МГТУ им Н.Э.Баумана

«Системы обработки информации и управления»

Отчет

«Рубежный контроль №2»

Дисциплина: Парадигмы и Конструкции Языков Программирования

Студент: Керимова Жанна Руслановна

Группа: ИУ5-31Б

Текст программы:

Изменённая программа РК1:

```
from operator import itemgetter
from typing import List, Tuple


class C:
    def __init__(self, id: int, model: str, ram: int, os_id: int):
        self.id = id
        self.model = model
        self.ram = ram
        self.os_id = os_id


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


class C_OS:
    def __init__(self, os_id: int, comp_id: int):
        self.os_id = os_id
        self.comp_id = comp_id


def one_to_many_mapping(comps: List[C], oski: List[OS]) -> List[Tuple[str, int,
str]]:
    return [(c.model, c.ram, o.name)
            for o in oski
            for c in comps
            if c.os_id == o.id]


def many_to_many_mapping(comps: List[C], oski: List[OS], comps_os:
List[C_OS]) -> List[Tuple[str, int, str]]:
    many_to_many_temp = [(o.name, co.os_id, co.comp_id)
                         for o in oski
                         for co in comps_os
                         if o.id == co.os_id]
    return [(c.model, c.ram, os_name)
```

```
        for os_name, os_id, comp_id in many_to_many_temp
        for c in comps if c.id == comp_id]


def task_a1(one_to_many: List[Tuple[str, int, str]]) -> List[Tuple[str, int, str]]:
    return sorted(one_to_many, key=itemgetter(2))


def task_a2(one_to_many: List[Tuple[str, int, str]], oski: List[OS]) ->
List[Tuple[str, int]]:
    res_12_unsorted = []
    for o in oski:
        o_comps = list(filter(lambda i: i[2] == o.name, one_to_many))
        if len(o_comps) > 0:
            o_rams = [ram for _, ram, _ in o_comps]
            o_rams_sum = sum(o_rams)
            res_12_unsorted.append((o.name, o_rams_sum))
    return sorted(res_12_unsorted, key=itemgetter(1), reverse=False)


def task_a3(many_to_many: List[Tuple[str, int, str]], oski: List[OS]) -> dict:
    res_13 = {}
    for o in oski:
        if 'Windows' in o.name or 'macOS' in o.name:
            o_comps = list(filter(lambda i: i[2] == o.name, many_to_many))
            o_comps_models = [x for x, _, _ in o_comps]
            res_13[o.name] = o_comps_models
    return res_13
```

Тесты:

```
import unittest
from RK2 import C, OS, C_OS, one_to_many_mapping,
many_to_many_mapping, task_a1, task_a2, task_a3

class TestComputerOSFunctions(unittest.TestCase):

    def setUp(self):
        self.Oski = [
            OS(1, 'Windows 10'),
            OS(2, 'Linux'),
```

```
            OS(3, 'macOS'),
            OS(11, 'Windows 11'),
            OS(22, 'Ubuntu'),
            OS(33, 'macOS Big Sur'),
        ]

        self.Comps = [
            C(1, 'Dell XPS 13', 16, 1),
            C(2, 'MacBook Air', 32, 3),
            C(3, 'Lenovo YOGA', 8, 2),
            C(4, 'HP Spectre', 16, 3),
            C(5, 'Asus TUF GAMING F15', 32, 2),
        ]

        self.comps_os = [
            C_OS(1, 1),
            C_OS(2, 2),
            C_OS(3, 3),
            C_OS(3, 4),
            C_OS(3, 5),
            C_OS(11, 1),
            C_OS(22, 2),
            C_OS(33, 3),
            C_OS(33, 4),
            C_OS(33, 5),
        ]

    def test_one_to_many_mapping(self):
        result = one_to_many_mapping(self.Comps, self.Oski)
        expected = [
            ('Dell XPS 13', 16, 'Windows 10'),
            ('MacBook Air', 32, 'macOS'),
            ('Lenovo YOGA', 8, 'Linux'),
            ('HP Spectre', 16, 'macOS'),
            ('Asus TUF GAMING F15', 32, 'Linux'),
        ]
        self.assertEqual(result, expected)

    def test_task_a2(self):
        one_to_many = one_to_many_mapping(self.Comps, self.Oski)
        result = task_a2(one_to_many, self.Oski)
        expected = [
```

```
            ('Linux', 40),
            ('macOS', 56),
            ('Windows 10', 16)
        ]
        self.assertEqual(result, expected)

    def test_task_a3(self):
        many_to_many = many_to_many_mapping(self.Comps, self.Oski,
self.comps_os)
        result = task_a3(many_to_many, self.Oski)
        expected = {
            'Windows 10': ['Dell XPS 13'],
            'macOS': ['MacBook Air', 'HP Spectre'],
            'macOS Big Sur': ['Asus TUF GAMING F15']
        }
        self.assertEqual(result, expected)

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

Результат работы программы:

Testing started at 0:09 ...

Launching unittests with arguments python -m unittest C:\Users\zhann\PycharmProjects\pythonBMSTU2COURSE\RK2tests.py in C:\Users\zhann\PycharmProjects\pythonBMSTU2COURSE

[('Dell XPS 13', 16, 'Windows 10'),

('MacBook Air', 32, 'macOS'),

('Lenovo YOGA', 8, 'Linux'),

('HP Spectre', 16, 'macOS'),

('Asus TUF GAMING F15', 32, 'Linux')] != [('Dell XPS 13', 16, 'Windows 10'),

('Lenovo YOGA', 8, 'Linux'),

('Asus TUF GAMING F15', 32, 'Linux'),

('MacBook Air', 32, 'macOS'),

('HP Spectre', 16, 'macOS')]

<Click to see difference>

Traceback (most recent call last):

File "C:\Users\zhann\PycharmProjects\pythonBMSTU2COURSE\RK2tests.py", line 46, in test_one_to_many_mapping

self.assertEqual(result, expected)

AssertionError: Lists differ: [('De[28 chars]), ('Lenovo YOGA', 8, 'Linux'), ('Asus TUF GAM[77 chars]OS')] != [('De[28 chars]), ('MacBook Air', 32, 'macOS'), ('Lenovo YOGA[77 chars]ux')]

First differing element 1:

('Lenovo YOGA', 8, 'Linux')

('MacBook Air', 32, 'macOS')

[('Dell XPS 13', 16, 'Windows 10'),

+ ('MacBook Air', 32, 'macOS'),

('Lenovo YOGA', 8, 'Linux'),

+ ('HP Spectre', 16, 'macOS'),

- ('Asus TUF GAMING F15', 32, 'Linux'),

? ^

+ ('Asus TUF GAMING F15', 32, 'Linux')]

? ^

- ('MacBook Air', 32, 'macOS'),

- ('HP Spectre', 16, 'macOS')]

[('Linux', 40), ('macOS', 56), ('Windows 10', 16)] != [('Windows 10', 16), ('Linux', 40), ('macOS', 48)]

Expected :[('Windows 10', 16), ('Linux', 40), ('macOS', 48)]

Actual   :[('Linux', 40), ('macOS', 56), ('Windows 10', 16)]

<Click to see difference>

Traceback (most recent call last):

  File "C:\Users\zhann\PycharmProjects\pythonBMSTU2COURSE\RK2tests.py", line 56, in test_task_a2

    self.assertEqual(result, expected)

AssertionError: Lists differ: [('Windows 10', 16), ('Linux', 40), ('macOS', 48)] != [('Linux', 40), ('macOS', 56), ('Windows 10', 16)]

First differing element 0:

('Windows 10', 16)

('Linux', 40)

- [('Windows 10', 16), ('Linux', 40), ('macOS', 48)]

+ [('Linux', 40), ('macOS', 56), ('Windows 10', 16)]

{'Windows 10': ['Dell XPS 13'],

'macOS': ['MacBook Air', 'HP Spectre'],

'macOS Big Sur': ['Asus TUF GAMING F15']} != {'Windows 10': ['Dell XPS 13'],

'Windows 11': ['Dell XPS 13'],

'macOS': ['Lenovo YOGA', 'HP Spectre', 'Asus TUF GAMING F15'],

'macOS Big Sur': ['Lenovo YOGA', 'HP Spectre', 'Asus TUF GAMING F15']}

<Click to see difference>

Traceback (most recent call last):

File "C:\Users\zhann\PycharmProjects\pythonBMSTU2COURSE\RK2tests.py", line 66, in test_task_a3

self.assertEqual(result, expected)

AssertionError: {'Win[33 chars]': ['Lenovo YOGA', 'HP Spectre', 'Asus TUF GAM[107 chars]15']} != {'Win[33 chars]': ['MacBook Air', 'HP Spectre'], 'macOS Big S[24 chars]15']}

{'Windows 10': ['Dell XPS 13'],

+ 'macOS': ['MacBook Air', 'HP Spectre'],

+ 'macOS Big Sur': ['Asus TUF GAMING F15']}

- 'Windows 11': ['Dell XPS 13'],

- 'macOS': ['Lenovo YOGA', 'HP Spectre', 'Asus TUF GAMING F15'],

- 'macOS Big Sur': ['Lenovo YOGA', 'HP Spectre', 'Asus TUF GAMING F15']}

Ran 3 tests in 0.011s