Угрюмов Михаил, ИУ5-34Б, вариант – 23(Г)

Рубежный контроль №2

Текст программы:

main.py

```
class Language:
    ''' язык программирования '''
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Library:
    ''' библиотека языка программирования '''
    def __init__(self, id, name, func_num, lang_id):
        ''' func_num - количество функций в библиотеке'''
        self.id = id
        self.name = name
        self.func_num = func_num
        self.lang_id = lang_id

class Link:
    ''' связь библиотеки с языком '''
    def __init__(self, lib_id, lang_id):
        self.lib_id = lib_id
        self.lang_id = lang_id

def get_db_links(langs, libs, links):
    ''' связь один-ко-многим'''
    one_to_many = [(lib.name, lib.func_num, lang.name)
                   for lib in libs
                   for lang in langs
                   if lib.lang_id == lang.id]

    many_to_many_temp = [(lang.name, link.lang_id, link.lib_id)
                         for lang in langs
                         for link in links
                         if lang.id == link.lang_id]

    ''' связь много-ко-многим'''
    many_to_many = [(lib.name, lib.func_num, lang_name)
                    for lang_name, lang_id, lib_id in many_to_many_temp
                    for lib in libs if lib.id == lib_id]

    return (one_to_many, many_to_many)

def task_1(one_to_many, symbol):
    ''' returns languages which starts with symbol and all it`s libraries'''
    ans = {}
    for lib_name, x, lang_name in one_to_many:
        if lang_name[0] == symbol:
            if lang_name in ans:
                ans[lang_name].append(lib_name)
            else:
                ans[lang_name] = [lib_name]
    return ans

def task_2(one_to_many):
    ''' returns languages with max functions numbers in one library of each
language
        languages are sorted by these numbers'''
    ans = {}
    for x, func_num, lang_name in one_to_many:
```

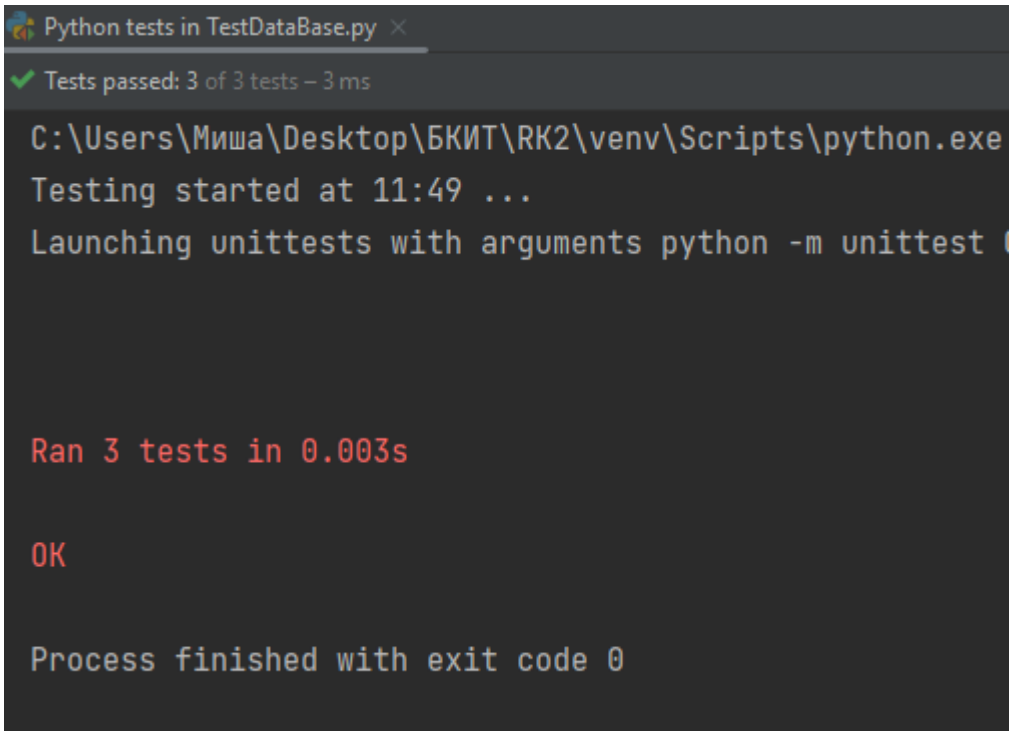
```
        if lang_name in ans:
            ans[lang_name] = max(ans[lang_name], func_num)
        else:
            ans[lang_name] = func_num
    ans = {key: value for key, value in sorted(ans.items(), key=lambda item:
item[1])}
    return ans

def task_3(many_to_many):
    ''' returns list of pairs (language, library)
        languages are sorted by names
        libraries in one language are not sorted'''
    ans = []
    for lib_name, x, lang_name in many_to_many:
        ans.append((lang_name, lib_name))
    ans = sorted(ans, key=lambda item: item[0])
    return ans

def main():
    langs = [
        Language(1, "Python"),  Language(10, "Python v0"),
        Language(2, "C++"),     Language(20, "C++ v0"),
        Language(3, "Pascal"),  Language(30, "Pascal v0"),
    ]

    libs = [
        Library(11, "random", 30, 1),
        Library(12, "math", 50, 1),
        Library(21, "vector", 40, 2),
        Library(22, "algorithm", 20, 2),
        Library(31, "Graph", 10, 3)
    ]

    links = [
        Link(11, 1),    Link(11, 10),
        Link(12, 1),    Link(12, 10),
        Link(21, 2),    Link(21, 20),
        Link(22, 2),    Link(22, 20),
        Link(31, 3),    Link(31, 30)
    ]

    one_to_many, many_to_many = get_db_links(langs, libs, links)

    print('\nЗадание 1')
    ans_1 = task_1(one_to_many, 'P')
    print(*ans_1.items())

    print('\nЗадание 2')
    ans_2 = task_2(one_to_many)
    print(*ans_2.items())

    print('\nЗадание 3')
    ans_3 = task_3(many_to_many)
    print(*ans_3)

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

TestDataBase.py

```
import unittest as ut
from main import *

class TestDB(ut.TestCase):
    def setUp(self):
        self.langs = [
            Language(1, "Python"),  Language(10, "Python v0"),
            Language(2, "C++"),     Language(20, "C++ v0"),
        ]

        self.libs = [
            Library(11, "random", 30, 1),
            Library(12, "math", 50, 1),
            Library(21, "vector", 40, 2)
        ]

        self.links = [
            Link(11, 1),    Link(11, 10),
            Link(12, 1),    Link(12, 10),
            Link(21, 2),    Link(21, 20)
        ]

        self.one_to_many, self.many_to_many = get_db_links(self.langs,
self.libs, self.links)

        self.ans_1 = {'C++' : ['vector']}
        self.ans_2 = {'C++' : 40, 'Python' : 50}
        self.ans_3 = [('C++', 'vector'), ('C++ v0', 'vector'), ('Python',
'random'), ('Python', 'math'), ('Python v0', 'random'), ('Python v0', 'math')]

    def test_task_1(self):
        self.assertEqual(task_1(self.one_to_many, 'C'), self.ans_1)

    def test_task_2(self):
        self.assertEqual(task_2(self.one_to_many), self.ans_2)

    def test_task_3(self):
        self.assertEqual(task_3(self.many_to_many), self.ans_3)

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

Результаты работы программы:

TestDataBase.py



main.py

В текстовом виде:

Задание 1

('Python', ['random', 'math']) ('Pascal', ['Graph'])

Задание 2

('Pascal', 10) ('C++', 40) ('Python', 50)

Задание 3

('C++', 'vector') ('C++', 'algorithm') ('C++ v0', 'vector') ('C++ v0', 'algorithm') ('Pascal', 'Graph') ('Pascal v0', 'Graph') ('Python', 'random') ('Python', 'math') ('Python v0', 'random') ('Python v0', 'math')

Process finished with exit code 0