

Рубежный контроль 2

Условия рубежного контроля №2 по курсу ПиК ЯП

Рубежный контроль представляет собой разработку тестов на языке Python.

- 1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
- 2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы:

1. Main.py:

```
from operator import itemgetter
```

```
class ProgLang:
```

```
    def __init__(self, id, name):
```

```
        self.id = id
```

```
        self.name = name
```

```
class DevEnv:
```

```
    def __init__(self, id, name, year, lang_id):
```

```
        self.id = id
```

```
        self.name = name
```

```
        self.year = year
```

```
        self.lang_id = lang_id
```

```
class ProgEnv:
```

```
    def __init__(self, lang_id, env_id):
```

```
        self.lang_id = lang_id
```

```
        self.env_id = env_id
```

```
def create_data():  
    prog_langs = [  
        ProgLang(1, "Python"),  
        ProgLang(2, "C++"),  
        ProgLang(3, "Rust"),  
    ]  
  
    dev_envs = [  
        DevEnv(1, "Pycharm", 2013, 1),  
        DevEnv(2, "Visual Studio", 1997, 2),  
        DevEnv(3, "Visual Studio Code", 2015, 3),  
        DevEnv(4, "CLion", 2014, 1),  
        DevEnv(5, "QT Creator", 1991, 2),  
        DevEnv(6, "Vim", 1988, 2),  
    ]  
  
    prog_envs = [  
        ProgEnv(1, 1),  
        ProgEnv(2, 2),  
        ProgEnv(3, 3),  
        ProgEnv(1, 4),  
        ProgEnv(2, 5),  
        ProgEnv(2, 6),  
    ]  
    return prog_langs, dev_envs, prog_envs
```

```
def task1(env_list):
```

```
res_11 = sorted(env_list, key=itemgetter(1))  
return res_11
```

```
def task2(env_list):
```

```
    res_12 = []
```

```
    temp_dict = {}
```

```
    for object in env_list:
```

```
        if object[2] in temp_dict:
```

```
            temp_dict[object[2]] += 1
```

```
        else:
```

```
            temp_dict[object[2]] = 1
```

```
    for object, count in temp_dict.items():
```

```
        res_12.append((object, count))
```

```
    res_12.sort(key=itemgetter(1))
```

```
    return res_12
```

```
def task3(env_list):
```

```
    res_13 = [(env_name, lang_name) for env_name, _, lang_name in env_list if _  
< 2014]
```

```
    return res_13
```

```
def generate_relationships(prog_langs, dev_envs, prog_envs):
```

```
    one_to_many = [(env.name, env.year, lang.name)
```

```
                    for lang in prog_langs
```

```
                    for env in dev_envs
```

```
                    if env.lang_id == lang.id]
```

```
many_to_many_temp = [(lang.name, le.lang_id, le.env_id)
                      for lang in prog_langs
                      for le in prog_envs
                      if le.lang_id == lang.id]

many_to_many = [(env.name, env.year, lang_name)
                 for lang_name, lang_id, env_id in many_to_many_temp
                 for env in dev_envs if env.id == env_id]

return one_to_many, many_to_many
```

```
def main():
```

```
    prog_langs, dev_envs, prog_envs = create_data()
```

```
    one_to_many, many_to_many = generate_relationships(prog_langs, dev_envs,
prog_envs)
```

```
    print("Задание Б1:")
```

```
    print(task1(one_to_many))
```

```
    print("Задание Б2:")
```

```
    print(task2(one_to_many))
```

```
    print("Задание Б3:")
```

```
    print(task3(many_to_many))
```

```
if __name__ == '__main__':
```

```
    main()
```

2. Unit_tests.py:

```
import unittest
```

```
import main
```

```
class TestTaskFunctions(unittest.TestCase):
```

```
    def test_task1(self):
```

```
        in_data = (('Pycharm', 2013, 'Python'), ('CLion', 2014, 'Python'), ('Visual  
Studio', 1997, 'C++'), ('QT Creator', 1991, 'C++'), ('Vim', 1988, 'C++'), ('Visual  
Studio Code', 2015, 'Rust'))
```

```
        correct_out = [('Vim', 1988, 'C++'), ('QT Creator', 1991, 'C++'), ('Visual  
Studio', 1997, 'C++'), ('Pycharm', 2013, 'Python'), ('CLion', 2014, 'Python'),  
('Visual Studio Code', 2015, 'Rust')]
```

```
        out = main.task1(in_data)
```

```
        self.assertEqual(out, correct_out)
```

```
    def test_task2(self):
```

```
        in_data = (('Pycharm', 2013, 'Python'), ('CLion', 2014, 'Python'), ('Visual  
Studio', 1997, 'C++'), ('QT Creator', 1991, 'C++'), ('Vim', 1988, 'C++'), ('Visual  
Studio Code', 2015, 'Rust'))
```

```
        correct_out = [('Rust', 1), ('Python', 2), ('C++', 3)]
```

```
        out = main.task2(in_data)
```

```
        self.assertEqual(out, correct_out)
```

```
    def test_task3(self):
```

```
        in_data = (('Pycharm', 2013, 'Python'), ('CLion', 2014, 'Python'), ('Visual  
Studio', 1997, 'C++'), ('QT Creator', 1991, 'C++'), ('Vim', 1988, 'C++'), ('Visual  
Studio Code', 2015, 'Rust'))
```

```
        correct_out = [('Pycharm', 'Python'), ('Visual Studio', 'C++'), ('QT Creator',  
'C++'), ('Vim', 'C++')]
```

```
        out = main.task3(in_data)
```

```
        self.assertEqual(out, correct_out)
```

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

Результаты выполнения:

...

Ran 3 tests in 0.000s

OK