Код программы

- main.py

```
from pprint import pp


class Producer:
    """ Производитель. """

    def __init__(self, ident: int, name: str):
        self.id = ident
        self.name = name


class Part:
    """ Деталь. """

    def __init__(self, ident: int, name: str, price: int, prod_id: int):
        self.id = ident
        self.name = name
        self.price = price
        self.prod_id = prod_id


class ProdPart:
    """ Связь многие-ко-многим для деталей производителя. """

    def __init__(self, prod_id: int, part_id: int):
        self.prod_id = prod_id
        self.part_id = part_id


def one_to_many(prods, parts):
    """ Генерация связей один-ко-многим. """
    return [
        (prod.name, part.name, part.price)
        for prod in prods
        for part in parts
        if part.prod_id == prod.id
    ]


def many_to_many(prods, parts, prod_parts):
    """ Генераация связей многие-ко-многим. """
    many_to_many_temp = [
        (prod.name, prod_part.prod_id, prod_part.part_id)
        for prod in prods
        for prod_part in prod_parts
        if prod.id == prod_part.prod_id
    ]
    return [
        (part.name, prod_name)
        for prod_name, prod_id, part_id in many_to_many_temp
        for part in parts
        if part.id == part_id
    ]


def task1(word: str, one_to_many: list) -> list:
    """ Решение задания 1. """
    return [x for x in one_to_many if word in x[0]]
```

```python
def average(a: list[int]) -> float:
    """ Подсчёт среднего по списку. """
    return sum(a) / len(a)


def task2(one_to_many: list) -> list:
    """ Решение задания 2. """
    avgs = {
        name: 0
        for name in set([x[0] for x in one_to_many])
    }
    for x in one_to_many:
        avgs[x[0]] = average(
            [y[2] for y in one_to_many if y[0] == x[0]]
        )
    return sorted(avgs.items(), key=lambda item: item[1])


def task3(letter: str, many_to_many: list) -> list:
    """ Решение задания 3. """
    return [x for x in many_to_many if x[0][0] == letter]


if __name__ == '__main__':
    # Производители
    prods = [
        Producer(1, 'ВАГОНМАШ'),
        Producer(2, 'Завод Драйв'),
        Producer(3, 'Партариум'),
        Producer(4, 'Фабрикатор'),
        Producer(5, 'Катлокси'),
        Producer(6, 'Завод.рф'),
    ]

    # Детали
    parts = [
        Part(1, 'Штифт', 100, 1),
        Part(2, 'Гайка', 50, 2),
        Part(3, 'Мост', 300, 3),
        Part(4, 'Вал', 700, 4),
        Part(5, 'Болт', 80, 5),
        Part(6, 'Уголок', 150, 6),
        Part(7, 'Двутавр', 500, 1),
        Part(8, 'Сетка', 400, 2),
        Part(9, 'Колесо', 1000, 3),
        Part(10, 'Скоба', 20, 4),
        Part(11, 'Панель', 600, 5),
        Part(12, 'Шайба', 50, 6),
    ]

    # Производитель-Деталь
    prod_parts = [
        ProdPart(1, 1),
        ProdPart(2, 2),
        ProdPart(3, 3),
        ProdPart(4, 4),
        ProdPart(5, 5),
        ProdPart(6, 6),
        ProdPart(1, 7),
        ProdPart(2, 8),
        ProdPart(3, 9),
        ProdPart(4, 10),
        ProdPart(5, 11),
```

```
        ProdPart(6, 12),
    ]

    one_to_many_data = one_to_many(prods, parts)
    many_to_many_data = many_to_many(prods, parts, prod_parts)

    print('Задание 1')
    pp(task1('Драйв', one_to_many_data))

    print('Задание 2')
    pp(task2(one_to_many_data))

    print('Задание 3')
    pp(task3('К', many_to_many_data))
```

- tests.py

```
import unittest

from main import *


class TestSolutions(unittest.TestCase):
    def setUp(self):
        self.prods = [
            Producer(1, 'ВАГОНМАШ'),
            Producer(2, 'Завод Драйв'),
            Producer(3, 'Партариум'),
        ]
        self.parts = [
            Part(1, 'Крепление', 100, 1),
            Part(2, 'Гайка', 50, 2),
            Part(3, 'Мост', 300, 3),
            Part(7, 'Двутавр', 500, 1),
            Part(8, 'Сетка', 400, 2),
            Part(9, 'Колесо', 1000, 3),
        ]
        self.prod_parts = [
            ProdPart(1, 1),
            ProdPart(2, 2),
            ProdPart(3, 3),
            ProdPart(1, 7),
            ProdPart(2, 8),
            ProdPart(3, 9),
        ]
        self.test_word = 'Драйв'
        self.test_letter = 'К'

    def test_task1_solution(self):
        result = task1(
            self.test_word,
            one_to_many(self.prods, self.parts)
        )
        self.assertEqual(
            result,
            [('Завод Драйв', 'Гайка', 50), ('Завод Драйв', 'Сетка', 400)]
        )

    def test_task2_solution(self):
        result = task2(one_to_many(self.prods, self.parts))
        self.assertEqual(
            result,
```

```
            [('Завод Драйв', 225.0), ('ВАГОНМАШ', 300.0), ('Партариум',
650.0)]
        )

    def test_task3_solution(self):
        result = task3(
            self.test_letter,
            many_to_many(self.prods, self.parts, self.prod_parts)
        )
        self.assertEqual(
            result,
            [('Крепление', 'ВАГОНМАШ'), ('Колесо', 'Партариум')]
        )


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

Результат работы

- Тесты пройдены успешно:

```
Ran 3 tests in 0.002s


OK
```

- Тесты не пройдены:

```
Ran 3 tests in 0.012s


FAILED (failures=1)



[('Крепление', 'Завод Драйв'), ('Колесо', 'Партариум')] != [('Крепление',
'ВАГОНМАШ'), ('Колесо', 'Партариум')]
```