# ТЕКСТ ПРОГРАММЫ

## *main*

```
from operator import itemgetter


# Определение класса Language для представления языков программирования
class Language:
    def __init__(self, id, name, complexity):
        self.id = id
        self.name = name
        self.complexity = complexity


# Определение класса Library для представления библиотек программного обеспечения
class Library:
    def __init__(self, id, name, language_id, symbols):
        self.id = id
        self.name = name
        self.language_id = language_id
        self.symbols = symbols


# Определение класса LanguageLibrary для представления связи между языками и библиотеками
class LanguageLibrary:
    def __init__(self, language_id, lib_id):
        self.language_id = language_id
        self.lib_id = lib_id


# Функция выполняет задание E1, создавая список соответствий языков и библиотек
def task_e1(languages, libs):
    return [(language.name, language.complexity, lib.name, lib.symbols)
            for language in languages
            for lib in libs
            if lib.language_id == language.id]


# Функция выполняет задание E2, вычисляя средний символ для каждого языка
def task_e2(languages, libs, one_to_many):
    language_symbols = {}
    for language in languages:
        language_symbols[language.name] = []
    for row in one_to_many:
        language_name, _, _, symbols = row
        language_symbols[language_name].append(symbols)
```

```
    res_12 = [(language, round(sum(symbols) / len(symbols), 2))
              for language, symbols in language_symbols.items() if symbols]
    res_12 = sorted(res_12, key=itemgetter(1), reverse=True)
    return res_12

# Функция выполняет задание E3, создавая список библиотек, начинающихся с буквы 'm'
def task_e3(languages, libs, language_libs):
    many_to_many_temp = [(language.name, language_lib.language_id, language_lib.lib_id)
                         for language_lib in language_libs
                         for language in languages
                         if language.id == language_lib.language_id]

    many_to_many = [(lib.name, language_name)
                    for language_name, language_id, lib_id in many_to_many_temp
                    for lib in libs if lib.id == lib_id]

    return list(filter(lambda i: i[0][0] == 'm', many_to_many))

# Основная функция main, где определены языки, библиотеки и связи, и вызываются функции заданий
def main():
    languages = [
        Language(1, 'Python', 'medium'),
        Language(2, 'C++', 'medium'),
        Language(3, 'C', 'easy'),
        Language(4, 'Swift', 'hard'),
        Language(5, 'JavaScript', 'medium'),
        Language(6, 'Java', 'medium'),
        Language(7, 'C#', 'medium'),
    ]

    libs = [
        Library(1, 'requests', 2, 100),
        Library(2, 'system', 2, 200),
        Library(3, 'numbers', 6, 250),
        Library(4, 'math', 1, 150),
        Library(5, 'menu', 1, 100),
        Library(6, 'play', 6, 50),
        Library(7, 'reject', 5, 250),
        Library(8, 'formuli', 3, 150),
        Library(9, 'picture', 4, 100),
        Library(10, 'css', 7, 200),
```

```
    ]

    language_libs = [
        LanguageLibrary(1, 4),
        LanguageLibrary(1, 5),
        LanguageLibrary(2, 1),
        LanguageLibrary(2, 2),
        LanguageLibrary(3, 8),
        LanguageLibrary(4, 9),
        LanguageLibrary(5, 7),
        LanguageLibrary(6, 6),
        LanguageLibrary(6, 3),
        LanguageLibrary(7, 10),
    ]

    # Задание E1: Вывод соответствий языков и библиотек
    one_to_many = task_e1(languages, libs)
    print('Задание E1')
    print(one_to_many)

    # Задание E2: Вывод среднего символа для каждого языка, отсортированного по убыванию
    res_12 = task_e2(languages, libs, one_to_many)
    print('Задание E2')
    print(res_12)

    # Задание E3: Вывод библиотек, начинающихся с буквы 'm'
    res_13 = task_e3(languages, libs, language_libs)
    print('Задание E3')
    print(res_13)

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

## ***tests***

```
import unittest
from main import Language, Library, LanguageLibrary, task_e1, task_e2, task_e3

class TestProgram(unittest.TestCase):
    def setUp(self):
        self.languages = [
```

```
        Language(1, 'Python', 'medium'),
        Language(2, 'C++', 'medium'),
        Language(3, 'C', 'easy'),
        Language(4, 'Swift', 'hard'),
        Language(5, 'JavaScript', 'medium'),
        Language(6, 'Java', 'medium'),
        Language(7, 'C#', 'medium'),
    ]

    self.libs = [
        Library(1, 'requests', 2, 100),
        Library(2, 'system', 2, 200),
        Library(3, 'numbers', 6, 250),
        Library(4, 'math', 1, 150),
        Library(5, 'menu', 1, 100),
        Library(6, 'play', 6, 50),
        Library(7, 'reject', 5, 250),
        Library(8, 'formuli', 3, 150),
        Library(9, 'picture', 4, 100),
        Library(10, 'css', 7, 200),
    ]

    self.language_libs = [
        LanguageLibrary(1, 4),
        LanguageLibrary(1, 5),
        LanguageLibrary(2, 1),
        LanguageLibrary(2, 2),
        LanguageLibrary(3, 8),
        LanguageLibrary(4, 9),
        LanguageLibrary(5, 7),
        LanguageLibrary(6, 6),
        LanguageLibrary(6, 3),
        LanguageLibrary(7, 10),
    ]

def test_task_e1(self):
    expected_result = [('C++', 'medium', 'requests', 100),
                       ('C++', 'medium', 'system', 200),
                       ('Java', 'medium', 'play', 50),
                       ('C#', 'medium', 'css', 200)]
```

```
        result = task_e1(self.languages, self.libs)
        self.assertEqual(result, expected_result)

    def test_task_e2(self):
        one_to_many = task_e1(self.languages, self.libs)
        expected_result = [('Java', 100.0),
                           ('C++', 75.0),
                           ('C#', 62.5),
                           ('Python', 62.5),
                           ('JavaScript', 62.5),
                           ('C', 50.0),
                           ('Swift', 0.0)]

        result = task_e2(self.languages, self.libs, one_to_many)
        self.assertEqual(result, expected_result)

    def test_task_e3(self):
        expected_result = [('math', 'JavaScript'),
                           ('menu', 'JavaScript'),
                           ('play', 'Java'),
                           ('reject', 'JavaScript'),
                           ('css', 'C#')]

        result = task_e3(self.languages, self.libs, self.language_libs)
        self.assertEqual(result, expected_result)

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

***Результаты выполнения:***