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 ЛАБОРАТОРНОЙ РАБОТЕ №12
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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.

Ход выполнения работы

1. Создал репозиторий GitHub.

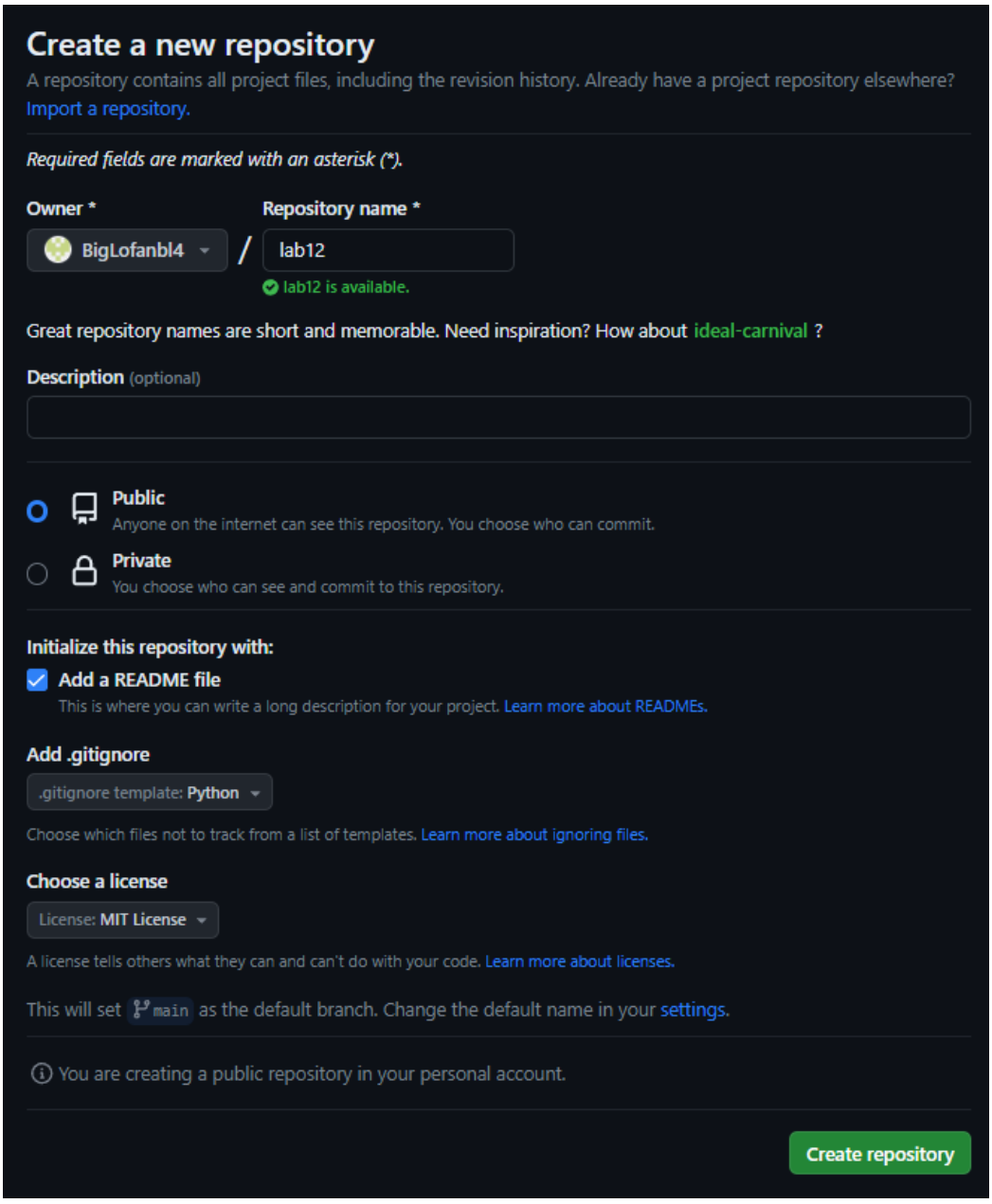

Рисунок 1.1 – Создание репозитория

2. Оцените с помощью этого модуля скорость работы итеративной и рекурсивной версий функций factorial и fib. Провел замеры скорости выполнения итеративной и рекурсивной функций factorial (n=50) и fib (n=20).

```
Функция факториала:
Время выполнения рекурсивной функции 5.950010381639004e-05
Время выполнения итеративной функции 4.409998655319214e-05

Функция чисел Фибоначчи:
Время выполнения рекурсивной функции 1.742133499821648
Время выполнения итеративной функции 2.2900057956576347e-05
```

Рисунок 2.1 – Время вычислений функций

Во сколько раз измениться скорость работы рекурсивных версий функций factorial и fib при использовании декоратора lru_cache?

```
Функция факториала:
Время выполнения рекурсивной функции 5.319993942975998e-05
Время выполнения итеративной функции 4.519987851381302e-05

Функция чисел Фибоначчи:
Время выполнения рекурсивной функции 1.3399869203567505e-05
Время выполнения итеративной функции 2.1900050342082977e-05
```

Рисунок 2.2 – Время вычисления функций с @lru_cache

Как видно из рисунка, скорость вычисления факториала для рекурсивной функции увеличилась примерно на 1,1 раза., скорость вычисления чисел Фибоначчи увеличилась 1,3 раза.

3. Проработал пример с оптимизацией хвостовых вызовов в Python. С помощью пакета timeit оценил скорость работы функций factorial и fib с использованием интроспекции стека и без использования интроспекции стека.

```
Время выполнения функции factorial: 5.250005051493645e-05
Время выполнения функции fib: 1.719989813864231e-05
```

Рисунок 3.1 – Время выполнения функций без интроспекции стека

```
Время выполнения функции factorial: 0.0010441998019814491
Время выполнения функции fib: 0.0003978000022470951
```

Рисунок 3.2 – Время выполнения функций с интроспекцией стека

4. Индивидуальное задание.

14. Напишите рекурсивную функцию, которая вычисляет $y = \sqrt[k]{x}$ по следующей формуле:

$$y_0 = 1; y_{n+1} = y_n + \frac{x/y_n^{k-1} - y_n}{k}, \quad (4)$$

$n = 0, 1, 2, \ldots$. За ответ принять приближение, для которого выполняется $|y_n - y_{n+1}| < \varepsilon$, где $\varepsilon = 0,0001$.

Рисунок 4.1 – Условие задания

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

# 14 вариант
def sqrt(x, k, st=1, eps=1e-4):
    y = st + ((x / st ** (k - 1)) - st) / k
    if abs(st - y) < eps:
        return y
    else:
        return sqrt(x, k, y, eps)


if __name__ == "__main__":
    print(sqrt(8, 3))
```

Рисунок 4.2 – Код программы

```
2.0000000000120624
```

Рисунок 4.3 – Вывод программы

Ответы на контрольные вопросы

1. Рекурсия - это процесс, при котором функция вызывает саму себя. Она используется для решения задач, которые могут быть разбиты на более мелкие подзадачи. Рекурсия позволяет писать более компактный и легко читаемый код.

2. База рекурсии - это условие, при котором рекурсивный процесс завершается. Это условие должно быть определено внутри рекурсивной функции, чтобы избежать бесконечной рекурсии.

3. Стек программы - это область памяти, которая используется для хранения информации о вызовах функций. При вызове функции информация о вызове помещается в стек, а при возврате из функции информация удаляется из стека. Это позволяет программе сохранять контекст выполнения функций и возвращаться к ним позже.

4. Максимальную глубину рекурсии можно получить с помощью функции sys.getrecursionlimit().

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

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

7. Декоратор lru_cache используется для кэширования результатов выполнения функции. Он сохраняет результаты выполнения функции в памяти и возвращает их при повторном вызове функции с теми же аргументами. Это может значительно ускорить выполнение функции в случае, если она вызывается многократно с одними и теми же аргументами.

8. Хвостовая рекурсия - это рекурсия, при которой вызов рекурсивной функции является последней операцией в функции. Оптимизация хвостовых вызовов заключается в замене рекурсивной функции на итеративную функцию, что может улучшить производительность и избежать переполнения стека вызовов.