ПРОГ-5

Лабораторная работа 2. Ряд Фибоначчи с помощью итераторов

Задание 1

gen_fib.py × fibonacci_iterator.py test_fib.py test_fibonacci_iterator.py

```python
import functools


def fib_elem_gen():  3 usages
    """Генератор, возвращающий элементы ряда Фибоначчи"""
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b


def fib_coroutine(g):  1 usage
    @functools.wraps(g)
    def inner(*args, **kwargs):
        gen = g(*args, **kwargs)
        gen.send(None)
        return gen

    return inner


@fib_coroutine  8 usages
def my_genn():
    """Сопрограмма для генерации ряда Фибоначчи"""
    result = []
    fib_gen = fib_elem_gen()

    while True:
        n = yield result
        if n is not None:
            result = [next(fib_gen) for _ in range(n)]
            fib_gen = fib_elem_gen()  # Сбрасываем для следующего вызова


# Демонстрация работы
if __name__ == "__main__":
    print("Демонстрация генератора Фибоначчи:")
    g = fib_elem_gen()
    for i in range(10):
        el = next(g)
        print(el, end=" ")
        if el > 20:
            break
    print("\n")

    print("Демонстрация сопрограммы:")
    gen = my_genn()
    print("gen.send(3):", gen.send(3))
    print("gen.send(5):", gen.send(5))
    print("gen.send(8):", gen.send(8))
```

gen_fib.py | fibonacci_iterator.py | test_fib.py | test_fibonacci_iterator.py

```python
from gen_fib import my_genn

def test_fib_1():  1 usage
    """Тривиальный случай n = 3, список [0, 1, 1]"""
    gen = my_genn()
    result = gen.send(3)
    assert result == [0, 1, 1]

def test_fib_2():  1 usage
    """Пять первых членов ряда"""
    gen = my_genn()
    result = gen.send(5)
    assert result == [0, 1, 1, 2, 3]

def test_fib_3():  1 usage
    """Восемь первых членов ряда"""
    gen = my_genn()
    result = gen.send(8)
    assert result == [0, 1, 1, 2, 3, 5, 8, 13]

def test_fib_4():  1 usage
    """Крайний случай: n = 1"""
    gen = my_genn()
    result = gen.send(1)
    assert result == [0]

def test_fib_5():  1 usage
    """Крайний случай: n = 0 (пустой список)"""
    gen = my_genn()
    result = gen.send(0)
    assert result == []

def test_fib_6():  1 usage
    """Крайний случай: n = 2"""
    gen = my_genn()
    result = gen.send(2)
    assert result == [0, 1]

# Запуск тестов
if __name__ == "__main__":
    test_fib_1()
    test_fib_2()
    test_fib_3()
    test_fib_4()
    test_fib_5()
    test_fib_6()
    print("Все тесты пройдены успешно!")
```

```
Terminal  Local

tarass1ky@MacBook-Air-Tarasova LR-1.1 % python3 gen_fib.py
Демонстрация генератора Фибоначчи:
0 1 1 2 3 5 8 13 21

Демонстрация сопрограммы:
gen.send(3): [0, 1, 1]
gen.send(5): [0, 1, 1, 2, 3]
gen.send(8): [0, 1, 1, 2, 3, 5, 8, 13]

tarass1ky@MacBook-Air-Tarasova LR-1.1 % python3 test_fib.py
Все тесты пройдены успешно!

tarass1ky@MacBook-Air-Tarasova LR-1.1 %
```

Задание 2

gen_fib.py | fibonacci_iterator.py | test_fib.py | test_fibonacci_iterator.py

```python
class FibonacchiLst:  8 usages

    def __init__(self, instance):
        self.instance = instance
        self.idx = 0  # инициализируем индекс для перебора элементов
        self.fib_set = self._generate_fibonacci_set(max(instance) if instance else 0)

    def __iter__(self):
        return self  # возвращает экземпляр класса, реализующего протокол итераторов

    def __next__(self):  # возвращает следующий по порядку элемент итератора
        while True:
            try:
                res = self.instance[self.idx]  # получаем очередной элемент из iterable

            except IndexError:
                raise StopIteration

            if res in self.fib_set:  # проверяем, является ли число числом Фибоначчи
                self.idx += 1  # если да, возвращаем значение и увеличиваем индекс
                return res

            self.idx += 1  # если нет, то просто увеличиваем индекс

    def _generate_fibonacci_set(self, max_value):  1 usage
        """Генерирует множество чисел Фибоначчи до максимального значения в списке"""
        if max_value == 0:
            return {0}

        fib_set = {0, 1}
        a, b = 0, 1

        while True:
            next_fib = a + b
            if next_fib > max_value:
                break
```

```
                fib_set.add(next_fib)
                a, b = b, next_fib

            return fib_set


    # Демонстрация работы
    if __name__ == "__main__":
        lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
        fib_iterator = FibonacchiLst(lst)
        result = list(fib_iterator)
        print(f"Для списка {lst} FibonacchiLst возвращает: {result}")
```

gen_fib.py | fibonacci_iterator.py | test_fib.py | test_fibonacci_iterator.py

```
from fibonacci_iterator import FibonacchiLst

def test_fibonacci_iterator_1():  1 usage
    """Базовый тест из задания"""
    lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
    result = list(FibonacchiLst(lst))
    expected = [0, 1, 2, 3, 5, 8, 1]
    assert result == expected

def test_fibonacci_iterator_2():  1 usage
    """Пустой список"""
    lst = []
    result = list(FibonacchiLst(lst))
    expected = []
    assert result == expected

def test_fibonacci_iterator_3():  1 usage
    """Список без чисел Фибоначчи"""
    lst = [4, 6, 7, 9, 10, 11]
    result = list(FibonacchiLst(lst))
    expected = []
    assert result == expected

def test_fibonacci_iterator_4():  1 usage
    """Список только с числами Фибоначчи"""
    lst = [0, 1, 2, 3, 5, 8, 13]
    result = list(FibonacchiLst(lst))
    expected = [0, 1, 2, 3, 5, 8, 13]
    assert result == expected

def test_fibonacci_iterator_5():  1 usage
    """Список с большими числами"""
    lst = [21, 34, 55, 89, 100]
    result = list(FibonacchiLst(lst))
    expected = [21, 34, 55, 89]
    assert result == expected
```

```python

def test_fibonacci_iterator_6():  1 usage
    """Список с повторяющимися числами Фибоначчи"""
    lst = [0, 0, 1, 1, 2, 2, 3, 3]
    result = list(FibonacchiLst(lst))
    expected = [0, 0, 1, 1, 2, 2, 3, 3]
    assert result == expected

if __name__ == "__main__":
    test_fibonacci_iterator_1()
    test_fibonacci_iterator_2()
    test_fibonacci_iterator_3()
    test_fibonacci_iterator_4()
    test_fibonacci_iterator_5()
    test_fibonacci_iterator_6()
    print("Все тесты FibonacchiLst пройдены успешно!")
```

```
Terminal  Local ×  +  ⌄

tarass1ky@MacBook-Air-Tarasova LR-1.1 % python3 fibonacci_iterator.py
Для списка [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1] FibonacchiLst возвращает: [0, 1, 2, 3, 5, 8, 1]

tarass1ky@MacBook-Air-Tarasova LR-1.1 % python3 test_fibonacci_iterator.py
Все тесты FibonacchiLst пройдены успешно!

tarass1ky@MacBook-Air-Tarasova LR-1.1 %
```