

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 «Программирование на Python»

Выполнил:
Горбунов Данила Евгеньевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

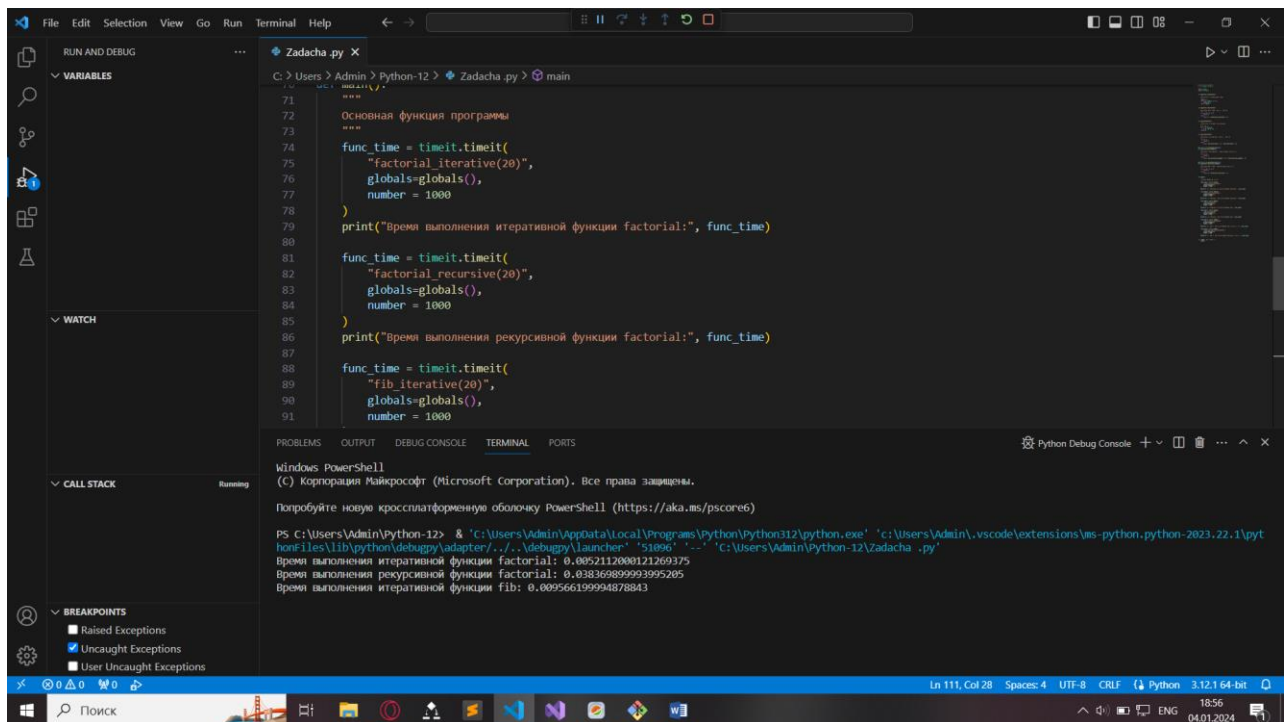
Ставрополь, 2023 г.

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

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл `gitignore` необходимыми правилами.
3. Организовал созданный репозиторий в соответствии с моделью ветвления `git-flow`.
4. Решил следующую задачу: 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 измениться скорость работы рекурсивных версий функций `factorial` и `fib` при использовании декоратора `lru_cache`? Приведите в отчет и обоснуйте полученные результаты.



```
File Edit Selection View Go Run Terminal Help
Zadacha.py X
C:\Users\Admin> Python-12 > Zadacha.py > main
71
72 Основная функция программы
73
74 func_time = timeit.timeit(
75     "factorial_iterative(20)",
76     globals=globals(),
77     number = 1000
78 )
79 print("Время выполнения итеративной функции factorial:", func_time)
80
81 func_time = timeit.timeit(
82     "factorial_recursive(20)",
83     globals=globals(),
84     number = 1000
85 )
86 print("Время выполнения рекурсивной функции factorial:", func_time)
87
88 func_time = timeit.timeit(
89     "fib_iterative(20)",
90     globals=globals(),
91     number = 1000
92 )

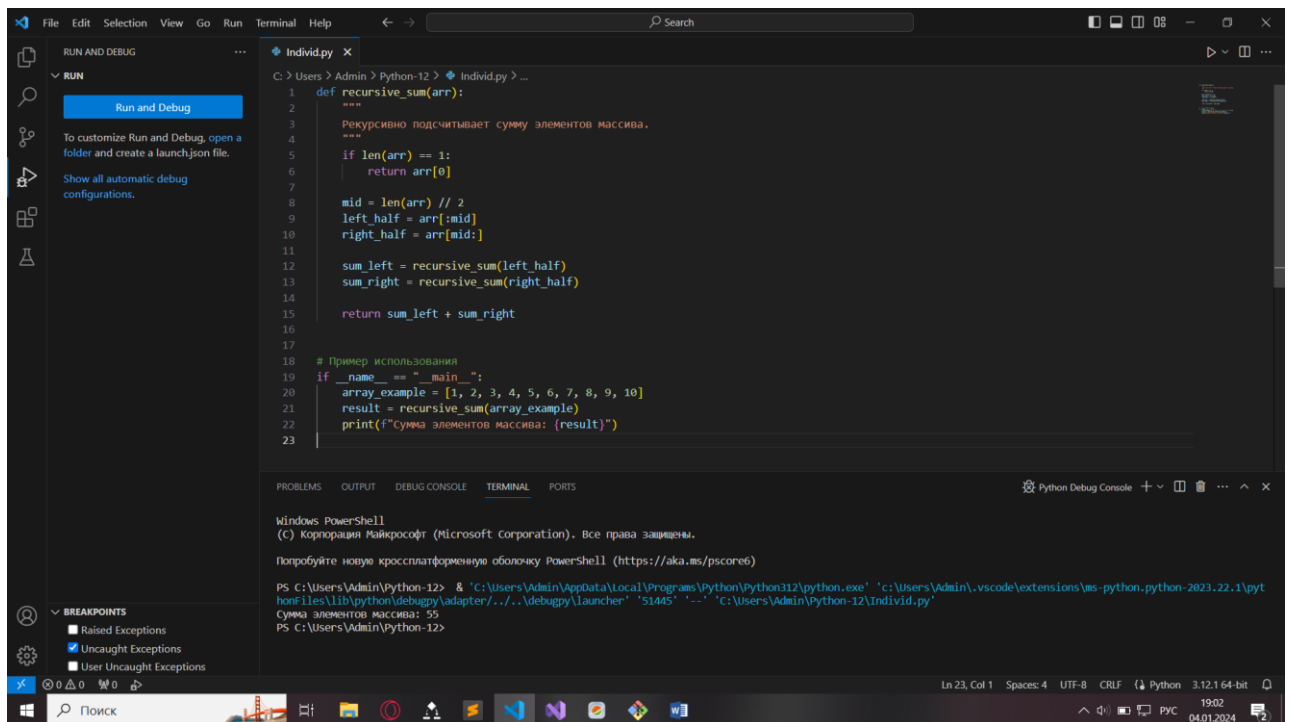
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попройте новую кроссплатформенную оболочку PowerShell (https://aka.ms/powershell)
PS C:\Users\Admin\Python-12> & "C:\Users\Admin\AppData\Local\Programs\Python\Python312\python.exe" "c:\Users\Admin\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher" "51096" "-..." "c:\Users\Admin\Python-12\Zadacha .py"
Время выполнения итеративной функции factorial: 0.0052112000121269375
Время выполнения рекурсивной функции factorial: 0.038369899993995205
Время выполнения итеративной функции fib: 0.009566199994878843
```

Рисунок 1. Результат работы программы из задачи 1

Анализируя полученные результаты времени работы функций, можно сказать, что итеративный подход в вычислении чисел Фибоначчи и факториалов на несколько порядков эффективнее, чем рекурсивный. Использование декоратора `@lru_cache` крайне значительно уменьшает время работы, это достигается путем оптимизации одинаковых повторяющихся подсчетов рекурсивной функции.

7. Выполнил индивидуальное задание.

Создайте функцию, подсчитывающую сумму элементов массива по следующему алгоритму: массив делится пополам, подсчитываются и складываются суммы элементов в каждой половине. Сумма элементов в половине массива подсчитывается по тому же алгоритму, то есть снова путем деления пополам. Деления происходят, пока в получившихся кусках массива не окажется по одному элементу и вычисление суммы, соответственно, не станет тривиальным.



```
1 def recursive_sum(arr):
2     """
3     Рекурсивно подсчитывает сумму элементов массива.
4     """
5     if len(arr) == 1:
6         return arr[0]
7
8     mid = len(arr) // 2
9     left_half = arr[:mid]
10    right_half = arr[mid:]
11
12    sum_left = recursive_sum(left_half)
13    sum_right = recursive_sum(right_half)
14
15    return sum_left + sum_right
16
17
18 # Пример использования
19 if __name__ == "__main__":
20     array_example = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
21     result = recursive_sum(array_example)
22     print(f"Сумма элементов массива: {result}")
23
```

Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/powershell)

```
PS C:\Users\Admin\Python-12> & 'C:\Users\Admin\AppData\Local\Programs\Python\Python312\python.exe' 'C:\Users\Admin\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '51445' '-.' 'C:\Users\Admin\Python-12\Individ.py'
Сумма элементов массива: 55
PS C:\Users\Admin\Python-12>
```

Рисунок 2. Результат работы программы из индивидуального задания

Контрольные вопросы

1. Для чего нужна рекурсия?

Рекурсия используется в программировании для решения задач, которые могут быть разбиты на более мелкие подзадачи того же типа. Она позволяет функции вызывать саму себя, что упрощает решение сложных задач путем разделения их на более простые подзадачи. Рекурсия также широко используется в алгоритмах, таких как алгоритмы обхода деревьев и графов.

2. Что называется базой рекурсии?

Условие, при котором рекурсивные вызовы функции прекращаются и начинается возврат из рекурсивных вызовов. Это базовый случай, который предотвращает бесконечное выполнение рекурсивной функции и обеспечивает завершение процесса.

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

Стек программы – это структура данных, которая хранит информацию о вызовах функций во время выполнения программы. При вызове функции, информация о текущем состоянии функции, такая как локальные переменные и адрес возврата, помещается в стек. Когда функция завершает выполнение, информация извлекается из стека, и управление передается обратно вызывающей функции. Это позволяет программе возвращаться к предыдущему состоянию после завершения выполнения функции.

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

В языке Python можно получить текущее значение максимальной глубины рекурсии с помощью функции `sys.getrecursionlimit()`. Она возвращает текущее максимальное количество рекурсивных вызовов, которое может быть выполнено до возникновения ошибки "RecursionError".

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

Если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python, возникнет ошибка "RecursionError". Это произойдет, когда программа пытается выполнить больше рекурсивных вызовов, чем разрешено текущей максимальной глубиной рекурсии.

6. Как изменить максимальную глубину рекурсии в языке Python?

Максимальную глубину рекурсии в языке Python можно изменить с помощью функции `sys.setrecursionlimit()`. Однако, изменение этого значения должно быть осуществлено с осторожностью, так как слишком большая глубина рекурсии может привести к переполнению стека и ошибкам выполнения.

7. Каково назначение декоратора `lru_cache` ?

Используется для кэширования результатов вызовов функции с определенными аргументами. Он сохраняет результаты предыдущих вызовов функции, чтобы избежать повторных вычислений при повторных вызовах с теми же аргументами. Это может значительно улучшить производительность функций, особенно при выполнении тяжелых вычислений.

8. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?

Это особый вид рекурсии, при котором рекурсивный вызов является последней операцией в функции. Оптимизация хвостовых вызовов заключается в том, что компилятор или интерпретатор может заменить рекурсивный вызов на цикл, что позволяет избежать увеличения стека вызовов. Это позволяет снизить использование памяти и улучшить производительность программ.

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