Текст программы

main.py

```python
from operator import itemgetter

class SyntaxConstruction:
    """Синтаксическая конструкция языка программирования"""

    def __init__(self, id, name, usage_frequency, lang_id):
        self.id = id
        self.name = name
        self.usage_frequency = usage_frequency
        self.lang_id = lang_id


class ProgrammingLanguage:
    """Язык программирования"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class LanguageConstruction:
    """Синтаксические конструкции языка программирования для реализации связи многие-
ко-многим"""

    def __init__(self, lang_id, construct_id):
        self.lang_id = lang_id
        self.construct_id = construct_id


# Языки программирования
langs = [
    ProgrammingLanguage(1, 'Python'),
    ProgrammingLanguage(2, 'JavaScript Language'),
    ProgrammingLanguage(3, 'Java Language'),
    ProgrammingLanguage(4, 'C++'),
    ProgrammingLanguage(5, 'Ruby Language'),
]

# Синтаксические конструкции
constructs = [
    SyntaxConstruction(1, 'if-else', 8, 1),  # Python
    SyntaxConstruction(2, 'for', 5, 1),  # Python
    SyntaxConstruction(3, 'function', 2, 2),  # JavaScript
    SyntaxConstruction(4, 'class', 9, 3),  # Java
    SyntaxConstruction(5, 'while', 6, 4),  # C++
    SyntaxConstruction(6, 'def', 3, 5),  # Ruby
]

# Связь многие-ко-многим между языками программирования и синтаксическими конструкциями
lang_construct = [
    LanguageConstruction(1, 1),  # Python - if-else
    LanguageConstruction(1, 2),  # Python - for
    LanguageConstruction(2, 3),  # JavaScript - function
    LanguageConstruction(3, 4),  # Java - class
    LanguageConstruction(4, 5),  # C++ - while
    LanguageConstruction(5, 6),  # Ruby - def
]
```

```python
def one_to_many(langs, constructs):
    # Соединение данных один-ко-многим
    return [
        (lang.name, construct.name, construct.usage_frequency)
        for lang in langs
        for construct in constructs
        if construct.lang_id == lang.id
    ]

def many_to_many(langs, lang_construct, constructs):
    # Соединение многие-ко-многим
    many_to_many_temp = [
        (lang.name, lang_part.lang_id, lang_part.construct_id)
        for lang in langs
        for lang_part in lang_construct
        if lang.id == lang_part.lang_id
    ]
    return [
        (construct.name, lang_name)
        for lang_name, lang_id, construct_id in many_to_many_temp
        for construct in constructs
        if construct.id == construct_id
    ]

def part1(word, one_to_many: list) -> list:
    print('\tЗадание Е1 (Вывод языков со словом "Language" в названии)')
    return [x for x in one_to_many if word in x[0]]

def part2(one_to_many: list) -> list:
    print('\tЗадание Е2 (Вывод языков по убыванию средней встречаемости конструкций)')
    lang_ufs = {}
    for lang in langs:
        lang_ufs[lang.name] = []
    for row in one_to_many:
        lang_name, _, usage_frequency = row
        lang_ufs[lang_name].append(usage_frequency)
    res2 = [(lang, round(sum(usage_frequency) / len(usage_frequency), 2))
            for lang, usage_frequency in lang_ufs.items() if usage_frequency]
    return sorted(res2, key=itemgetter(1), reverse=True)

def part3(key, many_to_many: list) -> list:
    print('\tЗадание Е3 (Вывод конструкций на "f" и языков, в которых они 
встречаются)')
    return list(filter(lambda i: i[0][0] == key, many_to_many))

if __name__ == '__main__':
    word = "Language"
    key = "f"
    otm_data = one_to_many(langs, constructs)
    print(part1(word, otm_data))
    print(part2(otm_data))
    print(part3(key, many_to_many(langs, lang_construct, constructs)))
```

tests.py

```
import unittest

from main import *

class TestSolutions(unittest.TestCase):
    def setUp(self):
        self.constructs = [
            SyntaxConstruction(1, 'if-else', 8, 1),  # Python
            SyntaxConstruction(2, 'for', 5, 1),  # Python
            SyntaxConstruction(3, 'function', 2, 2),  # JavaScript
            SyntaxConstruction(4, 'class', 9, 3),  # Java
            SyntaxConstruction(5, 'while', 6, 4),  # C++
            SyntaxConstruction(6, 'def', 3, 5),  # Ruby
        ]
        self.langs = [
            ProgrammingLanguage(1, 'Python'),
            ProgrammingLanguage(2, 'JavaScript Language'),
            ProgrammingLanguage(3, 'Java Language'),
            ProgrammingLanguage(4, 'C++'),
            ProgrammingLanguage(5, 'Ruby Language'),
        ]
        self.lang_construct = [
            LanguageConstruction(1, 1),  # Python - if-else
            LanguageConstruction(1, 2),  # Python - for
            LanguageConstruction(2, 3),  # JavaScript - function
            LanguageConstruction(3, 4),  # Java - class
            LanguageConstruction(4, 5),  # C++ - while
            LanguageConstruction(5, 6),  # Ruby - def
        ]
        self.test_word = 'Language'
        self.test_letter = 'f'

    def test_part1(self):
        result = part1(self.test_word, one_to_many(self.langs, self.constructs))
        self.assertEqual(result,[('JavaScript Language', 'function', 2), ('Java
Language', 'class', 9), ('Ruby Language', 'def', 3)])

    def test_part2(self):
        result = part2(one_to_many(self.langs, self.constructs))
        self.assertEqual(result,[('Java Language', 9.0), ('Python', 6.5), ('C++', 6.0),
('Ruby Language', 3.0), ('JavaScript Language', 2.0)])

    def test_part3(self):
        result = part3(self.test_letter,many_to_many(self.langs, self.lang_construct,
self.constructs))
        self.assertEqual(result,[('for', 'Python'), ('function', 'JavaScript
Language')])
        input("Press any button...")

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

Результаты

```
      Задание Е1 (Вывод языков со словом "Language" в названии).
      Задание Е2 (Вывод языков по убыванию средней встречаемости конструкций).
      Задание Е3 (Вывод конструкций на "f" и языков, в которых они встречаются).
Press any button...
----------------------------------------------------------------------
Ran 3 tests in 0.004s

OK

Press any key to continue. . .
```