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

Институт цифрового развития Кафедра инфокоммуникаций

## «Рекурсия в языке Python»

## ОТЧЕТ по лабораторной работе №12 дисциплины «Основы программной инженерия»

Выполнил:
Зиёдуллаев Жавохир Эркин угли
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения
(подпись) Проверил:
(подпись)
Дата защиты

Ставрополь, 2022 г.

Пример 1

```
#!/usr/bin/env python3
 ⊕# -*- coding: utf-8 -*-
 def recursion(n):
          return 1
      return n + recursion(n - 1)
dif __name__ == '__main__':
      print(f"Recursion: {recursion(5)}")
      for i in range(1, k + 1):
 if __name__ == '__main__'
C:\Users\Vova\AppData\Local\Programs\Python\
Recursion: 15
'for': 15
```

## Индивидуальное задание

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

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

new_set = [1,2,3,4]
power_set = [[1]

ifor x in new_set:

new_set = power_set|
for i in range(len(new_set)):
    new_set[1].append(x)
    power_set, extend(new_set)

print(power_set)

print(power_set)

print(power_set)

cw_set

indivek ×

C:\Users\work\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:\Users\work\laba-12\PyCharm\indivek.py

[[1, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4], [1, 2, 2, 3, 3, 3, 4
```

GitHub: https://github.com/javoxir21/laba-12-.git

### Контрольные вопросы:

- 1. Рекурсия существенно сокращает объем кода и входит во многие встроенные функции языков.
- 2. База рекурсии это тривиальный случай, при котором решение задачи очевидно, то есть не требуется обращение функции к себе.
- 3. Компьютер использует стек вызовов специальную область памяти, где хранит данные о точках перехода между фрагментами кода. последовательность шагов, выполняемых при вызове функции: а. Программа сталкивается с вызовом функции. b. Создается фрейм стека, который помещается в стек. с. Процессор переходит к точке начала выполнения функции. d. Инструкции внутри функции начинают выполняться. После завершения функции, выполняются следующие шаги: е. Регистры восстанавливаются из стека вызовов. f. Фрейм стека вытягивается из стека. Освобождается память, которая была выделена для всех локальных переменных и аргументов. g. Обрабатывается возвращаемое значение. h. ЦП возобновляет выполнение кода (исходя из обратного адреса).
- 4. Чтобы получить текущее значение максимальной глубины рекурсии следует вызвать функцию sys.getrecursionlimit()
- 5. Когда предел достигнут, возникает исключение: RuntimeError: Maximum Recursion Depth Exceeded

- 6. Можно изменить предел глубины рекурсии с помощью вызова: sys.setrecursionlimit(limit)
- 7. lru\_cache можно использовать для уменьшения количества лишних вычислений.
- 8. Хвостовая рекурсия частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Оптимизация происходит, вызывая исключение, если оно является его прародителем, и перехватывает исключения, чтобы подделать оптимизацию хвоста.