

Лабораторная работа 2. Ряд Фибоначчи с помощью итераторов

Задание 1

gen_fib.py

```
1  import functools
2  import itertools
3
4
5  def fib_elem_gen(): 3 usages
6      """Генератор, возвращающий элементы ряда Фибоначчи"""
7      a = 0
8      b = 1
9
10     while True:
11         yield a
12         res = a + b
13         a = b
14         b = res
15
16
17     def fib_coroutine(g): 1 usage
18         @functools.wraps(g)
19         def inner(*args, **kwargs):
20             gen = g(*args, **kwargs)
21             gen.send(None)
22             return gen
23
24         return inner
25
26
27     @fib_coroutine 11 usages
28     def my_genn():
29         """Сопрограмма для генерации ряда Фибоначчи"""
30         fib_gen = fib_elem_gen()
31
32         while True:
33             number_of_fib_elem = yield
34             if number_of_fib_elem <= 0:
35                 result = []
36             else:
37                 # Генерируем n элементов с помощью itertools.islice
38                 result = list(itertools.islice(fib_gen, number_of_fib_elem))
39                 # Сбрасываем генератор для следующего вызова
40                 fib_gen = fib_elem_gen()
41
42             yield result
43
44
45     if __name__ == "__main__":
46         g = fib_elem_gen()
47
48         print("Первые 10 элементов ряда Фибоначчи:")
49         for i in range(10):
50             el = next(g)
51             print(el, end=" ")
52         print("\n")
53
54         gen = my_genn()
55         print("gen.send(3):", gen.send(3))
56
```

Результат

```
Первые 10 элементов ряда Фибоначчи:  
0 1 1 2 3 5 8 13 21 34  
  
gen.send(3): [0, 1, 1]  
  
Process finished with exit code 0
```

test_fib.py

```
1 import pytest  
2 from gen_fib import my_genn  
3  
4  
5 def test_fib_1():  
6     """Тривиальный случай n = 3, список [0, 1, 1]"""  
7     gen = my_genn()  
8     assert gen.send(3) == [0, 1, 1]  
9  
10  
11 def test_fib_2():  
12     """Пять первых членов ряда"""  
13     gen = my_genn()  
14     assert gen.send(5) == [0, 1, 1, 2, 3]  
15  
16 def test_fib_zero():  
17     """Крайний случай: запрос 0 элементов"""  
18     gen = my_genn()  
19     assert gen.send(0) == []  
20  
21  
22 def test_fib_one():  
23     """Крайний случай: запрос 1 элемента"""  
24     gen = my_genn()  
25     assert gen.send(1) == [0]  
26  
27
```

```

28 > def test_fib_two():
29     """Крайний случай: запрос 2 элементов"""
30     gen = my_genn()
31     assert gen.send(2) == [0, 1]
32
33
34 > def test_fib_negative():
35     """Крайний случай: отрицательное количество элементов"""
36     gen = my_genn()
37     assert gen.send(-1) == []
38
39
40 > def test_fib_sequence():
41     """Последовательные вызовы с разными параметрами"""
42     gen = my_genn()
43     assert gen.send(3) == [0, 1, 1]
44     assert gen.send(5) == [0, 1, 1, 2, 3]
45     assert gen.send(2) == [0, 1]
46
47
48 > def test_fib_large():
49     """Запрос большего количества элементов"""
50     gen = my_genn()
51     result = gen.send(10)
52     expected = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
53     assert result == expected
54
55
56 > def test_fib_empty_after_zero():
57     """Проверка поведения после запроса 0 элементов"""
58     gen = my_genn()
59     assert gen.send(0) == []
60     assert gen.send(3) == [0, 1, 1]
61
62
63 > if __name__ == "__main__":
64     pytest.main([__file__, "-v"])

```

Результат

```

Testing started at 8:53 ...
Launching pytest with arguments test_fib.py::test_fib_large --no-header --no-summary -q in D:\PycharmProjects\Prog5\LR2

===== test session starts =====
collecting ... collected 1 item

test_fib.py::test_fib_large PASSED [100%]

===== 1 passed in 0.08s =====

Process finished with exit code 0

```

Задание 2

gen_fib.py

```
1  import functools
2  import itertools
3  import math
4
5
6  def fib_elem_gen(): 3 usages
7      """Генератор, возвращающий элементы ряда Фибоначчи"""
8      a = 0
9      b = 1
10     while True:
11         yield a
12         res = a + b
13         a = b
14         b = res
15
16  def fib_coroutine(g): 1 usage
17     @functools.wraps(g)
18     def inner(*args, **kwargs):
19         gen = g(*args, **kwargs)
20         gen.send(None)
21         return gen
22     return inner
23
24  @fib_coroutine 2 usages
25  def my_genn():
26     """Сопрограмма для генерации ряда Фибоначчи"""
27     fib_gen = fib_elem_gen()
28     while True:
29         number_of_fib_elem = yield
30         if number_of_fib_elem <= 0:
31             result = []
32         else:
33             # Генерируем n элементов с помощью itertools.islice
34             result = list(itertools.islice(fib_gen, number_of_fib_elem))
35             # Сбрасываем генератор для следующего вызова
36             fib_gen = fib_elem_gen()
37         yield result
38
39  class FibonacchiLst: 7 usages
40     """Итератор, возвращающий элементы из списка, которые принадлежат ряду Фибоначчи"""
41     def __init__(self, lst):
42         self.lst = lst
43         self.idx = 0 # инициализируем индекс для перебора элементов
44
45     def __iter__(self):
46         return self # возвращает экземпляр класса, реализующего протокол итераторов
47
48     def __next__(self):
49         """Возвращает следующий элемент из списка, который является числом Фибоначчи"""
50         while True:
51             if self.idx >= len(self.lst):
52                 raise StopIteration
53             if self.lst[self.idx] in fib_elem_gen():
54                 return self.lst[self.idx]
55             self.idx += 1
```

```

54         current_element = self.lst[self.idx]
55         self.idx += 1
56
57
58
59 # Демонстрация работы
60 ► if __name__ == "__main__":
61     g = fib_elem_gen()
62
63     print("Первые 10 элементов ряда Фибоначчи:")
64     for i in range(10):
65         el = next(g)
66         print(el, end=" ")
67     print("\n")
68
69 # Тестирование сопрограмы
70 gen = my_genn()
71 print("gen.send(3):", gen.send(3))
72
73
74 # Тестирование FibonacchiLst
75 print("\nТестирование FibonacchiLst:")
76 test_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
77 fib_iterator = FibonacchiLst(test_list)
78 result = list(fib_iterator)
79 print(f"Для списка {test_list}")
80 print(f"Числа Фибоначчи: {result}")
81 print(f"Ожидаемый результат: [0, 1, 2, 3, 5, 8, 1]")

```

Результат

```

Первые 10 элементов ряда Фибоначчи:
0 1 1 2 3 5 8 13 21 34

gen.send(3): [0, 1, 1]

Тестирование FibonacchiLst:
Для списка [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
Числа Фибоначчи: []
Ожидаемый результат: [0, 1, 2, 3, 5, 8, 1]

Process finished with exit code 0

```

test_fib.py

```

1 import pytest
2 from gen_fib import my_genn, FibonacchiLst
3
4 # Тесты для FibonacchiLst
5 def test_fibonacci_lst_basic():
6     """Базовый тест для FibonacchiLst"""
7     test_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
8     result = list(FibonacchiLst(test_list))
9     expected = [0, 1, 2, 3, 5, 8, 1]
10    assert result == expected
11
12 def test_fibonacci_lst_empty():
13     """Тест с пустым списком"""
14     result = list(FibonacchiLst([]))
15     assert result == []
16
17 def test_fibonacci_lst_single():
18     """Тест с одним элементом"""
19     result = list(FibonacchiLst([8]))
20     assert result == [8]
21
22 def test_fibonacci_lst_negative():
23     """Тест с отрицательными числами"""
24     result = list(FibonacchiLst([-1, 0, 1, -2]))
25     assert result == [0, 1]
26
27 def test_fibonacci_lst_large_numbers():
28     """Тест с большими числами Фибоначчи"""
29     test_list = [55, 89, 144, 200]
30     result = list(FibonacchiLst(test_list))
31     assert result == [55, 89, 144]
32
33 if __name__ == "__main__":
34     pytest.main([__file__, "-v"])

```

Результат

```

===== test session starts =====
collecting ... collected 1 item

test_fib.py::test_fibonacci_lst_large_numbers PASSED [100%]

===== 1 passed in 0.05s =====

Process finished with exit code 0

```