

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

Кафедра инфокоммуникаций

Отчет по лабораторной работе №4

Рекурсия в языке Python

По дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н. В. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

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

Ход работы:

1. Изучил методические рекомендации и приступил к выполнению заданий.
2. Выполнил первое задание.

```
65 ▶ if __name__ == '__main__':
66     print("Время выполнения рекурсивной функции числа Фибоначи: ",
67           timeit(setup=fib1, number=1000))
68     print("Время выполнения итеративной функции числа Фибоначи: ",
69           timeit(setup=fib2, number=1000))
70     print("Время выполнения рекурсивной функции числа Фибоначи с
71           " использованием декоратора lru_cache: ",
72           timeit(setup=fib3, number=1000))
73     print("Время выполнения рекурсивной функции факториала: ",
74           timeit(setup=factorial1, number=1000))
75     print("Время выполнения итеративной функции факториала: ",
76           timeit(setup=factorial2, number=10000))
77     print("Время выполнения рекурсивной функции факториала с
78           " использованием декоратора lru_cache: ",
79           timeit(setup=factorial3, number=10000))
80
```

Рисунок 1 – Код первого задания

```
task_1
C:\Users\DNS\AppData\Local\Programs\Python\Python39\python.exe "D:/ИВТ/2 курс/ТПА/Лаба 4/task_1.py"
Время выполнения рекурсивной функции числа Фибоначи: 2.6600000000001622e-05
Время выполнения итеративной функции числа Фибоначи: 3.240000000000187e-05
Время выполнения рекурсивной функции числа Фибоначи с использованием декоратора lru_cache: 3.0299999999996996e-05
Время выполнения рекурсивной функции факториала: 1.6500000000002624e-05
Время выполнения итеративной функции факториала: 0.000505599999999949
Время выполнения рекурсивной функции факториала с использованием декоратора lru_cache: 0.000269800000000006
```

Рисунок 2 – Результат выполнения первого задания

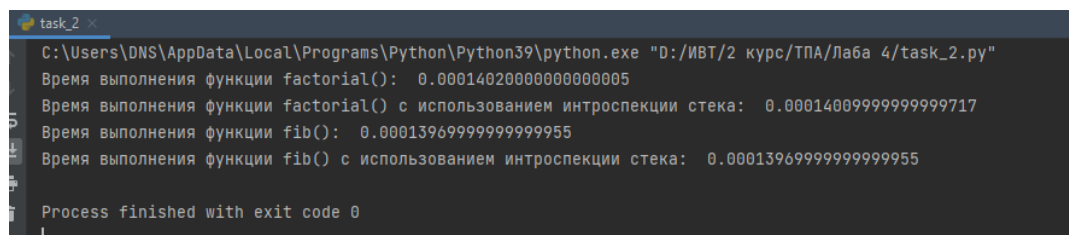
3. Приступил к выполнению второго задания.

```

73     def fib(i, current = 0, next = 1):
74         if i == 0:
75             return current
76         else:
77             return fib(i - 1, next, current + next)
78     """
79     if __name__ == '__main__':
80         print("Время выполнения функции factorial(): ",
81               timeit(setup=code1, number=10000))
82         print("Время выполнения функции factorial() с"
83               " использованием интроспекции стека: ",
84               timeit(setup=code3, number=10000))
85         print("Время выполнения функции fib(): ",
86               timeit(setup=code2, number=10000))
87         print("Время выполнения функции fib() с"
88               " использованием интроспекции стека: ",
89               timeit(setup=code4, number=10000))
90

```

Рисунок 3 – Код второго задания



```

task_2 x
C:\Users\DNS\AppData\Local\Programs\Python\Python39\python.exe "D:/ИВТ/2 курс/ТПА/Лаба 4/task_2.py"
Время выполнения функции factorial():  0.00014020000000000005
Время выполнения функции factorial() с использованием интроспекции стека:  0.000140099999999999717
Время выполнения функции fib():  0.000139699999999999955
Время выполнения функции fib() с использованием интроспекции стека:  0.000139699999999999955

Process finished with exit code 0

```

Рисунок 4 – Результат работы второго задания

Индивидуальное задание. Вариант 1.

1. Напишите рекурсивную функцию, проверяющую правильность расстановки скобок в строке.
При правильной расстановке выполняются условия:

- количество открывающих и закрывающих скобок равно.
- внутри любой пары открывающая – соответствующая закрывающая скобка, скобки расставлены правильно.

Рисунок 5 – Условие к заданию

1. Создал новый файл с расширением .py.
2. Написал код для реализации задачи

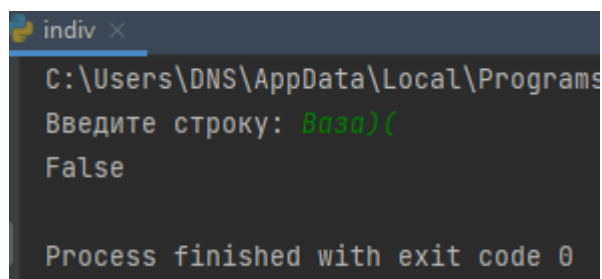
```

13 def brackets_check(s):
14     meetings = 0
15     for c in s:
16         if c == '(':
17             meetings += 1
18         elif c == ')':
19             meetings -= 1
20             if meetings < 0:
21                 return False
22
23     return meetings == 0
24
25
26 if __name__ == '__main__':
27     if brackets_check(input('Введите строку: ')):
28         print('True')
29     else:
30         print('False')
31

```

Рисунок 6 – Код для выполнения задачи

3. Проверил работоспособность кода.

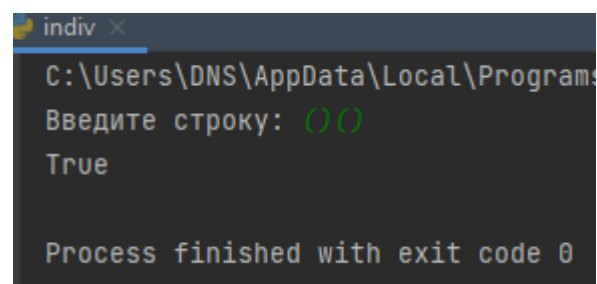


```

indiv x
C:\Users\DNS\AppData\Local\Programs
Введите строку: Baza(
False
Process finished with exit code 0

```

Рисунок 7 – Результат кода, пример 1

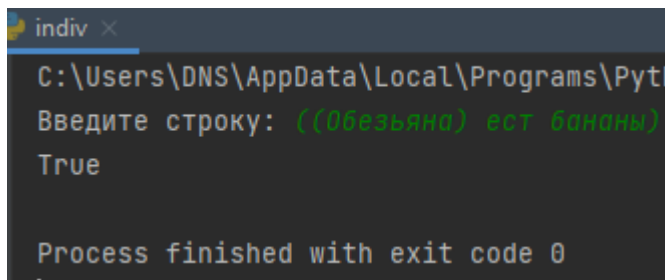


```

indiv x
C:\Users\DNS\AppData\Local\Programs
Введите строку: ()
True
Process finished with exit code 0

```

Рисунок 8 – Результат кода, пример 2



```
indiv x
C:\Users\DNS\AppData\Local\Programs\Python\Python39\python.exe
Введите строку: ((Обезьяна) ест бананы)
True

Process finished with exit code 0
```

Рисунок 9, пример 3

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 – 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 функций?

Стек в Python – это линейная структура данных, в которой данные расположены объектами друг над другом. Он хранит данные в режиме LIFO (Last in First Out). Данные хранятся в том же порядке, в каком на кухне тарелки располагаются одна над другой. Мы всегда выбираем последнюю тарелку из стопки тарелок. В стеке новый элемент вставляется с одного конца, и элемент может быть удален только с этого конца.

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`.

6. Как изменить максимальную глубину рекурсии в языке Python?

Изменить максимальную глубину рекурсии можно с помощью `sys.setrecursionlimit()`.

7. Каково назначение декоратора `lru_cache` ?

Декоратор `lru_cache` является полезным инструментом, который можно использовать для уменьшения количества лишних вычислений. Декоратор оборачивает функцию с переданными в нее аргументами и запоминает возвращаемый результат, соответствующий этим аргументам.

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 выполнения лабораторной работы приобрел навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.