

**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»**

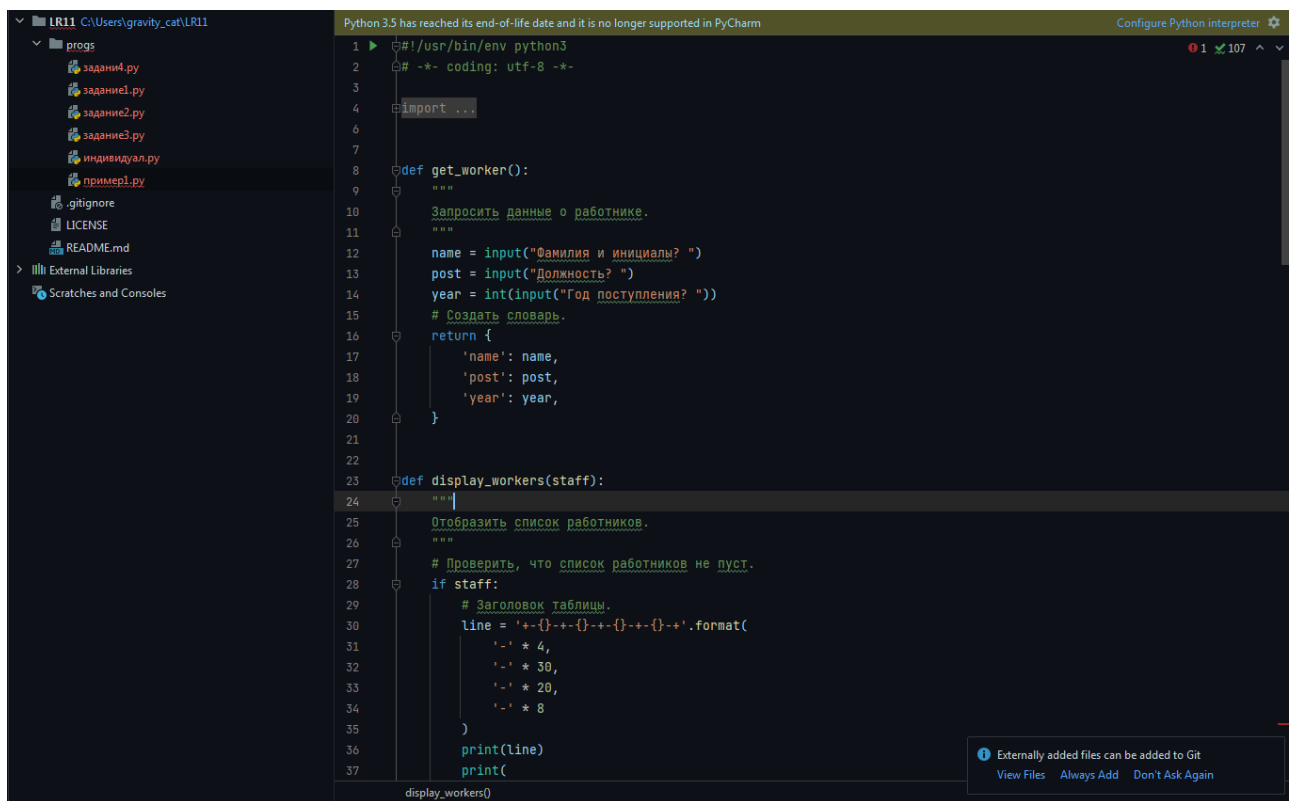
Выполнил:
Гребен Владислав
Александрович,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

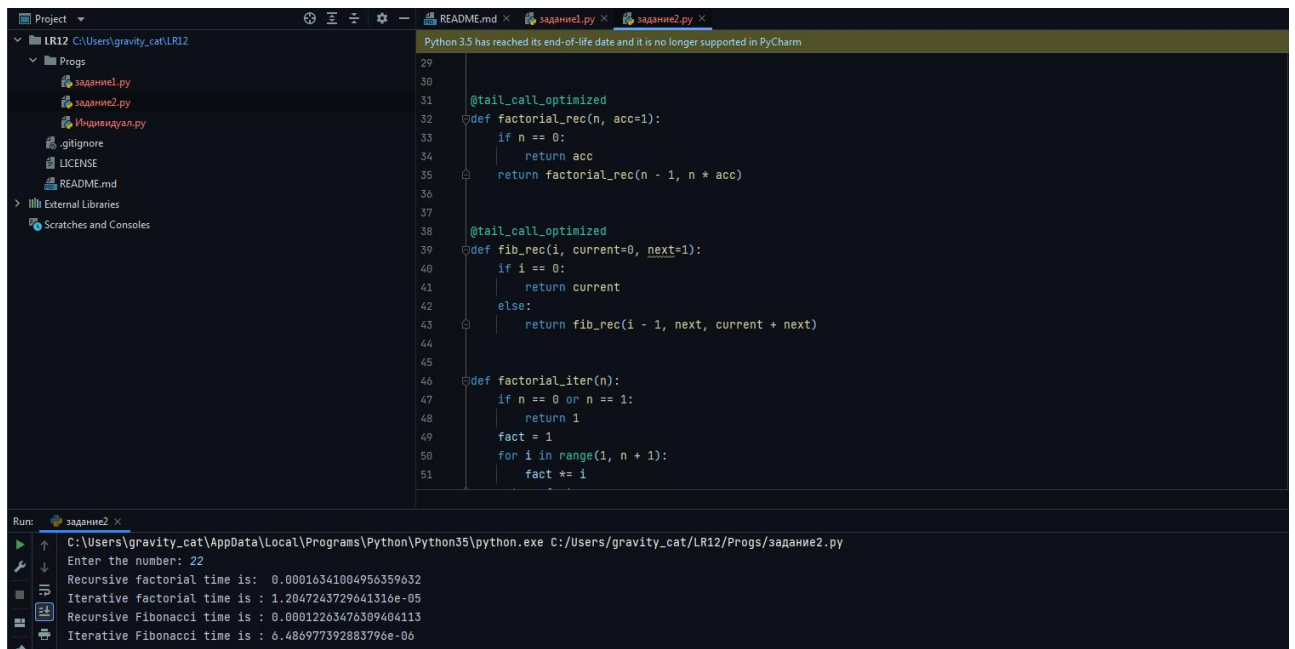
Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import ...
5
6
7
8  def get_worker():
9      """
10     Запросить данные о работнике.
11     """
12     name = input("Фамилия и инициалы? ")
13     post = input("Должность? ")
14     year = int(input("Год поступления? "))
15     # Создать словарь.
16     return {
17         'name': name,
18         'post': post,
19         'year': year,
20     }
21
22
23 def display_workers(staff):
24     """
25     Отобразить список работников.
26     """
27     # Проверить, что список работников не пуст.
28     if staff:
29         # Заголовок таблицы.
30         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
31             '-' * 4,
32             '-' * 30,
33             '-' * 20,
34             '-' * 8
35         )
36         print(line)
37         print(
```

Рисунок 12.1 – Задание №1



```
29
30
31 @tail_call_optimized
32 def factorial_rec(n, acc=1):
33     if n == 0:
34         return acc
35     return factorial_rec(n - 1, n * acc)
36
37
38 @tail_call_optimized
39 def fib_rec(i, current=0, next=1):
40     if i == 0:
41         return current
42     else:
43         return fib_rec(i - 1, next, current + next)
44
45
46 def factorial_iter(n):
47     if n == 0 or n == 1:
48         return 1
49     fact = 1
50     for i in range(1, n + 1):
51         fact *= i
```

Run: задание2

C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR12/Progs/задание2.py

Enter the number: 22

Recursive factorial time is: 0.00016341004956359632

Iterative factorial time is : 1.2047243729641316e-05

Recursive Fibonacci time is : 0.00012263476309404113

Iterative Fibonacci time is : 6.486977392883790e-06

Рисунок 12.2 – Задание №2

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 5

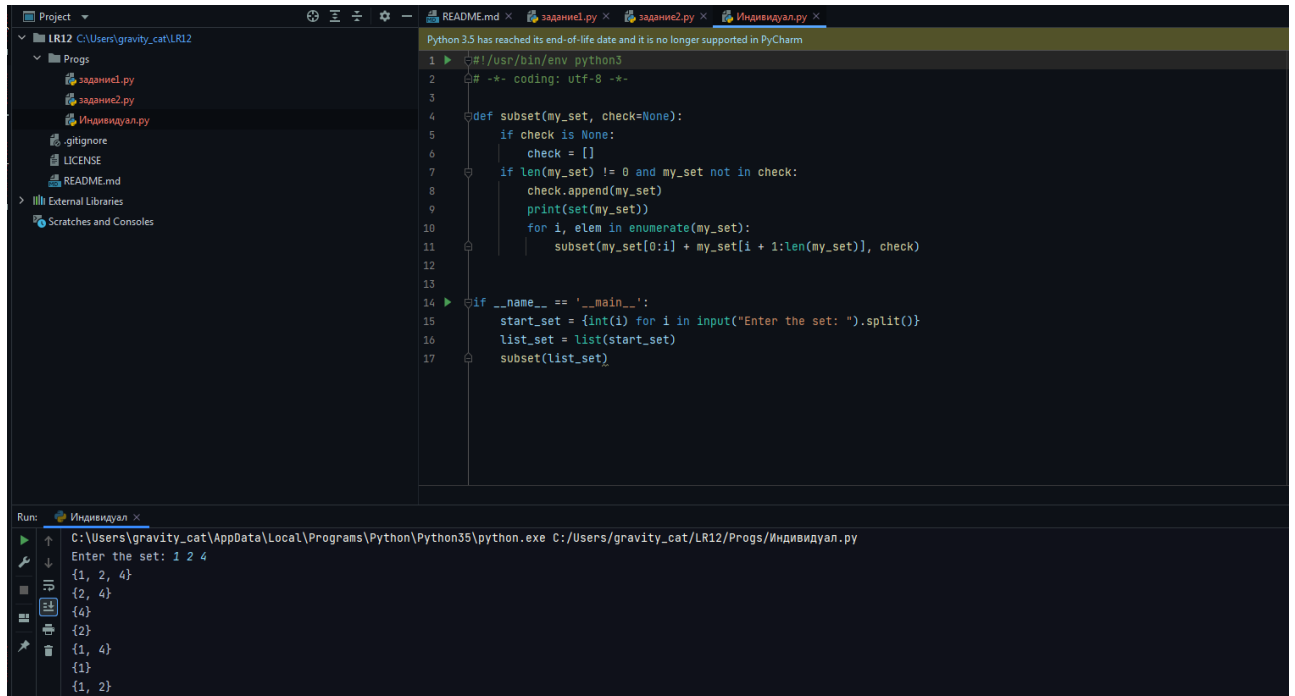


Рисунок 12.3 – ИДЗ

ОТВЕТЫ НА ВОПРОСЫ

1. Для чего нужна рекурсия? Рекурсия может работать в обратную сторону, иногда рекурсивное решение проще, чем итеративное решение. Это очевидно при реализации обращения связанного списка.
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 функции – 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 (в том числе при переключении задач в многозадачной среде). Когда функция производит вложенный вызов, происходит следующее:
 - 1) Выполнение текущей функции приостанавливается.
 - 2) Контекст выполнения, связанный с ней, запоминается в специальной структуре данных – стеке контекстов выполнения.
 - 3) Выполняются вложенные вызовы, для каждого из которых создаётся свой контекст выполнения.
 - 4) После их завершения старый контекст достаётся из стека, и выполнение внешней функции возобновляется с того места, где она была остановлена.
4. Как получить текущее значение максимальной глубины рекурсии в языке Python? Выполнить команду `sys.getrecursionlimit()`
5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python? Когда предел достигнут, возникает исключение `RuntimeError : RuntimeError: Maximum Recursion Depth`
6. Как изменить максимальную глубину рекурсии в языке Python? Нужно выполнить команду: `sys.setrecursionlimit(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. Такое поведение может сэкономить время и ресурсы, когда дорогая или связанная с вводом/выводом функция периодически вызывается с одинаковыми аргументами.
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 хвостового вызова (ТСО) — это способ автоматического сокращения рекурсии в рекурсивных функциях. Для

оптимизации, к примеру, можно постараться каждый раз при вызове функции сохранять только кадр стека внутренней вызываемой функции, что может значительно сэкономить память.