Коломийчук Илья ИУ5-31Б

Отчет по РК№2

Изменённая программа для модульного тестирования:

```
# используется для сортировки
# Дом улица
from operator import itemgetter
import unittest

class house:
    """Дом"""
    def __init__(self, id, name, square, price,
count_rooms, street_id):
        self.id = id
        self.name = name
        self.square = square
        self.price = price
        self.count_rooms = count_rooms
        self.street_id = street_id

class street:
    """Улица"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Ho_st:
    """
    'Дома улицы' для реализации
    связи многие-ко-многим
    """
    def __init__(self, house_id, street_id):
        self.house_id = house_id
        self.street_id = street_id
```

```python
# Дома
houses = [
    house(1, 'Дом', 100, 10_000_000, 6, 1),
    house(2, 'Дом с гаражом', 250, 15_000_000, 8, 1),
    house(3, 'Коттедж',300, 20_000_000, 10, 3),
    house(4, 'Коттедж', 350, 22_499_000, 12, 4),
    house(5, 'Вилла', 150, 15_000_000, 4, 2),
    house(6, 'Вилла', 500, 30_000_000, 15, 2),
]

# Сотрудники
streets = [
    street(1, 'Чертановская'),
    street(2, 'Бауманская'),
    street(3, 'Горького'),
    street(4, 'Скобелевская'),
    street(5, 'Бульвар адмирала Ушакова'),
]

houses_streets = [
    Ho_st(1,1),
    Ho_st(2,2),
    Ho_st(3,3),
    Ho_st(3,4),
    Ho_st(3,5),
    Ho_st(1,2),
    Ho_st(4,1),
    Ho_st(5,2),
    Ho_st(6,3),
    Ho_st(6,4),
    Ho_st(6,5),
]

def one_to_many_func(streets, houses):
```

```
    res = [(h.name, h.square, h.price, h.count_rooms,
s.name)
    for s in streets
    for h in houses
    if h.street_id==s.id]
    return res

def many_to_many_func(streets, houses_streets):
    res =  [(s.name, hs.street_id, hs.house_id)
    for s in streets
    for hs in houses_streets
    if s.id==hs.street_id]
    many_to_many = [(h.name ,h.square, h.price,
h.count_rooms, street_name)
    for street_name, street_id, house_id in res
    for h in houses if h.id==house_id]
    return many_to_many

# Соединение данных один-ко-многим
one_to_many = one_to_many_func(streets, houses)

# Соединение данных многие-ко-многим

many_to_many = many_to_many_func(streets,
houses_streets)

print('Задание А1')
res_11 = sorted(one_to_many, key=itemgetter(3))
print(*res_11, sep = '\n')

print('\nЗадание А2')
res_12_unsorted = []
# Перебираем все улицы
for s in streets:
    # Список домов на заданной улице
```

```
    h = list(filter(lambda i: i[4]==s.name,
one_to_many))
    # стоимость домов на заданной улице
    h_price = [price for _,_,price,_,_ in h]
    # Суммарная стоимость недвижимости на улице
    h_sals_sum = sum(h_price)
    res_12_unsorted.append((s.name, h_sals_sum))

# Сортировка по суммарной стоимости недвижимости
res_12 = sorted(res_12_unsorted, key=itemgetter(1),
reverse=True)

print(*res_12, sep = '\n')

print('\nЗадание А3')
res_13 = {}
# Перебираем все улицы
for s in streets:
    if len(s.name) > 0:
        # Список домов
        s_houses = list(filter(lambda i: i[4]==s.name,
many_to_many))
        # Только названия строений
        s_houses_names = [x for x,_,_,_,_ in s_houses]
        # Добавляем результат в словарь
        # ключ - название улицы, значение - список
названий строений
        res_13[s.name] = s_houses_names
for i in res_13.keys():
    print(i, '-', res_13[i], end = '\n')

class TestRK1(unittest.TestCase):
    def test_one_to_many(self):
        global streets
        global houses
```

```
        self.assertEqual(one_to_many_func(streets,
houses),[('Дом', 100, 10000000, 6, 'Чертановская'),
('Дом с гаражом', 250, 15000000, 8, 'Чертановская'),
('Вилла', 150, 15000000, 4, 'Бауманская'), ('Вилла',
500, 30000000, 15, 'Бауманская'), ('Коттедж', 300,
20000000, 10, 'Горького'), ('Коттедж', 350, 22499000,
12, 'Скобелевская')])

    def test_many_to_many(self):
        global streets
        global houses_streets
        self.assertEqual(many_to_many_func(streets,
houses_streets), [('Дом', 100, 10000000, 6,
'Чертановская'), ('Коттедж', 350, 22499000, 12,
'Чертановская'), ('Дом с гаражом', 250, 15000000, 8,
'Бауманская'), ('Дом', 100, 10000000, 6, 'Бауманская'),
('Вилла', 150, 15000000, 4, 'Бауманская'), ('Коттедж',
300, 20000000, 10, 'Горького'), ('Вилла', 500, 30000000,
15, 'Горького'), ('Коттедж', 300, 20000000, 10,
'Скобелевская'), ('Вилла', 500, 30000000, 15,
'Скобелевская'), ('Коттедж', 300, 20000000, 10, 'Бульвар
адмирала Ушакова'), ('Вилла', 500, 30000000, 15,
'Бульвар адмирала Ушакова')])

    def test1(self):
        global res_11
        self.assertEqual(res_11, [('Вилла', 150,
15000000, 4, 'Бауманская'), ('Дом', 100, 10000000, 6,
'Чертановская'), ('Дом с гаражом', 250, 15000000, 8,
'Чертановская'), ('Коттедж', 300, 20000000, 10,
'Горького'), ('Коттедж', 350, 22499000, 12,
'Скобелевская'), ('Вилла', 500, 30000000, 15,
'Бауманская')])

    def test2(self):
```

```
        global res_12
        self.assertEqual(res_12, [('Бауманская',
45000000), ('Чертановская', 25000000), ('Скобелевская',
22499000), ('Горького', 20000000), ('Бульвар адмирала
Ушакова', 0)])

    def test3(self):
        global res_13
        self.assertEqual(res_13, {'Чертановская':
['Дом', 'Коттедж'], 'Бауманская': ['Дом с гаражом',
'Дом', 'Вилла'], 'Горького': ['Коттедж', 'Вилла'],
'Скобелевская': ['Коттедж', 'Вилла'], 'Бульвар адмирала
Ушакова': ['Коттедж', 'Вилла']})
```

Результат тестирования: