Текст программы из РК1 для модульного тестирования:

Файл tasks.py

```
from operator import itemgetter

class Microprocessor:
    """Микропроцессор"""
    def __init__(self, id, name, dev, year, com_id):
        self.id = id
        self.name = name
        self.dev = dev
        self.year = year
        self.com_id = com_id

class Computer:
    """Компьютер"""
    def __init__(self, id, name, dev, display_size, storage, exp_storage):
        self.id = id
        self.name = name
        self.dev = dev
        self.display_size = display_size
        self.storage = storage
        self.exp_storage = exp_storage

class MicropComp:
    """
    'Микропроцессоры компьютера' для реализации связи многие-ко-многим
    """
    def __init__(self, com_id, microp_id):
        self.com_id = com_id
        self.microp_id = microp_id
# Компьютеры
computers = [Computer(1, 'Zenbook', 'Asus', 15.6, 512, 'MicroSD'),
             Computer(2, 'MacBook', 'Apple', 13.3, 256, 'SSD'),
             Computer(3, 'Inspiron', 'Dell', 15.6, 1024, 'HDD'),
             Computer(4, 'Latitude', 'Dell', 15.6, 1024, 'SSD'),
             Computer(5, 'ThinkPad', 'Lenovo', 15.6, 1024, 'SSD'),
             Computer(6, 'ProBook', 'HP', 15.6, 1024, 'SSD'), ]
# Микропроцессоры
microprocessors = [Microprocessor(1, 'Intel Core i5', 'Intel', 2014, 1),
                   Microprocessor(2, 'Intel Core i7', 'Intel', 2015, 1),
                   Microprocessor(3, 'Intel Core i9', 'Intel', 2016, 2),
                   Microprocessor(4, 'Intel Core i3', 'Intel', 2017, 4),
                   Microprocessor(5, 'AMD Ryzen 5', 'AMD', 2018, 2),
                   Microprocessor(6, 'AMD Ryzen 7', 'AMD', 2019, 3),
                   Microprocessor(7, 'AMD Ryzen 9', 'AMD', 2020, 5),
                   Microprocessor(8, 'AMD Ryzen 3', 'AMD', 2021, 3),
]
microp_comp = [ MicropComp(1, 1),
                MicropComp(1, 3),
                MicropComp(1, 7),
                MicropComp(2, 1),
                MicropComp(2, 5),
                MicropComp(3, 6),
                MicropComp(4, 4),
                MicropComp(4, 6),
                MicropComp(5, 6),
                MicropComp(5, 7),
                MicropComp(6, 1),
                MicropComp(6, 6),
                MicropComp(6, 8),
                MicropComp(7, 1),
```

```
                MicropComp(7, 7) ]

one_to_many = [(c.name, b.name, b.dev) for b in microprocessors for c in
computers if b.com_id == c.id]
many_to_many_curr = [(c.name, bc.com_id, bc.microp_id) for c in computers for
bc in microp_comp if c.id == bc.com_id]
many_to_many = [(com_name, b.name) for com_name, com_id, microp_id in
many_to_many_curr for b in microprocessors if b.id == microp_id]

def task_1():
    return [i for i in one_to_many if i[2][len(i[2]) - 1] == 'l']

def task_2():
    s=0
    count = 0
    avg = []
    for c in computers:
        for b in microprocessors:
            if b.com_id == c.id:
                count = count + 1
                s += b.year
        avg.append((c.name, '%.2f' % (s / count)))
    result2 = sorted(avg, key=itemgetter(1))
    return result2

def task_3():
    result3 = list(filter(lambda i: i[0][0] == 'Z', many_to_many))
    comp = result3[0][0]
    currLst = []
    for i in range(len(result3)):
        if comp == result3[i][0]:
            currLst.append(result3[i][1])
        else:
            comp = result3[i][0]
            currLst = []
            currLst.append(result3[i][1])
    return comp, currLst
```

Текст программы для модульного тестирования:

Файл tdd.py

```
import unittest
from tasks import task_1, task_2, task_3

task_1_result = [('Zenbook', 'Intel Core i5', 'Intel'), ('Zenbook', 'Intel
Core i7', 'Intel'), ('MacBook', 'Intel Core i9', 'Intel'), ('Latitude',
'Intel Core i3', 'Intel')]
task_2_result = [('Zenbook', '2014.50'), ('MacBook', '2015.75'), ('Latitude',
'2017.14'), ('Inspiron', '2017.17'), ('ThinkPad', '2017.50'), ('ProBook',
'2017.50')]
task_3_result = ('Zenbook', ['Intel Core i5', 'Intel Core i9', 'AMD Ryzen
9'])

class TasksTestCase(unittest.TestCase):
    def test_task_1(self):
        self.assertEqual(task_1_result, task_1())
    def test_task_2(self):
        self.assertEqual(task_2_result, task_2())
```

```
    def test_task_3(self):
        self.assertEqual(task_3_result, task_3())
```

Результаты выполнения модульного тестирования:

```
/Users/user/Documents/pythonProject3/bin/Python /Applications/PyCharm.app/Contents/plugins/python/helpers/pycharm/
Testing started at 21:15 ...


Ran 3 tests in 0.002s

OK
```