Рефакторинг текста:

```
from operator import itemgetter


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


class Computer:
    def __init__(self, id, model, price, proc_id):
        self.id = id
        self.model = model
        self.price = price
        self.proc_id = proc_id


class CompDevDept:
    def __init__(self, dept_id, comp_id):
        self.dept_id = dept_id
        self.comp_id = comp_id


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



def get_computers_with_processors_ending_with(procs, comps, suffix):
    return [(c.model, p.name)
```

```
        for p in procs
        for c in comps
        if c.proc_id == p.id and p.name.endswith(suffix)]


def get_departments_with_average_prices(dev_depts, comps_devs, comps):
    many_to_one = [(d.name, c.price)
                   for d in dev_depts
                   for cd in comps_devs
                   for c in comps
                   if c.id == cd.comp_id and d.id == cd.dept_id]

    dept_avg_price = {}
    for dept, price in many_to_one:
        if dept not in dept_avg_price:
            dept_avg_price[dept] = []
        dept_avg_price[dept].append(price)

    return [(dept, sum(prices) / len(prices)) for dept, prices in
dept_avg_price.items()]


def get_departments_starting_with_and_computers(dev_depts, comps_devs,
comps, prefix):
    many_to_many_temp = [(d.name, cd.comp_id)
                         for d in dev_depts
                         for cd in comps_devs
                         if d.id == cd.dept_id and d.name.startswith(prefix)]
```

```
return [(c.model, dept_name)
        for dept_name, comp_id in many_to_many_temp
        for c in comps if c.id == comp_id]
```

Модульные тесты:

```
import unittest
from refactored_rk1 import (
    Processor, Computer, CompDevDept, DevDept,
    get_computers_with_processors_ending_with,
    get_departments_with_average_prices,
    get_departments_starting_with_and_computers
)

class TestRK1Functions(unittest.TestCase):

    def setUp(self):
        # Test data
        self.procs = [
            Processor(1, 'Intel Core i7'),
            Processor(2, 'AMD Ryzen'),
            Processor(3, 'Intel Pentium'),
            Processor(4, 'AMD Athlon'),
            Processor(5, 'Intel Xeon')
        ]

        self.comps = [
            Computer(1, 'Dell XPS', 1200, 1),
            Computer(2, 'HP Spectre', 1500, 2),
            Computer(3, 'Lenovo ThinkPad', 1000, 3),
            Computer(4, 'Apple MacBook', 2000, 4),
            Computer(5, 'Asus', 200, 5),
            Computer(6, 'Acer Aspire', 800, 2)
        ]
```

```
        self.dev_depts = [
            DevDept(1, 'Architecture Dept'),
            DevDept(2, 'Advanced Systems'),
            DevDept(3, 'AI Research'),
            DevDept(4, 'Analytics Team')
        ]

        self.comps_devs = [
            CompDevDept(1, 1),
            CompDevDept(1, 5),
            CompDevDept(2, 2),
            CompDevDept(2, 3),
            CompDevDept(3, 4),
            CompDevDept(4, 6)
        ]

    def test_get_computers_with_processors_ending_with(self):
        result = get_computers_with_processors_ending_with(self.procs, self.comps,
'on')
        expected = [
            ('Apple MacBook', 'AMD Athlon'),
            ('Asus', 'Intel Xeon')
        ]
        self.assertEqual(set(result), set(expected))  # Compare as sets to ignore order

    def test_get_departments_with_average_prices(self):
        result = get_departments_with_average_prices(self.dev_depts,
self.comps_devs, self.comps)
```

```
        expected = [
            ('Architecture Dept', (1200 + 200) / 2),
            ('Advanced Systems', (1500 + 1000) / 2),
            ('AI Research', 2000),
            ('Analytics Team', 800)
        ]
        self.assertEqual(result, expected)


    def test_get_departments_starting_with_and_computers(self):
        result = get_departments_starting_with_and_computers(self.dev_depts, self.comps_devs, self.comps, 'A')
        expected = [
            ('Dell XPS', 'Architecture Dept'),
            ('Asus', 'Architecture Dept'),
            ('HP Spectre', 'Advanced Systems'),
            ('Lenovo ThinkPad', 'Advanced Systems'),
            ('Apple MacBook', 'AI Research'),
            ('Acer Aspire', 'Analytics Team')
        ]
        self.assertEqual(set(result), set(expected))  # Compare as sets to ignore order



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

```
C:\Users\Urbech\Desktop\labs\RK2>python test_refactored_rk1.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK

C:\Users\Urbech\Desktop\labs\RK2>
```