

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №8**  
**дисциплины «Программирование на Python»**

Выполнил:

Лейс Алексей Вячеславович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем»

---

(подпись)

Руководитель практики: кандидат тех.  
наук доцент кафедры  
инфокоммуникаций: Воронкин Р.А

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023г.

## Тема: Рекурсия в языке Python

**Цель работы:** приобретение навыков по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.

### Порядок выполнения работы:

Пример рекурсивной функции из лабораторной работы

```
1  @tail call optimized
2  def fib(i, current = 0, next = 1):
3      if i == 0:
4          return current
5      else:
6          return fib(i - 1, next, current + next)
7      if __name__ == '__main__':
8          print(fib(10000))
9
```

### Индивидуальное задание:

5. Создайте рекурсивную функцию, печатающую все подмножества множества  $\{1, 2, \dots, N\}$ .

```
4  def print_subsets(n, current_set=None, index=1):
5      """
6      Рекурсивная функция для печати всех подмножеств множества {1,2,...n}.
7      """
8      if current_set is None:
9          current_set = set()
10
11     if index > n:
12         print(current_set)
13         return
14
15     # Не включаем текущий элемент
16     print_subsets(n, current_set, index + 1)
17
18     # Включаем текущий элемент
19     current_set.add(index)
20     print_subsets(n, current_set, index + 1)
21     current_set.remove(index)
22
23 if __name__ == "__main__":
24     n = 3 # Замените значение n на необходимое
25     print_subsets(n)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Попробуйте новую кроссплатформенную оболочку PowerShell (<https://aka.ms/pscore6>)

```
PS I:\MBT-6-o-22-1\Программирование на Python\12\lab_2.9> & 'D:\LAV\Files\Programs\VisualSt
LAV\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/...
-1\Программирование на Python\12\lab_2.9\2.py'
set()
{3}
{2}
{3, 2}
{1}
{1, 3}
{1, 2}
{1, 2, 3}
```

### Ответы на вопросы:

### 1. Для чего нужна рекурсия?

- Рекурсия используется в программировании для решения задач, которые могут быть разбиты на более простые подзадачи. Функция в рекурсивном вызове вызывает сама себя для решения подзадачи, что делает код более читаемым и естественным.

### 2. Что называется базой рекурсии?

- База рекурсии - это условие, которое определяет конец рекурсивного процесса. Когда выполняется база рекурсии, функция прекращает вызывать саму себя.

### 3. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций?

- Стек программы - это структура данных, используемая для хранения локальных переменных и данных вызываемых функций. При вызове функции, ее локальные переменные и адрес возврата помещаются в стек. Когда функция завершается, данные извлекаются из стека.

### 4. Как получить текущее значение максимальной глубины рекурсии в языке Python?

- Текущую максимальную глубину рекурсии можно получить с помощью `sys.getrecursionlimit()` из модуля `sys`.

### 5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python?

- Превышение максимальной глубины рекурсии приведет к ошибке **RecursionError**. Это может произойти из-за неэффективной рекурсии, и решением может быть переписывание кода с использованием циклов или оптимизированных подходов.

### 6. Как изменить максимальную глубину рекурсии в языке Python?

- Максимальную глубину рекурсии можно изменить с помощью `sys.setrecursionlimit(new_limit)`. Однако, изменение этого значения может

повлиять на стабильность работы программы, поэтому следует использовать осторожно.

7. **Каково назначение декоратора `lru_cache`?**

- Декоратор **`lru_cache`** используется для кэширования результатов вызовов функции. Это увеличивает производительность, избегая повторных вычислений для одних и тех же аргументов.

8. **Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?**

- Хвостовая рекурсия - это форма рекурсии, при которой рекурсивный вызов является последней операцией в функции. Оптимизация хвостовых вызовов (tail call optimization, TCO) в Python официально не поддерживается, но в некоторых случаях интерпретатор может проводить оптимизации, основанные на конкретных условиях кода. Однако, в общем случае, Python не оптимизирует хвостовые вызовы так, как делают некоторые другие языки, такие как Scheme.

Вывод в ходе работы приобрел навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.