Рубежный контроль №2

Парадигмы и конструкции языков программирования

Барабаш Диана

ИБМ3-34Б

**Вариант 6**

Класс 1 – Дом

Класс 2 – Улица

**Программа(текст)1:**

from operator import itemgetter

class Dom:

"""Дом"""

def \_\_init\_\_(self, id, number, area, street\_id):

self.id = id

self.number = number

self.area = area

self.street\_id = street\_id

class Ulitsa:

"""Улица"""

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class DomUlitsa:

"""

'Дома на улице' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, street\_id, dom\_id):

self.street\_id = street\_id

self.dom\_id = dom\_id

def get\_one\_to\_many(streets, houses):

"""Соединение данных один-ко-многим"""

return [

(house.number, house.area, street.name)

for street in streets

for house in houses

if house.street\_id == street.id

]

def get\_many\_to\_many(streets, houses, houses\_streets):

"""Соединение данных многие-ко-многим"""

many\_to\_many\_temp = [

(street.name, hs.street\_id, hs.dom\_id)

for street in streets

for hs in houses\_streets

if street.id == hs.street\_id

]

return [

(house.number, house.area, street\_name)

for street\_name, \_, dom\_id in many\_to\_many\_temp

for house in houses if house.id == dom\_id

]

def task\_a1(one\_to\_many, streets):

"""Задание A1: список всех домов, отсортированный по порядку улиц"""

# Составляем словарь {street\_name: index}, чтобы учитывать порядок из streets

street\_order = {street.name: idx for idx, street in enumerate(streets)}

# Сортируем по индексу улицы в order

return sorted(one\_to\_many, key=lambda x: street\_order[x[2]])

def task\_a2(one\_to\_many, streets):

"""Задание A2: улицы с суммарной площадью домов, отсортированные по убыванию площади"""

result = []

for street in streets:

# Фильтруем дома на текущей улице

houses\_on\_street = list(filter(lambda x: x[2] == street.name, one\_to\_many))

# Если дома есть

if houses\_on\_street:

# Суммируем площади домов

total\_area = sum(house[1] for house in houses\_on\_street)

result.append((street.name, total\_area))

# Сортируем по убыванию площади

return sorted(result, key=itemgetter(1), reverse=True)

def task\_a3(many\_to\_many, streets):

"""Задание A3: улицы, содержащие 'улица', и номера домов"""

result = {}

for street in streets:

if 'улица' in street.name:

# Номера домов на этой улице

houses\_on\_street = list(filter(lambda x: x[2] == street.name, many\_to\_many))

house\_numbers = [house[0] for house in houses\_on\_street]

result[street.name] = house\_numbers

return result

**Программа(текст)2:**

import unittest

from main import get\_one\_to\_many, get\_many\_to\_many, task\_a1, task\_a2, task\_a3, Ulitsa, Dom, DomUlitsa

class TestTasks(unittest.TestCase):

def setUp(self):

"""Настройка данных для тестов"""

self.streets = [

Ulitsa(1, 'Красная улица'),

Ulitsa(2, 'Зеленая улица'),

Ulitsa(3, 'Синяя улица'),

Ulitsa(4, 'Отделочная улица'),

]

self.houses = [

Dom(1, '10', 100, 1),

Dom(2, '20', 150, 1),

Dom(3, '30', 120, 2),

Dom(4, '40', 80, 2),

Dom(5, '50', 200, 3),

]

self.houses\_streets = [

DomUlitsa(1, 1),

DomUlitsa(1, 2),

DomUlitsa(2, 3),

DomUlitsa(2, 4),

DomUlitsa(3, 5),

DomUlitsa(4, 5),

]

def test\_task\_a1(self):

"""Тест для задания A1"""

one\_to\_many = get\_one\_to\_many(self.streets, self.houses)

result = task\_a1(one\_to\_many, self.streets)

print("Task A1 result:", result)

self.assertEqual(result, [

('10', 100, 'Красная улица'),

('20', 150, 'Красная улица'),

('30', 120, 'Зеленая улица'),

('40', 80, 'Зеленая улица'),

('50', 200, 'Синяя улица'),

])

def test\_task\_a2(self):

"""Тест для задания A2"""

one\_to\_many = get\_one\_to\_many(self.streets, self.houses)

result = task\_a2(one\_to\_many, self.streets)

self.assertEqual(result, [

('Красная улица', 250),

('Зеленая улица', 200),

('Синяя улица', 200),

])

def test\_task\_a3(self):

"""Тест для задания A3"""

many\_to\_many = get\_many\_to\_many(self.streets, self.houses, self.houses\_streets)

result = task\_a3(many\_to\_many, self.streets)

self.assertEqual(result, {

'Красная улица': ['10', '20'],

'Зеленая улица': ['30', '40'],

'Синяя улица': ['50'],

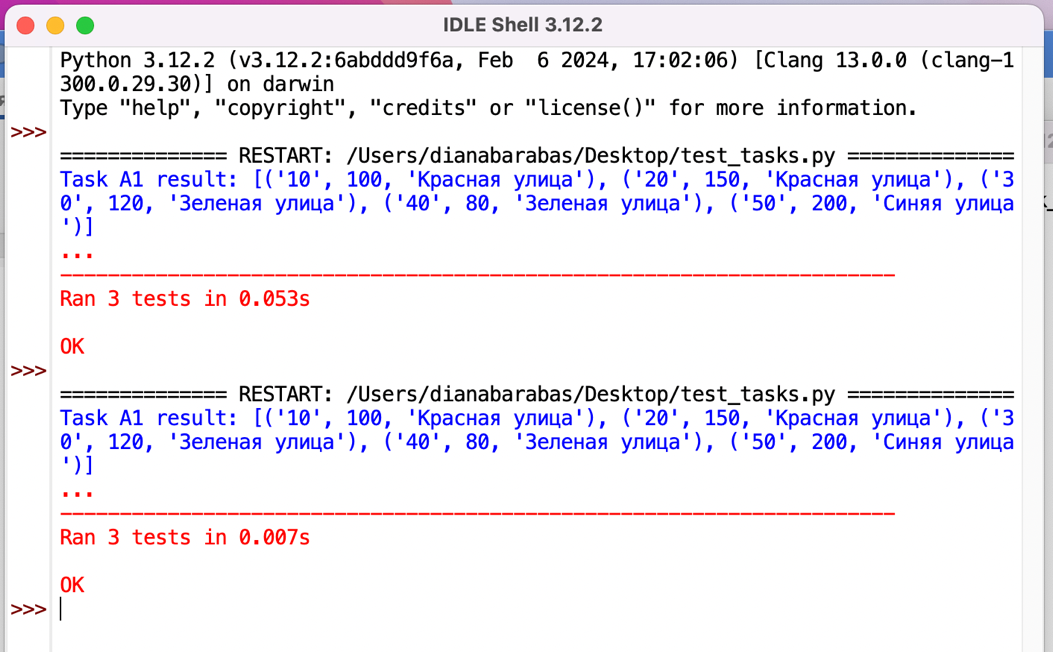
'Отделочная улица': ['50'],

})

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**Вывод:**

****