

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе № 2.9

Работа со словарями в языке Python

Выполнил студент группы ИВТ-б-о-20-1

Ищенко М.А.

Работа защищена « » _____ 20__ г.

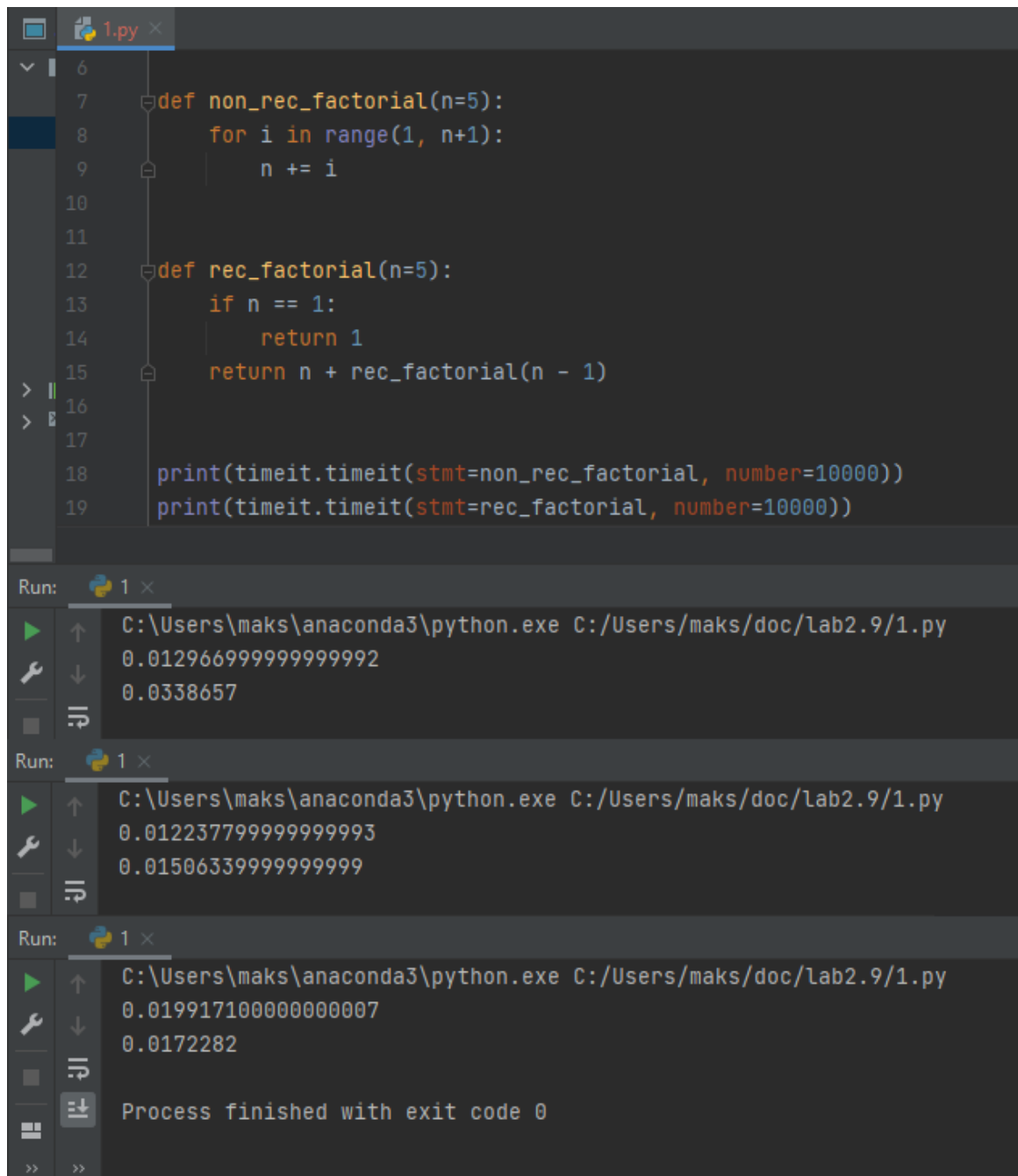
Проверил(а) _____

Ставрополь 2021

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

Создан общедоступный репозиторий на GitHub. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.

С помощью модуля timeit оценена скорость работы итеративной и рекурсивной версий функций, рис. 1



The screenshot displays the PyCharm IDE interface. The top pane shows a Python file named '1.py' with the following code:

```
6
7 def non_rec_factorial(n=5):
8     for i in range(1, n+1):
9         n += i
10
11
12 def rec_factorial(n=5):
13     if n == 1:
14         return 1
15     return n + rec_factorial(n - 1)
16
17
18 print(timeit.timeit(stmt=non_rec_factorial, number=10000))
19 print(timeit.timeit(stmt=rec_factorial, number=10000))
```

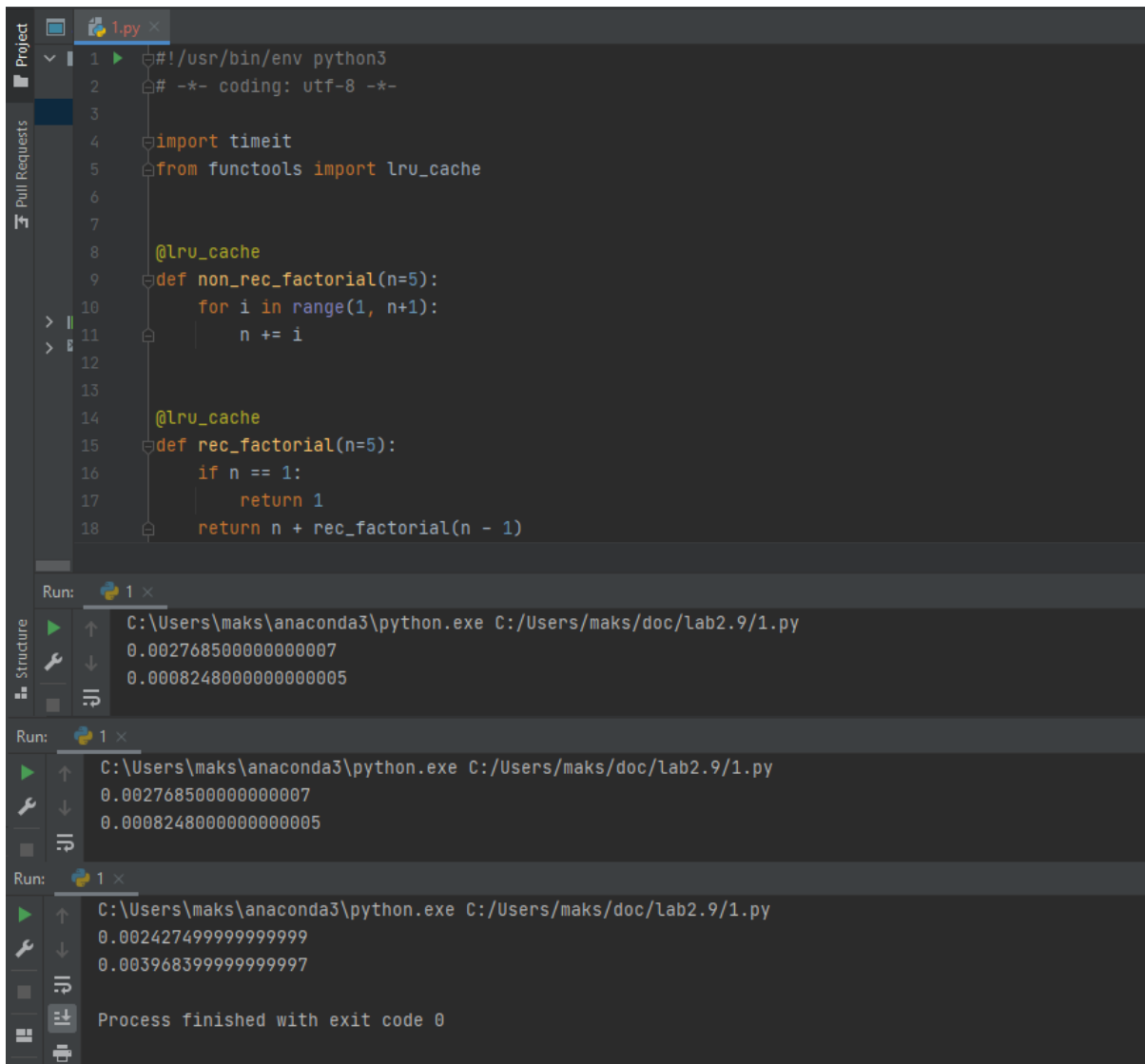
The bottom pane shows the execution results for three runs of the code. Each run shows the command being executed and the output values for the iterative and recursive functions.

Run	Command	non_rec_factorial	rec_factorial
1	C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py	0.012966999999999992	0.0338657
2	C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py	0.012237799999999993	0.015063399999999999
3	C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py	0.019917100000000007	0.0172282

The final status message indicates: "Process finished with exit code 0".

Рисунок 1 – Пример

То же самое с использованием декоратора lru_cache, рис. 2



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import timeit
5  from functools import lru_cache
6
7
8  @lru_cache
9  def non_rec_factorial(n=5):
10     for i in range(1, n+1):
11         n += i
12
13
14  @lru_cache
15  def rec_factorial(n=5):
16     if n == 1:
17         return 1
18     return n + rec_factorial(n - 1)
```

Run: 1 x

```
C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py
0.002768500000000007
0.0008248000000000005
```

Run: 1 x

```
C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py
0.002768500000000007
0.0008248000000000005
```

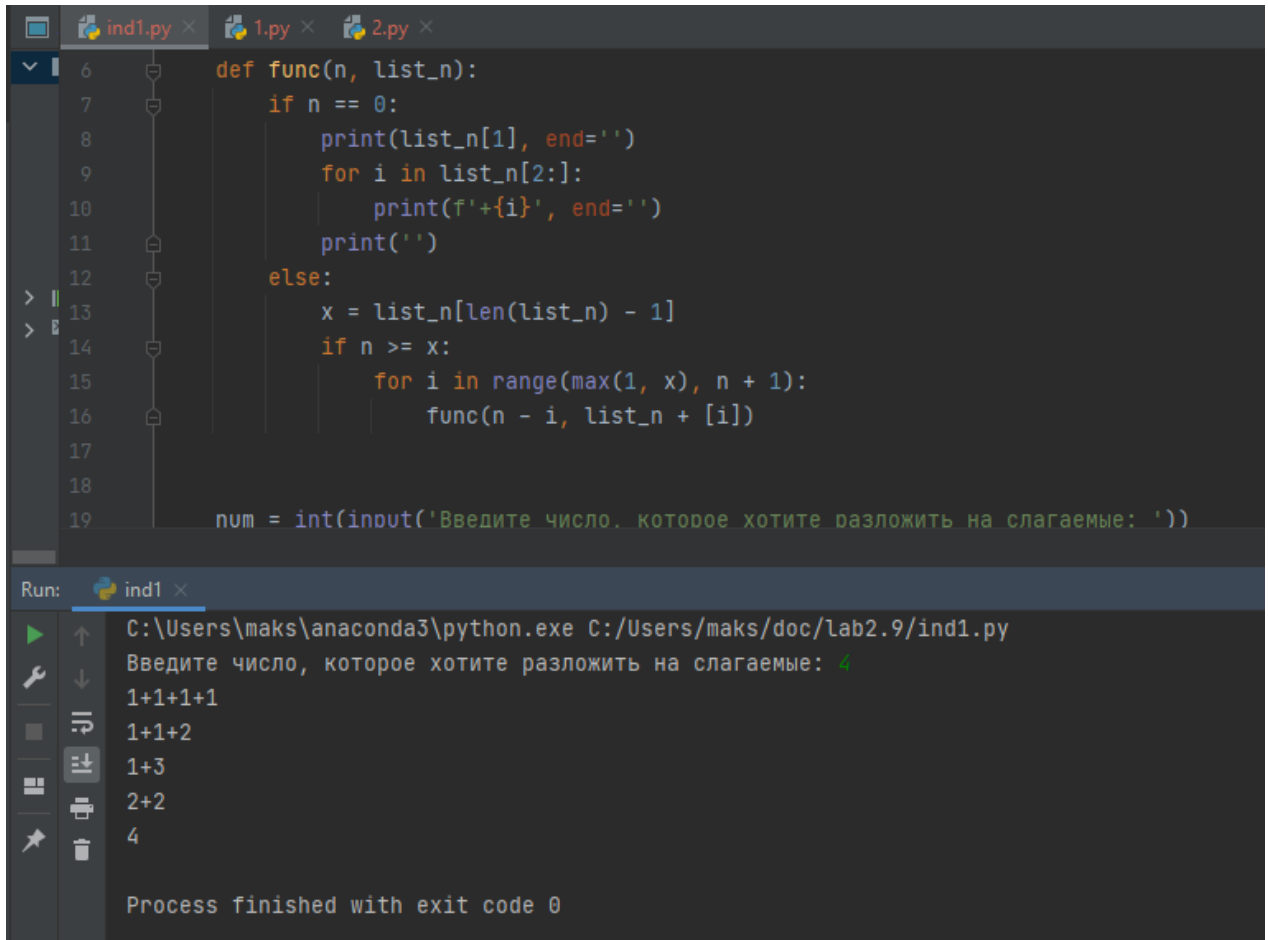
Run: 1 x

```
C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/1.py
0.002427499999999999
0.003968399999999997
Process finished with exit code 0
```

Рисунок 2 – Использование декоратора lru_cache

На основании работы декоратора можно сделать вывод о том, что он действительно уменьшает время на обработку данных, что позволяет увеличить скорость работы программы.

Выполнено индивидуальное задание



```
def func(n, list_n):
    if n == 0:
        print(list_n[1], end='')
        for i in list_n[2:]:
            print(f'#{i}', end='')
        print('')
    else:
        x = list_n[len(list_n) - 1]
        if n >= x:
            for i in range(max(1, x), n + 1):
                func(n - i, list_n + [i])

num = int(input('Введите число, которое хотите разложить на слагаемые: '))
```

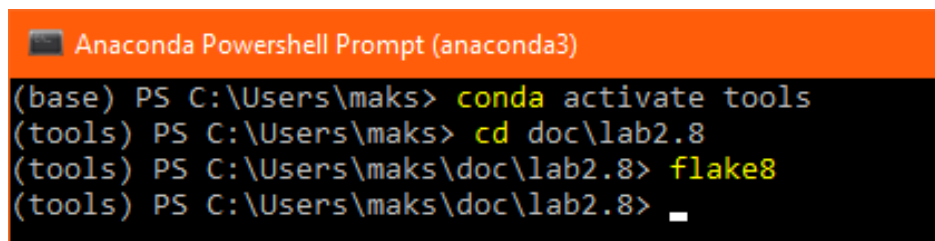
Run: ind1 x

```
C:\Users\maks\anaconda3\python.exe C:/Users/maks/doc/lab2.9/ind1.py
Введите число, которое хотите разложить на слагаемые: 4
1+1+1+1
1+1+2
1+3
2+2
4

Process finished with exit code 0
```

Рисунок 3 – Индивидуальное задание

Программы проверены на flake8, рис. 5



```
Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\maks> conda activate tools
(tools) PS C:\Users\maks> cd doc\lab2.8
(tools) PS C:\Users\maks\doc\lab2.8> flake8
(tools) PS C:\Users\maks\doc\lab2.8> _
```

Рисунок 4 – Проверка заданий

Контрольные вопросы:

1. Для чего нужна рекурсия?

Функция может содержать вызов других функций. В том числе процедура может вызвать саму себя. Никакого парадокса здесь нет – компьютер лишь последовательно выполняет встретившиеся ему в программе команды и, если встречается вызов процедуры, просто начинает выполнять эту функцию. Без разницы, какая функция дала команду это делать

2. Что называется базой рекурсии?

База рекурсии – аргументы, для которых значения функции определены (элементарные задачи).

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

При вызове подпрограммы или возникновении прерывания, в стек заносится адрес возврата – адрес в памяти следующей инструкции приостановленной программы и управление передается подпрограмме или подпрограмме-обработчику.

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

Чтобы проверить текущие параметры лимита, нужно запустить:
`sys.getrecursionlimit()`

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

Существует предел глубины возможной рекурсии, который зависит от реализации Python. Когда предел достигнут, возникает исключение `RuntimeError: Maximum Recursion Depth Exceeded`.

6. Как изменить максимальную глубину рекурсии в языке Python?

Можно изменить предел глубины рекурсии с помощью вызова:
`sys.setrecursionlimit(limit)`

7. Каково назначение декоратора `lru_cache`?

Декоратор `lru_cache` 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.

Вывод: в ходе занятия были приобретены навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.