
 'Операции': ['Python', 'JavaScript', 'C#'],
 'Конструкции (задания)': ['Pascal'],
 'Функции (задания)': ['C++'],
 'Операции (задания)': ['Python', 'JavaScript']}]

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

Результат программы:

Ran 3 tests in 0.001s