

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 №2.9
дисциплины «Программирование на Python»
Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Рекурсия в языке Python

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

Ход работы

7. Самостоятельно изучите работу со стандартным пакетом Python `timeit`. Оцените с помощью этого модуля скорость работы итеративной и рекурсивной версий функций `factorial` и `fib`. Во сколько раз изменится скорость работы рекурсивных версий функций `factorial` и `fib` при использовании декоратора `lru_cache`? Приведите в отчет и обоснуйте полученные результаты.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import timeit as ti
from functools import lru_cache

def factorial(n):
    """Рекурсивное вычисление факториала"""
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

def fib(n):
    """Рекурсивное вычисление числа фибонначи"""
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)

def factorial_iter(n):
    """Вычисление факториала итеративно"""
    rez = 1
    while n > 1:
        rez *= n
        n -= 1

    return rez

def fib_iter(n):
    """Вычисление числа фибонначи итеративно"""
    a, b = 0, 1
    while n > 0:
        a, b = b, a + b
        n -= 1

    return a

@lru_cache
def factorial_lru(n):
    """
    Рекурсивное вычисление факториала
    Оптимизация с помощью lru_cache
    """
```

```

"""
if n == 0:
    return 1
else:
    return n * factorial_lru(n - 1)

@lru_cache
def fib_lru(n):
    """
    Рекурсивное вычисление числа фибонначи
    Оптимизация с помощью lru_cache
    """
    if n == 0 or n == 1:
        return n
    else:
        return fib_lru(n - 2) + fib_lru(n - 1)

def timer(func, n):
    repeat = 30
    print(f'{func[1]} время работы: ', ti.timeit(lambda: func[0](n),
number=repeat) / repeat)

def main():
    funcs = {
        factorial: "Рекурсивный факториал",
        factorial_iter: "Итеративный факториал",
        fib: "Рекурсивный фибонначи",
        fib_iter: "Итеративный фибонначи",
        factorial_lru: "Рекурсивный факториал с lru_cache",
        fib_lru: "Рекурсивный фибонначи с lru_cache"
    }

    n = 15
    for func in funcs.items():
        timer(func, n)

if __name__ == '__main__':
    main()

```

```

Рекурсивный факториал время работы: 1.7133332827749352e-06
Итеративный факториал время работы: 1.3199999860565488e-06
Рекурсивный фибонначи время работы: 0.0001709633333424184
Итеративный фибонначи время работы: 1.26000000142507877e-06
Рекурсивный факториал с lru_cache время работы: 5.800000508315862e-07
Рекурсивный фибонначи с lru_cache время работы: 5.066666441659133e-07

```

Рисунок 1. Результат выполнения программы

При использовании lru_cache скорость работы рекурсивных функций увеличивается в несколько раз. Для рекурсивных функций вычисления факториала и чисел Фибоначчи с большими значениями n, где происходит

множественное повторение одних и тех же вычислений, использование lru_cache может ускорить выполнение функций на несколько порядков.

Индивидуальное задание:

2. В строке могут присутствовать скобки как круглые, так и квадратные скобки. Каждой открывающей скобке соответствует закрывающая того же типа (круглой – круглая, квадратной – квадратная). Напишите рекурсивную функцию, проверяющую правильность расстановки скобок в этом случае.

Пример неправильной расстановки: ([]).

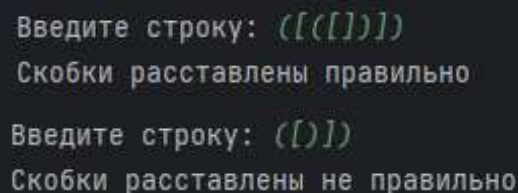
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def check_par(par_str):
    """Проверка правильности расстановки скобок"""
    if len(par_str) == 0:
        return True
    else:
        left = par_str[0]
        right = par_str[-1]
        kf = "[".find(left)
        if kf == -1:
            return False
        if right == "]"[kf]:
            return check_par(par_str[1:len(par_str) - 1])
        else:
            return False

def main():
    par = input("Введите строку: ")

    if check_par(par):
        print("Скобки расставлены правильно")
    else:
        print("Скобки расставлены не правильно")

if __name__ == '__main__':
    main()
```



```
Введите строку: ([([)])
Скобки расставлены правильно
Введите строку: ([)])
Скобки расставлены не правильно
```

Рисунок 2. Результат работы программы

Контрольные вопросы:

1. Рекурсия используется для повторного вызова функции из самой себя, что позволяет решать задачи, которые могут быть разбиты на более простые подзадачи.

2. База рекурсии - это условие, при котором рекурсивные вызовы прекращаются, и функция начинает возвращать значения.

3. Стек программы - это структура данных, которая используется для хранения временных данных вызовов функций. При вызове функции, информация о вызове помещается в стек, а при завершении функции - извлекается из стека.

4. Текущее значение максимальной глубины рекурсии в Python можно получить с помощью `sys.getrecursionlimit()`.

5. Если число рекурсивных вызовов превысит максимальную глубину рекурсии в Python, то будет сгенерировано исключение `RecursionError`.

6. Максимальную глубину рекурсии можно изменить с помощью функции `sys.setrecursionlimit()`.

7. Декоратор `lru_cache` используется для кеширования результатов вызовов функции с определенными аргументами, что позволяет избежать повторных вычислений.

8. Хвостовая рекурсия - это вид рекурсии, при котором рекурсивный вызов является последней операцией в функции. Оптимизация хвостовых вызовов проводится путем замены рекурсивных вызовов на циклы, что позволяет избежать переполнения стека при большом числе рекурсивных вызовов.

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

