Задание

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

RK_PCPL_Peresipko.py

```
from operator import itemgetter


class symphony:
    """Музыкальное произведение"""
    def __init__(self, id, name, mins, orch_id):
        self.id = id
        self.name = name
        self.mins = mins
        self.orch_id = orch_id


class orchestra:
    #"""Оркестр"""
    def __init__(self, id, name):
        self.id = id
        self.name = name


class SymphOrcs:
    """
    'Музыкальное произведение' для реализации
    связи многие-ко-многим
    """
    def __init__(self, orch_id, symph_id):
        self.orch_id = orch_id
        self.symph_id = symph_id


# Оркестры
orchestras = [
    orchestra(1, 'Российский национальный'),
    orchestra(2, 'Национальный филармонический оркестр'),
    orchestra(3, 'Большой симфонический оркестр им. Чайковского'),
    orchestra(4, 'Симфонический оркестр Большого театра'),
    orchestra(5, 'Государственный симфонический оркестр «Новая Россия»'),
    orchestra(6, 'Государственная академическая симфоническая капелла России'),
]


# Музыкальные произведения
symphonys = [
    symphony(1, 'Вторая (богатырская) симфония Александра Бородина', 111, 3),
    symphony(2, 'Шестая (патетическая) симфония Петра Чайковского', 97, 1),
    symphony(3, 'Третья симфония («Божественная поэма») Александра Скрябина', 89, 2),
    symphony(4, 'Первая (классическая) симфония Сергея Прокофьева', 145, 1),
```

```
    symphony(5, 'Седьмая (ленинградская) симфония Дмитрия Шостаковича', 134, 3),
]


symphs_orcs = [
    SymphOrcs(1,1),
    SymphOrcs(2,2),
    SymphOrcs(3,3),
    SymphOrcs(3,4),
    SymphOrcs(3,5),


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



def task_1(symphonys, orchestras):
    # Соединение данных один-ко-многим
    one_to_many = [(s.name, s.mins, o.name)
        for o in orchestras
        for s in symphonys
        if s.orch_id==o.id]

    res_11 = list(filter(lambda i:i[2].find('оркестр')!=-1, one_to_many))
    return(res_11)

def task_2(symphonys, orchestras):

    one_to_many = [(s.name, s.mins, o.name)
        for o in orchestras
        for s in symphonys
        if s.orch_id==o.id]

    res_12_unsorted = []
    for d in orchestras:
        d_Symphs = list(filter(lambda i: i[2]==d.name, one_to_many))
        if len(d_Symphs) > 0:
            d_Symph_lenght = [mins for _,mins,_ in d_Symphs]
            d_lenght_avg = round(sum(d_Symph_lenght)/len(d_Symph_lenght), 2)
            res_12_unsorted.append((d.name, d_lenght_avg))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return(res_12)

def task_3(symphonys, orchestras, symphs_orcs):
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(o.name, so.orch_id, so.symph_id)
        for o in orchestras
        for so in symphs_orcs
        if o.id==so.orch_id]

    many_to_many = [(s.name, s.mins, orch_name)
        for orch_name, orch_id, symph_id in many_to_many_temp
        for s in symphonys if s.id==orch_id]
    res_13 = list(filter(lambda i: i[0][0]=='П' or i[0][0]=='В', many_to_many))
    return(res_13)

def main():
    print('Задание E1\nСписок всех оркестров, в названии которых присутствует слово "оркестр":')
    print(task_1(symphonys, orchestras))
    print('\nЗадание E2\nСписок оркестров со средней длительностью исполняемых произведений, сортированный
по средней длительности:')
```

```
    print(task_2(symphonys, orchestras))
    print('\nЗадание Е3\nСписок всех симфоний, названия которых начинаются с букв "П" или "В":')
    print(task_3(symphonys, orchestras, symphs_orcs))
```

tests.py

```
import unittest
from RK_PCPL_Peresipko import *

class TestRK2(unittest.TestCase):
    # Оркестры
    orchestras = [
        orchestra(1, 'Российский национальный'),
        orchestra(2, 'Национальный филармонический оркестр'),
        orchestra(3, 'Большой симфонический оркестр им. Чайковского'),
        orchestra(4, 'Симфонический оркестр Большого театра'),
        orchestra(5, 'Государственный симфонический оркестр «Новая Россия»'),
        orchestra(6, 'Государственная академическая симфоническая капелла России'),
    ]

    # Музыкальные произведения
    symphonys = [
        symphony(1, 'Вторая (богатырская) симфония Александра Бородина', 111, 3),
        symphony(2, 'Шестая (патетическая) симфония Петра Чайковского', 97, 1),
        symphony(3, 'Третья симфония («Божественная поэма») Александра Скрябина', 89, 2),
        symphony(4, 'Первая (классическая) симфония Сергея Прокофьева', 145, 1),
        symphony(5, 'Седьмая (ленинградская) симфония Дмитрия Шостаковича', 134, 3),
    ]

    def test1(self):
        self.assertEqual(task_1(symphonys, orchestras), [('Третья симфония («Божественная поэма») Александра
Скрябина', 89, 'Национальный филармонический оркестр'),
('Вторая (богатырская) симфония Александра Бородина', 111, 'Большой симфонический оркестр им.
Чайковского'),
('Седьмая (ленинградская) симфония Дмитрия Шостаковича', 134, 'Большой симфонический оркестр им.
Чайковского')])

    def test2(self):
        self.assertEqual(task_2(symphonys, orchestras), [('Большой симфонический оркестр им. Чайковского', 122.5),
('Российский национальный', 121.0),
('Национальный филармонический оркестр', 89.0)])

    def test3(self):
        self.assertEqual(task_3(symphonys, orchestras, symphs_orcs), [('Вторая (богатырская) симфония Александра
Бородина', 111, 'Российский национальный'),
('Первая (классическая) симфония Сергея Прокофьева', 145, 'Симфонический оркестр Большого театра')])

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

Результаты выполнения