МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет по лабораторной работе №12
«Рекурсия в языке Python»

по дисциплине «Основы программной инженерии»

Выполнила:
Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

```
def rec (n):
    if n > 0:
        rec(n - 1)

    print()
```

Рисунок 1 – Пример использования функции

```
n = 0
for i in range(1, n+1):
    n += i
```

Рисунок 2 – Пример решения задачи с помощью цикла for

```
def recursion(n):
    if n == 1:
        return 1

    return n + recursion(n - 1)
```

Рисунок 3 – Пример использования рекурсивной функции

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

Рисунок 4 – Пример нахождения факториала

```python
def factorial(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

Рисунок 5 – Пример нахождения факториала

```python
def fib(n):
    if n == 0 or n == 1:
        return 0
    else:
        return fib(n - 2) + fib(n - 1)
```

Рисунок 6 – Пример нахождения последовательности Фибоначчи

```python
def factorial(n):
    product = 1
    while n > 1:
        product *= n
        n -= 1
    return product


def fib(n):
    a, b = 0, 1
    while n > 0:
        a, b = b, a + b
        n -= 1
    return a
```

Рисунок 7 – Пример итеративного кода

```python
from functools import lru_cache

@lru_cache
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)
```

Рисунок 8 – Пример использования декоратора

```python
def fib(n):
    if n <= 1:
        return n, 0
    else:
        (a, b) = fib(n - 1)
        return a + b, a
```

Рисунок 9 – Пример использования линейной рекурсии

```python
def cursing(depth):
    try:
        cursing(depth + 1)
    except RuntimeError as RE:
        print('I recursed {} times!'.format(depth))


if __name__ == '__main__':
    cursing(0)
```

Рисунок 10 – Пример программы с превышением максимальной глубины рекурсии

```
def countdown(n):
    if n == 0:
        print("Blastoff")
    else:
        print(n)
        countdown(n - 1)
```

Рисунок 11 – Пример использования хвостовой рекурсии

```
def find_max(seq, max_so_far):
    if not seq:
        return max_so_far
    if max_so_far < seq[0]:
        return find_max(seq[1:], seq[0])
    else:
        return find_max(seq[1:], max_so_far)
```

Рисунок 12 – Пример использования поиска максимального значения в последовательном контейнере, написанная с использованием хвостовой рекурсии

```
                raise TailRecurseException(args, kwargs)
            else:
                while True:
                    try:
                        return g(*args, **kwargs)
                    except TailRecurseException as e:
                        args = e.args
                        kwargs = e.kwargs

        func.__doc__ = g.__doc__
        return func

    @tail_call_optimized
    def fib(i, current = 0, next = 1):

main ×
C:\Users\podar\study\anaconda\envs\12LR\python.exe "C:/Use
3364476487643178326662161200510754331030214846068006390656
```

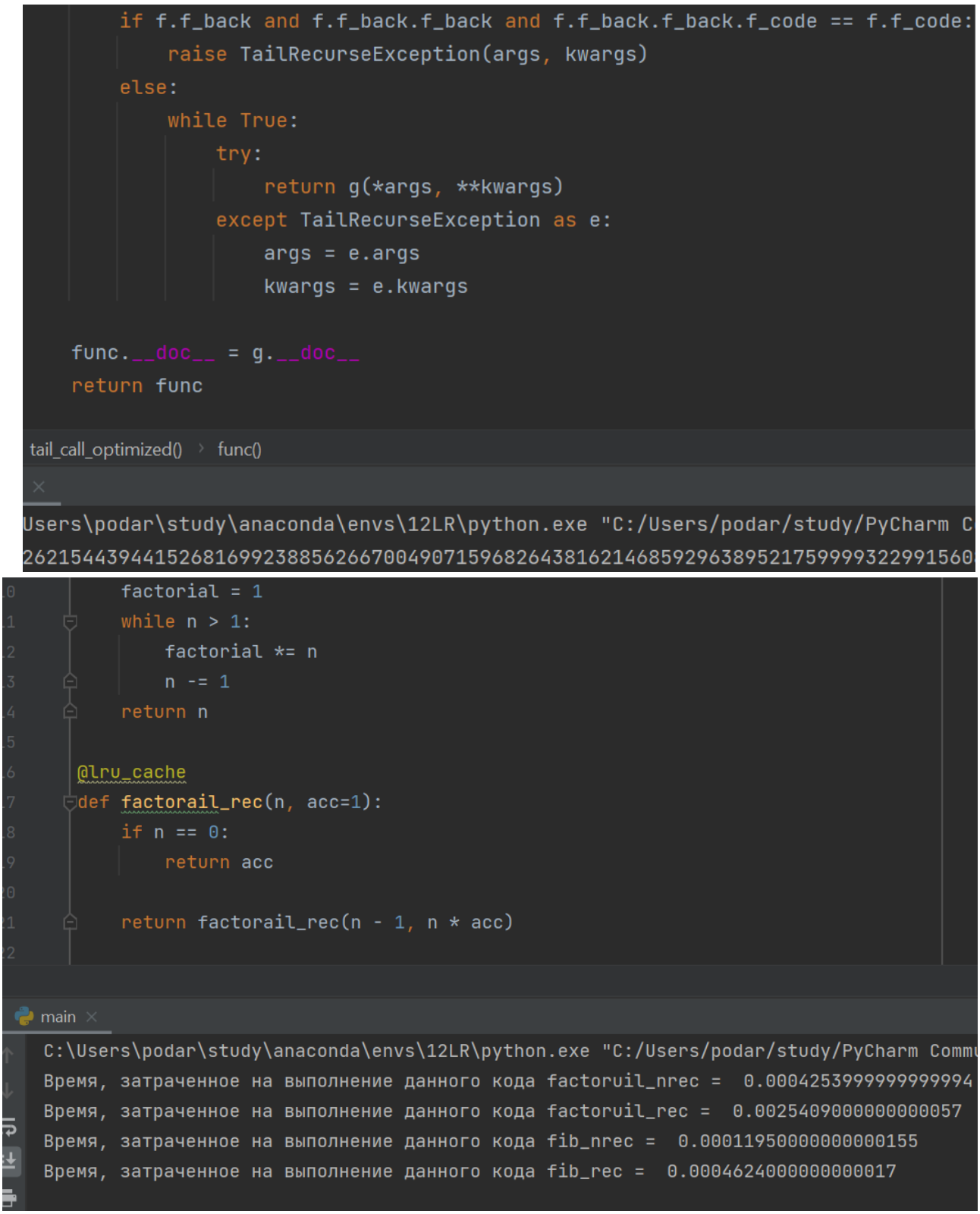

Рисунок 13 – Пример использования декоратора @tail_call_optimized

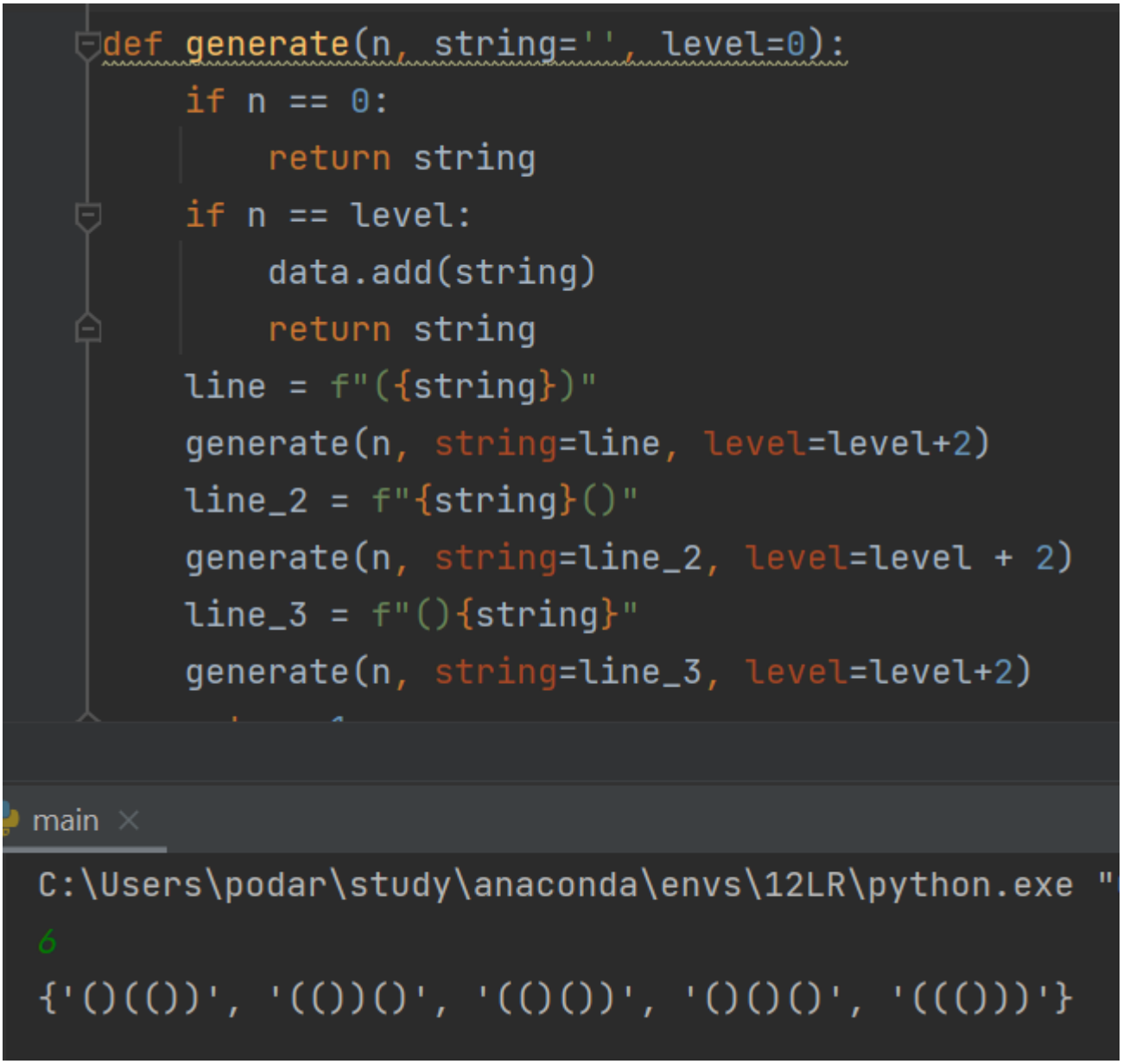

Рисунок 15 – Пример решения индивидуального задания

Вопросы

1. **Для чего нужна рекурсия?**
   Рекурсия подразумевает более компактный вид записи выражения. Обычно это зависимость процедур (функций, членов прогресс и т.д.) соседних порядковых номеров. Некоторые зависимости очень сложно выразить какой-либо формулой, кроме как рекурсивной. Рекурсия незаменима в ряде случаев при программировании замкнутых циклов.

2. **Что называется базой рекурсии?**
   База рекурсии – это такие аргументы функции, которые делают задачу настолько простой, что решение не требует дальнейших вложенных вызовов.

3. **Самостоятельно изучите что является стеком программы.**

Как используется стек программы при вызове функции?
Стек вызовов – в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм в программу и для возврата в программу из обработчика прерывания.
При вызове подпрограммы или возникновении прерываний, в стек заносится адрес возврата – адрес в памяти следующей инструкции приостановленной программы и управление передаётся подпрограмме или подпрограмме обработчику.

4. **Как получить текущее значение максимальной глубины рекурсии в языке?**
   Чтобы проверить текущие параметры лимита нужно запустить:
   sys.getrecursionlimit()

5. **Что произойдёт если число рекурсивных вызовов превысит максимальную глубину рекурсии?**
   Программа выдаст ошибку: RuntimeError: Maximum Recursion Depth Exceeded

6. **Как изменить максимальную глубину рекурсии?**
   Изменить максимальную глубины рекурсии можно с помощью sys.setrecursionlimit(limit). Чтобы проверить параметры лимита, нужно запустить sys.getrecursionlimit().

7. **Каково назначение декоратор lru_cache?**
   Декоратор можно использовать для уменьшения количества лишних вычислений.

8. **Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?**
   Хвостовая рекурсия – частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Подобный вид рекурсии примечателен тем, что может быть легко заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции.
   Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах. В некоторых функциональных языках программирования спецификация гарантирует обязательную оптимизацию хвостовой рекурсии.