Текст программы

```
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'),
]
```

```python
# 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
]


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

    # 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
    ]

    many_to_many = [
```

```
        (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
    ]

    print('\tЗадание E1 (Вывод языков со словом "Language" в
названии)')
    result1 = [x for x in one_to_many if 'Language' in x[0]]
    print(result1)

    print('\n\tЗадание E2 (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]
    res2 = sorted(res2, key=itemgetter(1), reverse=True)
    print(res2)

    print('\n\tЗадание E3 (Вывод конструкций на "f" и языков, в
которых они встречаются)')
    res3 = list(filter(lambda i: i[0][0] == 'f', many_to_many))
    print(res3)


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

Результаты

Задание E1 (Вывод языков со словом "Language" в названии)

```
[('JavaScript Language', 'function', 2), ('Java Language',
'class', 9), ('Ruby Language', 'def', 3)]
```

Задание E2 (Вывод языков по убыванию средней встречаемости конструкций)

```
[('Java Language', 9.0), ('Python', 6.5), ('C++', 6.0), ('Ruby
Language', 3.0), ('JavaScript Language', 2.0)]
```

Задание E3 (Вывод конструкций на "f" и языков, в которых они встречаются)

```
[('for', 'Python'), ('function', 'JavaScript Language')]
```