Текст программы main.py:

```
from operator import itemgetter


class Program:
    """Класс программы"""

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


class Computer:
    """Класс компьютера"""

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


class ProgramComputerLink:
    """Класс для реализации связи многие ко многим"""

    def __init__(self, prog_id, comp_id):
        self.prog_id = prog_id
        self.comp_id = comp_id


def create_sample_data():
    progs = [Program(1, 'Photoshop', 789, 1),
             Program(2, 'Euro Truck Simulator', 2300, 3),
             Program(3, 'Euro Truck Simulator 2', 2566, 3),
             Program(4, 'Internet Explorer', 2, 1),
             Program(5, 'Microsoft To Do', 40, 2)]

    comps = [Computer(1, 'Домашний компьютер'),
             Computer(2, 'Рабочий компьютер'),
             Computer(3, 'PlayStation'),
             Computer(4, 'Планшет'),
             Computer(5, 'Xbox'),
             Computer(6, 'Бабушкин пк')]

    comps_progs = [ProgramComputerLink(1, 1),
                   ProgramComputerLink(2, 3),
                   ProgramComputerLink(3, 3),
                   ProgramComputerLink(4, 1),
                   ProgramComputerLink(5, 2),

                   ProgramComputerLink(1, 4),
                   ProgramComputerLink(2, 5),
                   ProgramComputerLink(4, 6)]

    return progs, comps, comps_progs


def get_one_to_many_relationship(progs, comps):
    return [(p.name, p.mem, c.name)
            for c in comps
            for p in progs
            if p.comp_id == c.id]


def get_many_to_many_relationship(comps, comps_progs, progs):
    many_to_many_temp = [(c.name, cp.comp_id, cp.prog_id)
                         for c in comps
                         for cp in comps_progs
                         if c.id == cp.comp_id]
```

```
    return [(p.name, p.mem, c_name)
            for c_name, c_id, pr_id in many_to_many_temp
            for p in progs
            if p.id == pr_id]


def filter_computers_with_word(comps, word):
    return [[c.name, c.id] for c in comps if word in c.name]


def calculate_average_program_size(comps, one_to_many_relationship):
    answer = []
    for c in comps:
        c_progs = list(filter(lambda i: i[2] == c.name, one_to_many_relationship))
        if len(c_progs) > 0:
            c_mem = [mem for _, mem, _ in c_progs]
            sr_mem = round(sum(c_mem) / len(c_progs), 2)
            answer.append((c.name, sr_mem))
    return sorted(answer, key=itemgetter(1))


def get_programs_starting_with_letter(progs, letter, many_to_many_relationship):
    return {p.name: [c_name for p_name, _, c_name in many_to_many_relationship if
p.name == p_name]
            for p in progs
            if p.name[0] == letter}


def main():
    progs, comps, comps_progs = create_sample_data()

    one_to_many_relationship = get_one_to_many_relationship(progs, comps)
    many_to_many_relationship = get_many_to_many_relationship(comps, comps_progs,
progs)

    answer_1 = {i[0]: [p.name for p in progs if p.comp_id == i[1]] for i in
filter_computers_with_word(comps, "компьютер")}
    print('Задание E1 (присутствует слово "компьютер"):\n', answer_1)

    answer_2 = calculate_average_program_size(comps, one_to_many_relationship)
    print('Задание E2 (средний размер программы):\n', answer_2)

    answer_3 = get_programs_starting_with_letter(progs, 'E',
many_to_many_relationship)
    print('Задание E3 (название начинается с буквы "E"):\n', answer_3)


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

Текст программы test.py:

```
import unittest
from main import *

class TestAnswer1(unittest.TestCase):
    def data(self):
        self.progs = [Program(1, 'Photoshop', 789, 1),
                  Program(2, 'Euro Truck Simulator', 2300, 3),
                  Program(3, 'Euro Truck Simulator 2', 2566, 3),
                  Program(4, 'Internet Explorer', 2, 1),
                  Program(5, 'Microsoft To Do', 40, 2)]

        self.comps = [Computer(1, 'Домашний компьютер'),
                  Computer(2, 'Рабочий компьютер'),
```

```
                Computer(3, 'PlayStation'),
                Computer(4, 'Планшет'),
                Computer(5, 'Xbox'),
                Computer(6, 'Бабушкин пк')]

        self.comps_progs = [ProgramComputerLink(1, 1),
                            ProgramComputerLink(2, 3),
                            ProgramComputerLink(3, 3),
                            ProgramComputerLink(4, 1),
                            ProgramComputerLink(5, 2),

                            ProgramComputerLink(1, 4),
                            ProgramComputerLink(2, 5),
                            ProgramComputerLink(4, 6)]

    def test_answer_1(self):
        self.data()
        result = {i[0]: [p.name for p in self.progs if p.comp_id == i[1]] for i in
filter_computers_with_word(self.comps, "компьютер")}
        self.assertEqual(result, {'Домашний компьютер': ['Photoshop', 'Internet
Explorer'], 'Рабочий компьютер': ['Microsoft To Do']})

    def test_answer_2(self):
        self.data()
        result = calculate_average_program_size(self.comps,
get_one_to_many_relationship(self.progs, self.comps))
        self.assertEqual(result, [('Рабочий компьютер', 40.0), ('Домашний компьютер',
395.5), ('PlayStation', 2433.0)])

    def test_answer_3(self):
        self.data()
        result = get_programs_starting_with_letter(self.progs, 'E',
get_many_to_many_relationship(self.comps, self.comps_progs, self.progs))
        self.assertEqual(result, {'Euro Truck Simulator': ['PlayStation', 'Xbox'],
'Euro Truck Simulator 2': ['PlayStation']})

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

Результат тестирования:

```
Ran 3 tests in 0.006s

OK
```