Отчёт по Рубежному контролю №2

Вариант предметной области: №23
Вариант запросов: D

Текст программы

Ниже представлены фотографии исходного кода программы и тестов.

rk1_refactored.py

rk1_refactored.py > query_d1

```python
from operator import itemgetter


class SyntaxConstruction:
    def __init__(self, id, name, complexity, lang_id):
        self.id = id
        self.name = name
        self.complexity = complexity
        self.lang_id = lang_id


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


class LangSyntax:
    def __init__(self, lang_id, constr_id):
        self.lang_id = lang_id
        self.constr_id = constr_id


def get_test_data():
    langs = [
        ProgrammingLanguage(1, "Ada"),
        ProgrammingLanguage(2, "Assembly"),
        ProgrammingLanguage(3, "Python"),
        ProgrammingLanguage(4, "Java"),
        ProgrammingLanguage(5, "C++"),
    ]

    constructions = [
        SyntaxConstruction(1, "condition", 2, 3),
        SyntaxConstruction(2, "iteration", 3, 3),
        SyntaxConstruction(3, "recursion", 5, 4),
        SyntaxConstruction(4, "lambda", 4, 3),
        SyntaxConstruction(5, "class definition", 3, 4),
        SyntaxConstruction(6, "function definition", 2, 1),
    ]

```

rk1_refactored.py

rk1_refactored.py > query_d1

```python
def get_test_data():
    lang_syntax = [
        LangSyntax(1, 1),
        LangSyntax(3, 1),
        LangSyntax(4, 1),
        LangSyntax(3, 2),
        LangSyntax(4, 2),
        LangSyntax(3, 3),
        LangSyntax(4, 3),
        LangSyntax(3, 4),
        LangSyntax(4, 5),
        LangSyntax(1, 6),
    ]

    return langs, constructions, lang_syntax


def build_relations(langs, constructions, lang_syntax):
    one_to_many = [
        (c.name, c.complexity, l.name)
        for l in langs
        for c in constructions
        if c.lang_id == l.id
    ]
    many_to_many_temp = [
        (l.name, ls.lang_id, ls.constr_id)
        for l in langs
        for ls in lang_syntax
        if l.id == ls.lang_id
    ]
    many_to_many = [
        (c.name, c.complexity, lang_name)
        for lang_name, lang_id, constr_id in many_to_many_temp
        for c in constructions
        if c.id == constr_id
    ]

    return one_to_many, many_to_many


def query_d1(one_to_many):
    result = [
        (name, lang_name)
        for name, complexity, lang_name in one_to_many
        if name.endswith("ion") and " " not in name
    ]
```

rk1_refactored.py

rk1_refactored.py > query_d1

```python
def query_d1(one_to_many):
    result = [
        (name, lang_name)
        for name, complexity, lang_name in one_to_many
        if name.endswith("ion") and " " not in name
    ]
    return result


def query_d2(langs, one_to_many):
    res_unsorted = []

    for l in langs:
        l_constrs = [
            (name, complexity, lang_name)
            for name, complexity, lang_name in one_to_many
            if lang_name == l.name
        ]

        if l_constrs:
            complexities = [complexity for _, complexity, _ in l_constrs]
            avg_complexity = sum(complexities) / len(complexities)
            res_unsorted.append((l.name, avg_complexity))

    res_sorted = sorted(res_unsorted, key=itemgetter(1))
    return res_sorted


def query_d3(langs, many_to_many):
    res = {}

    for l in langs:
        if l.name.startswith("A"):
            l_constrs = [
                (name, complexity, lang_name)
                for name, complexity, lang_name in many_to_many
                if lang_name == l.name
            ]
            constr_names = [name for name, _, _ in l_constrs]
            res[l.name] = constr_names

    return res


```

rk1_refactored.py

rk1_refactored.py > query_d1

```python
def main():
    langs, constructions, lang_syntax = get_test_data()
    one_to_many, many_to_many = build_relations(langs, constructions, lang_syntax)

    print("Задание D1")
    print(query_d1(one_to_many))

    print("\nЗадание D2")
    print(query_d2(langs, one_to_many))

    print("\nЗадание D3")
    print(query_d3(langs, many_to_many))


if __name__ == "__main__":
    main()

```

Тесты

test_rk1.py

test_rk1.py > TestRk1Queries > setUpClass

```python
import unittest

from rk1_refactored import (
    get_test_data,
    build_relations,
    query_d1,
    query_d2,
    query_d3,
)


class TestRk1Queries(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        langs, constructions, lang_syntax = get_test_data()
        one_to_many, many_to_many = build_relations(
            langs, constructions, lang_syntax
        )
        cls.langs = langs
        cls.one_to_many = one_to_many
        cls.many_to_many = many_to_many

    def test_query_d1(self):
        result = query_d1(self.one_to_many)

        expected = [
            ("condition", "Python"),
            ("iteration", "Python"),
            ("recursion", "Java"),
        ]

        self.assertEqual(result, expected)

    def test_query_d2(self):
        result = query_d2(self.langs, self.one_to_many)

        expected = [
            ("Ada", 2.0),
            ("Python", 3.0),
            ("Java", 4.0),
        ]

        self.assertEqual(result, expected)

```

test_rk1.py

test_rk1.py > TestRk1Queries > setUpClass

```
12  class TestRk1Queries(unittest.TestCase):
45      def test_query_d3(self):
46          result = query_d3(self.langs, self.many_to_many)
47
48          self.assertIn("Ada", result)
49          self.assertIn("Assembly", result)
50
51          self.assertEqual(
52              result["Ada"],
53              ["condition", "function definition"]
54          )
55          self.assertEqual(
56              result["Assembly"],
57              []
58          )
59
60
61  if __name__ == "__main__":
62      unittest.main()
63
```

Результаты работы программы

Вывод, полученный при запуске программы, приведён на фото ниже.

```
sejrandovletov@MacBook-Pro-Sejran rk % python3 test_rk1.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```