Текст программы
Main.py

```python
from operator import itemgetter

class HardDrive:
    """Hard Drive"""
    def __init__(self, id, model, capacity_gb, computer_id):
        self.id = id
        self.model = model
        self.capacity_gb = capacity_gb
        self.computer_id = computer_id

class Computer:
    """Computer"""
    def __init__(self, id, name, type, hard_drive_id, cost):
        self.id = id
        self.name = name
        self.type = type
        self.hard_drive_id = hard_drive_id
        self.cost = cost

class ComputerHardDrive:
    """'Computers with Hard Drives' for implementing many-to-many
relationship"""
    def __init__(self, computer_id, hard_drive_id):
        self.computer_id = computer_id
        self.hard_drive_id = hard_drive_id

# Hard Drives
hard_drives = [
    HardDrive(1, 'Seagate 1TB', 1000, 1),
    HardDrive(2, 'Western Digital 2TB', 2000, 2),
    HardDrive(3, 'Samsung 500GB', 500, 2),
]

# Computers
computers = [
    Computer(1, 'Computer 1', 'Desktop', 1, 800),
    Computer(2, 'Laptop 1', 'Laptop', 2, 1200),
    Computer(3, 'Computer 2', 'Desktop', 3, 700),
]

computer_hard_drives = [
    ComputerHardDrive(1, 1),
    ComputerHardDrive(2, 2),
    ComputerHardDrive(3, 3),
    ComputerHardDrive(3, 2),
]

# используется для сортировки
from operator import itemgetter



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 все жесткие диски
    for h in hard_drives:
        # Список компьютеров с данным жестким диском
        h_comps = list(filter(lambda i: i[0] == h.model, one_to_many))
        # Если есть компьютеры с этим жестким диском
        if len(h_comps) > 0:
            # Сумма объемов жестких дисков компьютеров
            h_capacities = [capacity for _, capacity, _ in h_comps]
            # Сумма объемов жестких дисков компьютера
            h_capacities_sum = sum(h_capacities)
            res_a2_unsorted.append((h.model, h_capacities_sum))

    # Сортировка по сумме объемов
    res_a2 = sorted(res_a2_unsorted, key=itemgetter(1), reverse=True)
    return res_a2



def a3_solution(many_to_many):
    res_a3 = {}
    # Перебираем все компьютеры
    for c in computers:
        if 'Com' in c.name:
            # Список жестких дисков компьютера
            c_drives = list(filter(lambda ch: ch.computer_id == c.id,
computer_hard_drives))
            # Только модели жестких дисков
            c_drives_models = [hard_drive.model for hard_drive in hard_drives
if hard_drive.id in [drive.hard_drive_id for drive in c_drives]]
            # Добавляем результат в словарь
            # ключ - компьютер, значение - список моделей жестких дисков
            res_a3[c.name] = c_drives_models
    return res_a3





def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(h.model, h.capacity_gb, c.name)
        for c in computers
        for h in hard_drives
        if h.computer_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, ch.computer_id, ch.hard_drive_id)
        for c in computers
        for ch in computer_hard_drives
        if c.id == ch.computer_id]

    many_to_many =  [(h.model, h.capacity_gb, comp_name)
        for comp_name, comp_id, hd_id in many_to_many_temp
        for h in hard_drives if h.id == hd_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()
```

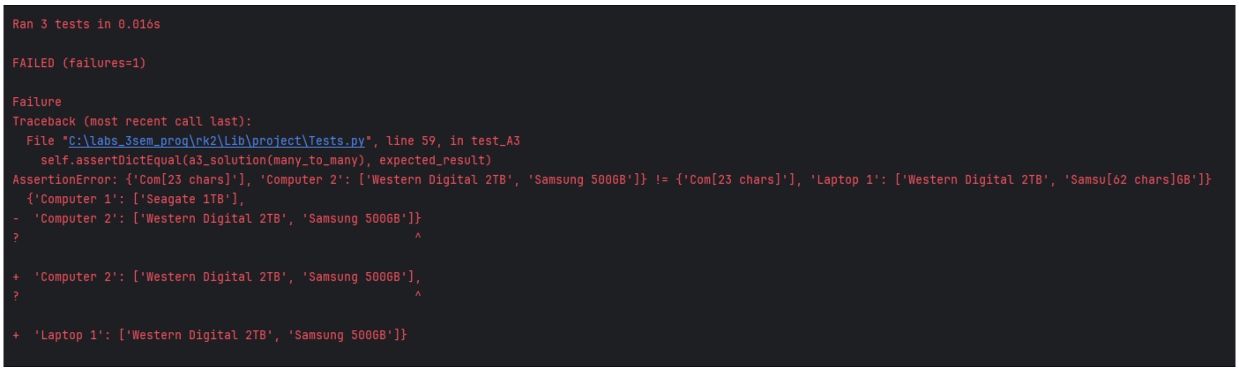
Tests.py

```
import unittest
from main import *

class TestRK2(unittest.TestCase):
    # Жесткие диски
    hard_drives = [
        HardDrive(1, 'Seagate 1TB', 1000, 1),
        HardDrive(2, 'Western Digital 2TB', 2000, 2),
        HardDrive(3, 'Samsung 500GB', 500, 2),
    ]

    # Компьютеры
    computers = [
        Computer(1, 'Computer 1', 'Desktop', 1, 800),
        Computer(2, 'Laptop 1', 'Laptop', 2, 1200),
        Computer(3, 'Computer 2', 'Desktop', 3, 700),
    ]

    # Связь многие-ко-многим
    computer_hard_drives = [
        ComputerHardDrive(1, 1),
        ComputerHardDrive(2, 2),
        ComputerHardDrive(3, 3),
        ComputerHardDrive(3, 2),
    ]

    def test_A1(self):
        one_to_many = [(h.model, h.capacity_gb, c.name)
                       for c in self.computers
                       for h in self.hard_drives
                       if h.computer_id == c.id]
        self.assertEqual(a1_solution(one_to_many),
                         [('Seagate 1TB', 1000, 'Computer 1'), ('Western
Digital 2TB', 2000, 'Laptop 1'),
                          ('Samsung 500GB', 500, 'Laptop 1')])

    def test_A2(self):
        one_to_many = [(h.model, h.capacity_gb, c.name)
                       for c in self.computers
                       for h in self.hard_drives
                       if h.computer_id == c.id]
        expected_result = [('Western Digital 2TB', 2000), ('Seagate 1TB',
1000), ('Samsung 500GB', 500)]
        actual_result = a2_solution(one_to_many)
        self.assertCountEqual(actual_result, expected_result)

    def test_A3(self):
        many_to_many_temp = [(c.name, ch.computer_id, ch.hard_drive_id)
                             for c in self.computers
                             for ch in self.computer_hard_drives
                             if c.id == ch.computer_id]
```

```python
        many_to_many = [(h.model, h.capacity_gb, comp_name)
                        for comp_name, comp_id, hd_id in many_to_many_temp
                        for h in self.hard_drives if h.id == hd_id]

        actual_result = a3_solution(many_to_many)
        expected_result = {
            'Computer 1': ['Seagate 1TB'],
            'Computer 2': ['Western Digital 2TB', 'Samsung 500GB']
        }

        # Сортируем значения в словарях перед сравнением
        for key in actual_result:
            actual_result[key] = sorted(actual_result[key])
        for key in expected_result:
            expected_result[key] = sorted(expected_result[key])

        print("Фактический результат:", actual_result)
        print("Ожидаемый результат:", expected_result)

        self.assertDictEqual(actual_result, expected_result)


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

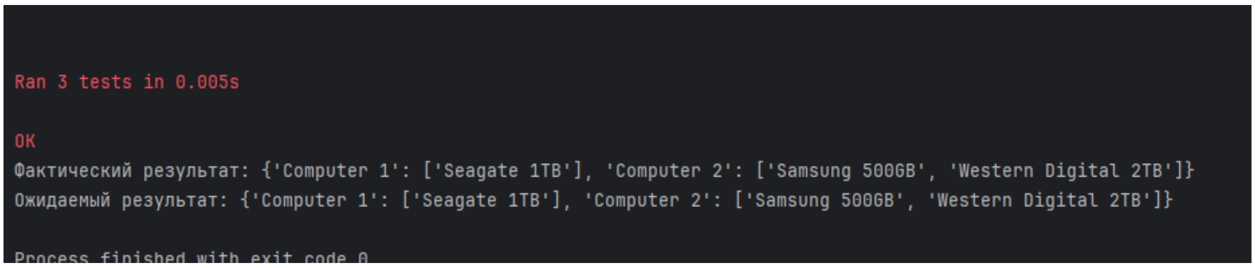
Резултаты работы

Ошибка:



```
Ran 3 tests in 0.016s

FAILED (failures=1)

Failure
Traceback (most recent call last):
  File "C:\labs_3sem_prog\rk2\Lib\project\Tests.py", line 59, in test_A3
    self.assertDictEqual(a3_solution(many_to_many), expected_result)
AssertionError: {'Com[23 chars]'], 'Computer 2': ['Western Digital 2TB', 'Samsung 500GB']} != {'Com[23 chars]'], 'Laptop 1': ['Western Digital 2TB', 'Samsu[62 chars]GB']}
  {'Computer 1': ['Seagate 1TB'],
-  'Computer 2': ['Western Digital 2TB', 'Samsung 500GB']}
?                                                        ^

+  'Computer 2': ['Western Digital 2TB', 'Samsung 500GB'],
?                                                        ^

+  'Laptop 1': ['Western Digital 2TB', 'Samsung 500GB']}
```

Успех



```
Ran 3 tests in 0.005s

OK
Фактический результат: {'Computer 1': ['Seagate 1TB'], 'Computer 2': ['Samsung 500GB', 'Western Digital 2TB']}
Ожидаемый результат: {'Computer 1': ['Seagate 1TB'], 'Computer 2': ['Samsung 500GB', 'Western Digital 2TB']}

Process finished with exit code 0
```