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

# 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:

