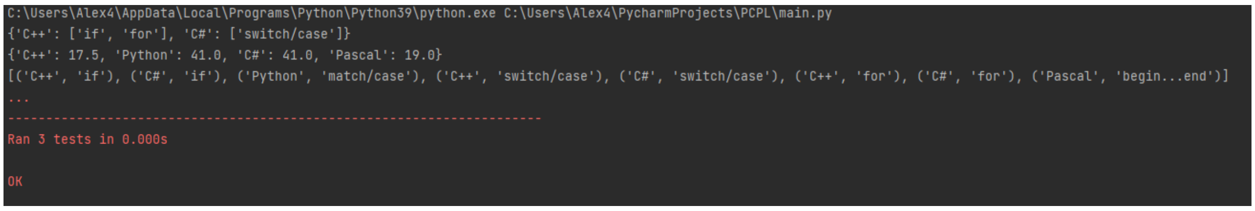
Code:

```python
import unittest


class ProgrammingLanguage:  # язык программирования

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


class SyntacticConstruction:  # синтаксическая конструкция

    def __init__(self, id, name, description, PL_id=None):
        self.id = id
        self.name = name
        self.description = description
        self.PL_id = PL_id


class ProgrammingLanguageToSyntacticConstruction:  # связь многие ко многим

    def __init__(self, PL_id, SC_id):
        self.PL_id = PL_id
        self.SC_id = SC_id


languages = [
    ProgrammingLanguage(1, 'C++'),
    ProgrammingLanguage(2, 'Python'),
    ProgrammingLanguage(3, 'C#'),
    ProgrammingLanguage(4, 'Pascal'),
]

constructions = [
    SyntacticConstruction(1, 'if', 'Выбор выполнения', 1),
    SyntacticConstruction(2, 'match/case', 'Выбор выполнения с несколькими
вариантами', 2),
    SyntacticConstruction(3, 'switch/case', 'Выбор выполнения с несколькими
вариантами', 3),
    SyntacticConstruction(4, 'for', 'Повторение операций', 1),
    SyntacticConstruction(5, 'begin...end', 'Повторение операций', 4),
]

languages_constructions = [
    ProgrammingLanguageToSyntacticConstruction(1, 1),
    ProgrammingLanguageToSyntacticConstruction(1, 4),
    ProgrammingLanguageToSyntacticConstruction(2, 2),
    ProgrammingLanguageToSyntacticConstruction(3, 1),
    ProgrammingLanguageToSyntacticConstruction(3, 4),
    ProgrammingLanguageToSyntacticConstruction(1, 3),
    ProgrammingLanguageToSyntacticConstruction(3, 3),
    ProgrammingLanguageToSyntacticConstruction(4, 5),
]


def res1(languages, constructions):
    res = dict()
    for l in languages:
        if "C" in l.name:
            tmp = []
            for c in constructions:
                if c.PL_id == l.id:
                    tmp.append(c.name)
            res[l.name] = tmp
    return res


def res2(languages, constructions):
    res = dict()
    for l in languages:
        count = 0
        sum = 0
        for c in constructions:
            if c.PL_id == l.id:
                count += 1
                sum += len(c.description)
        res[l.name] = round(sum / count, 2)
        sorted(res.items())
    return res


def res3(languages, constructions, languages_constructions):
    res = []
    for c in constructions:
        # if c.name[0] == "i": условие убрал, потому что не было
повторяющихся первых букв в словах
        for lc in languages_constructions:
            if lc.SC_id == c.id:
                res.append((languages[lc.PL_id - 1].name, c.name))
    return res


class ClassTest(unittest.TestCase):
    def test_first(self):
        test_list_1 = [
            ProgrammingLanguage(1, 'C++'),
            ProgrammingLanguage(2, 'Python'),
            ProgrammingLanguage(3, 'C#'),
            ProgrammingLanguage(4, 'Pascal'),
        ]

        test_list_2 = [
            SyntacticConstruction(1, 'if', 'Выбор выполнения', 1),
            SyntacticConstruction(2, 'match/case', 'Выбор выполнения с
несколькими вариантами', 2),
            SyntacticConstruction(3, 'switch/case', 'Выбор выполнения с
несколькими вариантами', 3),
            SyntacticConstruction(4, 'for', 'Повторение операций', 1),
            SyntacticConstruction(5, 'begin...end', 'Повторение операций',
4),
        ]
        result = res1(test_list_1, test_list_2)
        expected = {'C++': ['if', 'for'], 'C#': ['switch/case']}
        print(result)
        self.assertEqual(result, expected)

    def test_second(self):
        test_list_1 = [
            ProgrammingLanguage(1, 'C++'),
            ProgrammingLanguage(2, 'Python'),
            ProgrammingLanguage(3, 'C#'),
            ProgrammingLanguage(4, 'Pascal'),
        ]

        test_list_2 = [
            SyntacticConstruction(1, 'if', 'Выбор выполнения', 1),
            SyntacticConstruction(2, 'match/case', 'Выбор выполнения с
несколькими вариантами', 2),
            SyntacticConstruction(3, 'switch/case', 'Выбор выполнения с
несколькими вариантами', 3),
            SyntacticConstruction(4, 'for', 'Повторение операций', 1),
            SyntacticConstruction(5, 'begin...end', 'Повторение операций',
4),
        ]

        result = res2(test_list_1, test_list_2)
        expected = {'C++': 17.5, 'Python': 41.0, 'C#': 41.0, 'Pascal': 19.0}
        print(result)
        self.assertEqual(result, expected)

    def test_third(self):
        test_list_1 = [
            ProgrammingLanguage(1, 'C++'),
            ProgrammingLanguage(2, 'Python'),
            ProgrammingLanguage(3, 'C#'),
            ProgrammingLanguage(4, 'Pascal'),
        ]

        test_list_2 = [
            SyntacticConstruction(1, 'if', 'Выбор выполнения', 1),
            SyntacticConstruction(2, 'match/case', 'Выбор выполнения с
несколькими вариантами', 2),
            SyntacticConstruction(3, 'switch/case', 'Выбор выполнения с
несколькими вариантами', 3),
            SyntacticConstruction(4, 'for', 'Повторение операций', 1),
            SyntacticConstruction(5, 'begin...end', 'Повторение операций',
4),
        ]
        test_list_3 = [
            ProgrammingLanguageToSyntacticConstruction(1, 1),
            ProgrammingLanguageToSyntacticConstruction(1, 4),
            ProgrammingLanguageToSyntacticConstruction(2, 2),
            ProgrammingLanguageToSyntacticConstruction(3, 1),
            ProgrammingLanguageToSyntacticConstruction(3, 4),
            ProgrammingLanguageToSyntacticConstruction(1, 3),
            ProgrammingLanguageToSyntacticConstruction(3, 3),
            ProgrammingLanguageToSyntacticConstruction(4, 5),
        ]
        result = res3(test_list_1,test_list_2,test_list_3)
        expected = [('C++', 'if'), ('C#', 'if'), ('Python', 'match/case'),
('C++', 'switch/case'), ('C#', 'switch/case'), ('C++', 'for'), ('C#', 'for'),
('Pascal', 'begin...end')]
        print(result)
        self.assertEqual(result, expected)


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

Результат выполнения программы:



```
C:\Users\Alex4\AppData\Local\Programs\Python\Python39\python.exe C:\Users\Alex4\PycharmProjects\PCPL\main.py
{'C++': ['if', 'for'], 'C#': ['switch/case']}
{'C++': 17.5, 'Python': 41.0, 'C#': 41.0, 'Pascal': 19.0}
[('C++', 'if'), ('C#', 'if'), ('Python', 'match/case'), ('C++', 'switch/case'), ('C#', 'switch/case'), ('C++', 'for'), ('C#', 'for'), ('Pascal', 'begin...end')]
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```