

## Вариант 23. Сысоев Иван ИУ5-31Б

### Код main.py

```
from operator import itemgetter
```

```
class Syntax:
```

```
    """Синтаксическая конструкция"""
```

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

```
        self.id = id
```

```
        self.name = name
```

```
        self.execution_time = execution_time
```

```
        self.lang_id = lang_id
```

```
class Language:
```

```
    """Язык программирования"""
```

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

```
        self.id = id
```

```
        self.name = name
```

```
class SyntaxLang:
```

```
    """Связь синтаксических конструкций и языков программирования для отношения многие-ко-многим"""
```

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

```
        self.lang_id = lang_id
```

```
        self.syntax_id = syntax_id
```

```
# Языки программирования
```

```
langs = [  
    Language(1, 'Python'),  
    Language(2, 'Java'),  
    Language(3, 'C++'),  
    Language(4, 'C')  
]
```

```
# Синтаксические конструкции
```

```
syntaxes = [  
    Syntax(1, 'Функция', 0.04, 1),  
    Syntax(2, 'Цикл do while', 0.01, 3),  
    Syntax(3, 'Условный оператор if', 0.01, 3),  
    Syntax(4, 'Цикл for', 0.02, 4),  
    Syntax(5, 'Класс', 0.05, 2)  
]
```

```
# Связи многие-ко-многим
```

```
syntax_langs = [  
    SyntaxLang(1, 1),  
    SyntaxLang(1, 3),  
    SyntaxLang(1, 4),  
    SyntaxLang(1, 5),  
    SyntaxLang(2, 1),  
    SyntaxLang(2, 2),  
    SyntaxLang(2, 3),  
    SyntaxLang(2, 4),  
    SyntaxLang(2, 5),  
]
```

```
SyntaxLang(3, 1),
SyntaxLang(3, 2),
SyntaxLang(3, 3),
SyntaxLang(3, 4),
SyntaxLang(3, 5),
SyntaxLang(4, 1),
SyntaxLang(4, 2),
SyntaxLang(4, 3),
SyntaxLang(4, 4)
]
```

# Функции

```
def get_sorted_syntax_by_language():
```

```
    """Задание A1: список связанных конструкций и языков, отсортированный по конструкциям"""
```

```
    one_to_many = [(s.name, s.execution_time, l.name) for l in langs for s in syntaxes if s.lang_id == l.id]
```

```
    return sorted(one_to_many, key=itemgetter(0))
```

```
def get_languages_by_execution_time():
```

```
    """Задание A2: список языков с суммарным временем выполнения конструкций,
отсортированный по времени"""
```

```
    one_to_many = [(s.name, s.execution_time, l.name) for l in langs for s in syntaxes if s.lang_id == l.id]
```

```
    result = []
```

```
    for l in langs:
```

```
        l_syntaxes = list(filter(lambda i: i[2] == l.name, one_to_many))
```

```
        if l_syntaxes:
```

```
            l_execution_times = [time for _, time, _ in l_syntaxes]
```

```
            result.append((l.name, sum(l_execution_times)))
```

```

return sorted(result, key=itemgetter(1), reverse=True)

def get_languages_with_c_and_syntax():
    """Задание А3: список всех языков с "C" в названии и их синтаксических конструкций"""
    many_to_many_temp = [(l.name, sl.lang_id, sl.syntax_id) for l in langs for sl in syntax_langs if l.id ==
sl.lang_id]
    many_to_many = [(s.name, s.execution_time, lang_name)
                     for lang_name, lang_id, syntax_id in many_to_many_temp
                     for s in syntaxes if s.id == syntax_id]
    result = {}
    for l in langs:
        if 'C' in l.name:
            l_syntaxes = list(filter(lambda i: i[2] == l.name, many_to_many))
            result[l.name] = [x for x, _, _ in l_syntaxes]
    return result

```

## Код test.py

```

from main import *

# Тесты

import unittest

class TestSyntaxFunctions(unittest.TestCase):
    def test_get_sorted_syntax_by_language(self):
        result = get_sorted_syntax_by_language()

```

```
expected = [  
    ('Класс', 0.05, 'Java'),  
    ('Условный оператор if', 0.01, 'C++'),  
    ('Функция', 0.04, 'Python'),  
    ('Цикл do while', 0.01, 'C++'),  
    ('Цикл for', 0.02, 'C')  
]
```

```
self.assertEqual(result, expected)
```

```
def test_get_languages_by_execution_time(self):  
    result = get_languages_by_execution_time()  
    expected = [  
        ('Java', 0.05),  
        ('Python', 0.04),  
        ('C++', 0.02),  
        ('C', 0.02)  
    ]
```

```
self.assertEqual(result, expected)
```


```
def test_get_languages_with_c_and_syntax(self):  
    result = get_languages_with_c_and_syntax()  
    expected = {  
        'C++': ['Функция', 'Цикл do while', 'Условный оператор if', 'Цикл for',  
        'Класс'],
```

```
        'C': ['Функция', 'Цикл do while', 'Условный оператор if', 'Цикл for']
    }

    self.assertEqual(result, expected)
```

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

### Результат:

A terminal window with a dark background. The title bar shows two tabs: 'ivan@ivan-B650M-D3HP: ~/prog/test' and 'ivan@ivan-B650M-D3HP: ~/prog/labs\_3\_PIKAP/RK2'. The active tab is the second one. The terminal content shows the command 'python3 test.py' being executed. The output consists of three dots, a dashed line, and the text 'Ran 3 tests in 0.000s'. At the bottom, the word 'OK' is displayed. The terminal window has standard Linux window controls (minimize, maximize, close) on the right side.

```
ivan@ivan-B650M-D3HP: ~/prog/test
ivan@ivan-B650M-D3HP: ~/prog/labs_3_PIKAP/RK2$ python3 test.py
...
-----
Ran 3 tests in 0.000s
OK
```