МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №12

по дисциплине основы программной инженерии

Выполнила: Гробова
Софья Кирилловна,

2 курс, группа ПИЖ-б-о-20-1,

Проверил: Доцент кафедры
инфокоммуникаций, Воронкин Р.А.

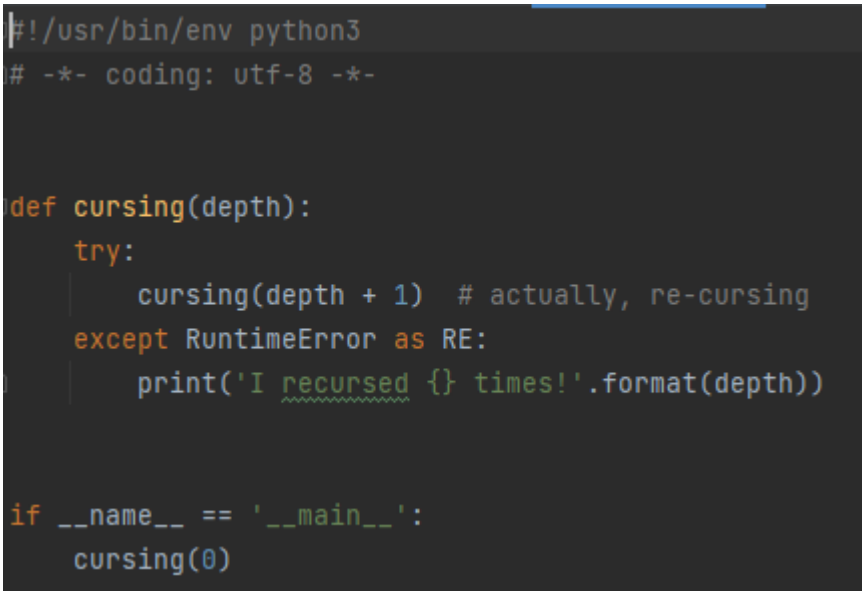
Ставрополь, 2021 г

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


def recursion(n):
    if n == 1:
        return 1
    return n + recursion(n - 1)


def main():
    n = int(input("Введите n = "))
    summa = 0
    for i in range(1, n + 1):
        summa += i
    print(f"Сумма посчитаная без рекурсии = {summa}")
    print(f"Сумма посчитанная с помощью рекурсии = {recursion(n)}")


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

Пример №1

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

from functools import lru_cache


@lru_cache
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)


def main():
    n = int(input("Введите n = "))
    print(f"Вычисление {n} числа Фибоначи с помощью рекурсии = {fib(n)}")


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

Пример №2



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


def cursing(depth):
    try:
        cursing(depth + 1)  # actually, re-cursing
    except RuntimeError as RE:
        print('I recursed {} times!'.format(depth))


if __name__ == '__main__':
    cursing(0)
```

Пример №3

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Эта программа показыает работу декоратора, который производит оптимизацию
# хвостового вызова. Он делает это, вызывая исключение, если оно является его
# прародителем, и перехватывает исключения, чтобы вызвать стек.
import sys


class TailRecurseException(Exception):
    def __init__(self, args, kwargs):
        self.args = args
        self.kwargs = kwargs


def tail_call_optimized(g):
    def func(*args, **kwargs):
        f = sys._getframe()
        if f.f_back and f.f_back.f_back and f.f_back.f_back.f_code == f.f_code:
            raise TailRecurseException(args, kwargs)
        else:
            while True:
                try:
                    return g(*args, **kwargs)
                except TailRecurseException as e:
                    args = e.args
                    kwargs = e.kwargs

    func.__doc__ = g.__doc__
    return func


@tail_call_optimized
def factorial(n, acc=1):
    """calculate a factorial"""
    if n == 0:
        return acc
    return factorial(n - 1, n * acc)


@tail_call_optimized
```

```python
@tail_call_optimized
def fib(i, current=0, next=1):
    if i == 0:
        return current
    else:
        return fib(i - 1, next, current + next)


if __name__ == '__main__':
    print(factorial(10000))
    print(fib(10000))
```

Пример №4

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

def recursion(arr, num):
    if num <= 1:
        return arr[0]
    else:
        arr1 = arr[0:len(arr) // 2]
        arr2 = arr[len(arr) // 2:len(arr)]
        res = int(recursion(arr1, len(arr1)))
        res += int(recursion(arr2, len(arr2)))
        return res


if __name__ == '__main__':
    arr = input('Введите числа через пробел: ')
    arr = arr.split()
    print(f'Sum = {recursion(arr, len(arr))}')
```

Код программы индивидуального задания, вариант 7

Контрольные вопросы:

1. Для чего нужна рекурсия?

В некоторых случаях лучше использовать рекурсию (например, путешествие по дереву), в таких случаях более естественно использовать "think recursively". Однако, если использование циклов не сложнее и намного сложнее, чем рекурсия, я предпочитаю их.

2. Что называется базой рекурсии?

База рекурсии – это такие аргументы функции, которые делают задачу настолько простой, что решение не требует дальнейших вложенных вызовов. Рекурсивно определяемая структура

данных – это структура данных, которая может быть определена с использованием самой себя

3. Самостоятельно изучите что является стеком программы. Как используется стек программы

при вызове функций?

Стек вызовов – в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм (процедур, функций) в программу (или подпрограмму, при вложенных или рекурсивных вызовах) и/или для возврата в программу из обработчика прерывания

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

Вывести на печать значение изменяемого параметра функции в условии, которое определяет конец рекурсии

5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии

в языке Python?

Произойдёт переполнение памяти и функция упадёт в ошибку

6. Как изменить максимальную глубину рекурсии в языке Python?

Sys.setrecurionlimit(limit)

7. Каково назначение декоратора lru_cache ?

Декоратор @lru_cache() модуля functools 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

Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах.