МГТУ им. Н.Э. Баумана

Отчёт по РК2  
по курсу «ПиКЯП»

Студент группы ИУ5-31Б  
Крюков В. А.

2024 г.

Код программы:

from operator import itemgetter  
import unittest  
  
class Program:  
 def \_\_init\_\_(self, id, name, version, comp\_id):  
 self.id = id  
 self.name = name  
 self.version = version  
 self.comp\_id = comp\_id  
  
class Computer:  
 def \_\_init\_\_(self, id, model):  
 self.id = id  
 self.model = model  
  
class ProgramComputer:  
 def \_\_init\_\_(self, comp\_id, program\_id):  
 self.comp\_id = comp\_id  
 self.program\_id = program\_id  
  
*# Данные*computers = [  
 Computer(1, 'Macbook Pro M2'),  
 Computer(2, 'Lenovo ThinkPad 1'),  
 Computer(3, 'Asus E210'),  
 Computer(11, 'Lenovo Yoga Slim 7x'),  
 Computer(22, 'Macbook Pro M1'),  
 Computer(33, 'Asus X510'),  
]  
  
programs = [  
 Program(1, 'Microsoft Ofiice', 2012, 1),  
 Program(2, 'Adobe Photoshop', 2021, 2),  
 Program(3, 'GoogleChrome', 2023, 3),  
 Program(4, 'Visual Studio', 2022, 3),  
 Program(5, 'Intellij IDEA', 2024, 1),  
]  
  
programs\_computers = [  
 ProgramComputer(1, 1),  
 ProgramComputer(1, 5),  
 ProgramComputer(2, 2),  
 ProgramComputer(3, 3),  
 ProgramComputer(3, 4),  
 ProgramComputer(2, 1),  
 ProgramComputer(11, 3),  
 ProgramComputer(22, 3),  
 ProgramComputer(33, 3),  
]  
  
def get\_one\_to\_many(computers, programs):  
 return [(p.name, p.version, c.model)  
 for c in computers  
 for p in programs  
 if p.comp\_id == c.id]  
  
def get\_many\_to\_many(computers, programs, programs\_computers):  
 many\_to\_many\_temp = [(c.model, pc.comp\_id, pc.program\_id)  
 for c in computers  
 for pc in programs\_computers  
 if c.id == pc.comp\_id]  
  
 return [(p.name, p.version, comp\_model)  
 for comp\_model, comp\_id, program\_id in many\_to\_many\_temp  
 for p in programs  
 if p.id == program\_id]  
  
def task\_a1(one\_to\_many):  
 return sorted(one\_to\_many, key=itemgetter(2))  
  
def task\_a2(one\_to\_many, computers):  
 res = []  
 for c in computers:  
 c\_progs = list(filter(lambda i: i[2] == c.model, one\_to\_many))  
 if c\_progs:  
 versions = [ver for \_, ver, \_ in c\_progs]  
 res.append((c.model, max(versions)))  
  
 return sorted(res, key=itemgetter(1), reverse=True)  
  
def task\_a3(many\_to\_many, computers):  
 res = {}  
 for c in computers:  
 if 'Pro' in c.model:  
 c\_progs = list(filter(lambda i: i[2] == c.model, many\_to\_many))  
 prog\_names = [x for x, \_, \_ in c\_progs]  
 res[c.model] = prog\_names  
  
 return res  
  
*# Модульные тесты*class TestTasks(unittest.TestCase):  
 def setUp(self):  
 self.one\_to\_many = get\_one\_to\_many(computers, programs)  
 self.many\_to\_many = get\_many\_to\_many(computers, programs, programs\_computers)  
  
 def test\_task\_a1(self):  
 expected = [('GoogleChrome', 2023, 'Asus E210'),  
 ('Visual Studio', 2022, 'Asus E210'),  
 ('Adobe Photoshop', 2021, 'Lenovo ThinkPad 1'),  
 ('Microsoft Ofiice', 2012, 'Macbook Pro M2'),  
 ('Intellij IDEA', 2024, 'Macbook Pro M2')]  
 result = task\_a1(self.one\_to\_many)  
 self.assertEqual(result[:5], expected)  
  
 def test\_task\_a2(self):  
 expected = [('Macbook Pro M2', 2024), ('Asus E210', 2023), ('Lenovo ThinkPad 1', 2021)]  
 result = task\_a2(self.one\_to\_many, computers)  
 self.assertEqual(result[:3], expected)  
  
 def test\_task\_a3(self):  
 expected = {  
 'Macbook Pro M2': ['Microsoft Ofiice', 'Intellij IDEA'],  
 'Macbook Pro M1': ['GoogleChrome']  
 }  
 result = task\_a3(self.many\_to\_many, computers)  
 self.assertEqual({k: v for k, v in result.items() if 'Pro' in k}, expected)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Результат:

