Текст программы

main.py

```
from operator import itemgetter
class Emp:

    def __init__(self, id, name, price, dep_id):
        self.id = id
        self.name = name
        self.price = price
        self.dep_id = dep_id
class Dep:
    """Производитель"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
class EmpDep:
    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id
# Производители
deps = [
    Dep(1, 'AZ Spa'),
    Dep(2, 'Magido'),
    Dep(3, 'Carmec'),
    Dep(11, 'VTM Group'),
    Dep(22, 'Carmec Pro'),
    Dep(33, 'Newen'),
]
# Детали
emps = [
    Emp(1, 'Фреза', 2500, 1),
    Emp(2, 'Головка', 3000, 2),
    Emp(3, 'Форсунка', 4500, 3),
    Emp(4, 'Цилиндр', 5000, 3),
    Emp(5, 'Кардан', 2500, 22),
]
emps_deps = [
    EmpDep(1, 1),
    EmpDep(2, 2),
    EmpDep(3, 3),
    EmpDep(3, 4),
    EmpDep(3, 5),
    EmpDep(11, 1),
    EmpDep(22, 2),
    EmpDep(33, 3),
    EmpDep(33, 4),
    EmpDep(33, 5),
]

def a1_solution(one_to_many):
```

```
    res_a1 = sorted(one_to_many, key=itemgetter(2))
    return res_a1

def a2_solution(one_to_many):
    res_a2_unsorted = []
    # Перебираем всех производителей
    for d in deps:
        # Список деталей производителя
        d_emps = list(filter(lambda i: i[2] == d.name, one_to_many))
        # Если производитель пустой
        if len(d_emps) > 0:
            # Цены деталей производителя
            d_sals = [price for
            _
            , price, _ in d_emps]
            # Суммарная цена деталей производителя
            d_sals_sum = sum(d_sals)
            res_a2_unsorted.append((d.name, d_sals_sum))
    # Сортировка по суммарной цене
    res_a2 = sorted(res_a2_unsorted, key=itemgetter(1), reverse=True)
    return res_a2

def a3_solution(many_to_many):
    res_a3 = {}
    # Перебираем всех производителей
    for d in deps:
        if 'Carmec' in d.name:
            # Список деталей производителя
            d_emps = list(filter(lambda i: i[2] == d.name, many_to_many))
            # Только названия деталей производителя
            d_emps_names = [x for x,
            _
            , _ in d_emps]
            # Добавляем результат в словарь
            # ключ - производитель, значение - список названий
            res_a3[d.name] = d_emps_names
    return res_a3

def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.price, d.name)
            for d in deps
            for e in emps
            if e.dep_id == d.id]
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
            for d in deps
            for ed in emps_deps
            if d.id == ed.dep_id]
    many_to_many = [(e.name, e.price, dep_name)
            for dep_name, dep_id, emp_id in many_to_many_temp
            for e in emps if e.id == emp_id]
```

```
    print('Задание А1')
    print(a1_solution(one_to_many))
    print('\nЗадание А2')
    print(a2_solution(one_to_many))
    print('\nЗадание А3')
    print(a3_solution(many_to_many))

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

tddtests.py

```
import unittest
from main import *


class TestRK2(unittest.TestCase):
    # Производители
    deps = [
    Dep(1, 'AZ Spa'),
    Dep(2, 'Magido'),
    Dep(3, 'Carmec'),
    Dep(11, 'VTM Group'),
    Dep(22, 'Carmec Pro'),
    Dep(33, 'Newen'),
]
    # Детали
    emps = [
    Emp(1, 'Фреза', 2500, 1),
    Emp(2, 'Головка', 3000, 2),
    Emp(3, 'Форсунка', 4500, 3),
    Emp(4, 'Цилиндр', 5000, 3),
    Emp(5, 'Кардан', 2500, 22),
]

    def test_A1(self):
        one_to_many = [(e.name, e.price, d.name)
            for d in deps
            for e in emps
            if e.dep_id == d.id]
        self.assertEqual(a1_solution(one_to_many), [('Фреза', 2500, 'AZ Spa'), ('Форсунка', 4500, 'Carmec'),
('Цилиндр', 5000, 'Carmec'),
('Кардан', 2500, 'Carmec Pro'), ('Головка', 3000, 'Magido')])
    def test_A2(self):
        one_to_many = [(e.name, e.price, d.name)
            for d in deps
            for e in emps
            if e.dep_id == d.id]
        self.assertEqual(a2_solution(one_to_many), [('Carmec', 9500), ('AZ Spa', 2500), ('Carmec Pro', 2500)])
    def test_A3(self):
        many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
```

```
        for d in deps
        for ed in emps_deps
        if d.id == ed.dep_id]
    many_to_many = [(e.name, e.price, dep_name)
        for dep_name, dep_id, emp_id in many_to_many_temp
        for e in emps if e.id == emp_id]
    self.assertEqual(a3_solution(many_to_many), {'Carmec': ['Форсунка', 'Цилиндр', 'Кардан'], 'Carmec Pro':
['Головка']})

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

Пример успешного прохождения тестов

```
baga@MacBook-Air-Bagauddin рк % python3 -u "/Users/baga/Desktop/рк/tddtests.py"
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Пример неудачного прохождения тестов

```
baga@MacBook-Air-Bagauddin рк % python3 -u "/Users/baga/Desktop/рк/tddtests.py"
..F
======================================================================
FAIL: test_A3 (__main__.TestRK2.test_A3)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/baga/Desktop/рк/tddtests.py", line 45, in test_A3
    self.assertEqual(a3_solution(many_to_many), {'Carmec': ['Форсунка', 'Кардан'], 'Carmec Pro': ['Головка']})
AssertionError: {'Carmec': ['Форсунка', 'Цилиндр', 'Кардан'], 'Carmec Pro': ['Головка']} != {'Carmec': ['Форсунка', 'Кардан'], 'Carmec Pro': ['Головка']}
- {'Carmec': ['Форсунка', 'Цилиндр', 'Кардан'], 'Carmec Pro': ['Головка']}
?                          ----------

+ {'Carmec': ['Форсунка', 'Кардан'], 'Carmec Pro': ['Головка']}

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
```