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

**Рубежный контроль №2**

**Текст программы:**

main.py

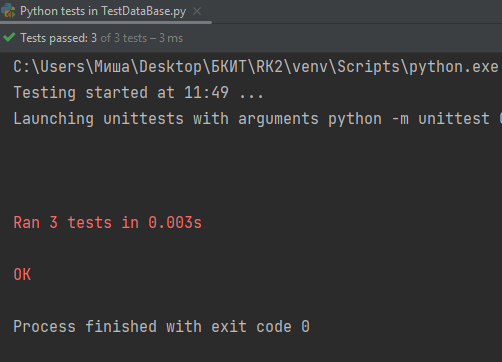
class Language:  
 *''' язык программирования '''* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
class Library:  
 *''' библиотека языка программирования '''* def \_\_init\_\_(self, id, name, func\_num, lang\_id):  
 *''' func\_num - количество функций в библиотеке'''* self.id = 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

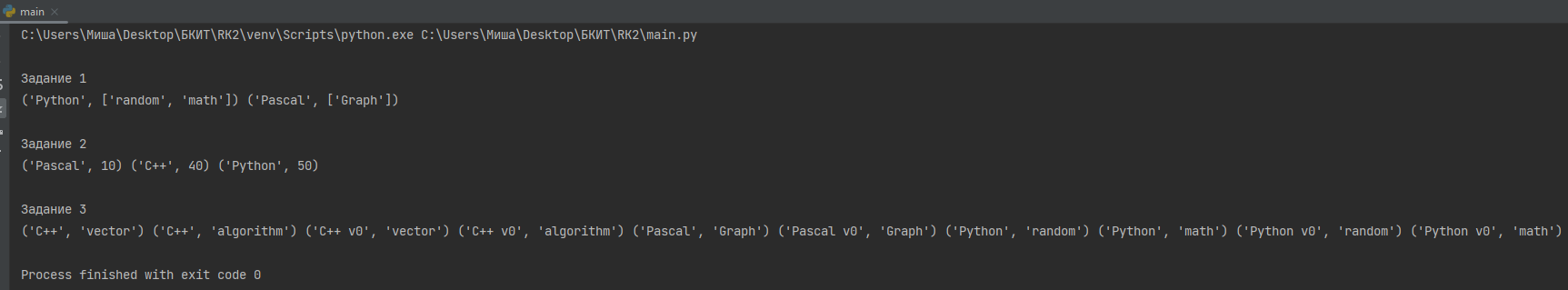
import unittest as ut  
from main import \*  
  
class TestDB(ut.TestCase):  
 def setUp(self):  
 self.langs = [  
 Language(1, "Python"), Language(10, "Python v0"),  
 Language(2, "C++"), Language(20, "C++ v0"),  
 ]  
  
 self.libs = [  
 Library(11, "random", 30, 1),  
 Library(12, "math", 50, 1),  
 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