**🎯 Цель и задачи работы  
Шамиев Ринат ИС 24-22**

**Цель работы:**  
Изучить принцип работы рекурсии и научиться применять рекурсивные функции для решения различных вычислительных задач.

**Задачи:**

1. Ознакомиться с понятием рекурсии и её принципом работы.
2. Реализовать примеры рекурсивных функций на языке Python.
3. Исследовать работу базового условия и стек вызовов.
4. Сравнить рекурсивные и итерационные алгоритмы.
5. Научиться применять рекурсию для решения практических задач (поиск, сортировка, вычисления).

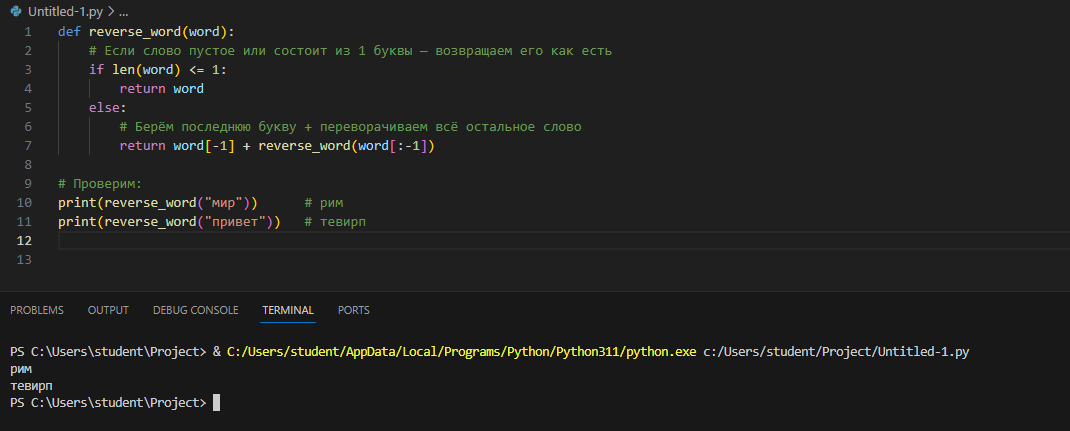
**📘 Теоретические сведения по теме**

**Рекурсия** — это способ организации вычислений, при котором функция вызывает саму себя до достижения **базового условия (точки выхода)**.

Каждый рекурсивный вызов порождает **новый контекст выполнения**, который сохраняется в **стеке вызовов**. Когда функция достигает базового условия, начинается обратный процесс — возврат результатов из стека.

Я написал **свою рекурсивную функцию**, которая переворачивает слово.  
Функция называется reverse\_word.  
Она проверяет: если слово короткое (1 буква) — просто возвращает его.  
Если слово длиннее — берёт **последнюю букву** и **добавляет к ней перевёрнутое остальное слово**.  
Так функция вызывает саму себя, пока не дойдёт до одной буквы.

Например, для слова **"мир"**:  
берётся "р", потом "и", потом "м", и получается **"рим"**.

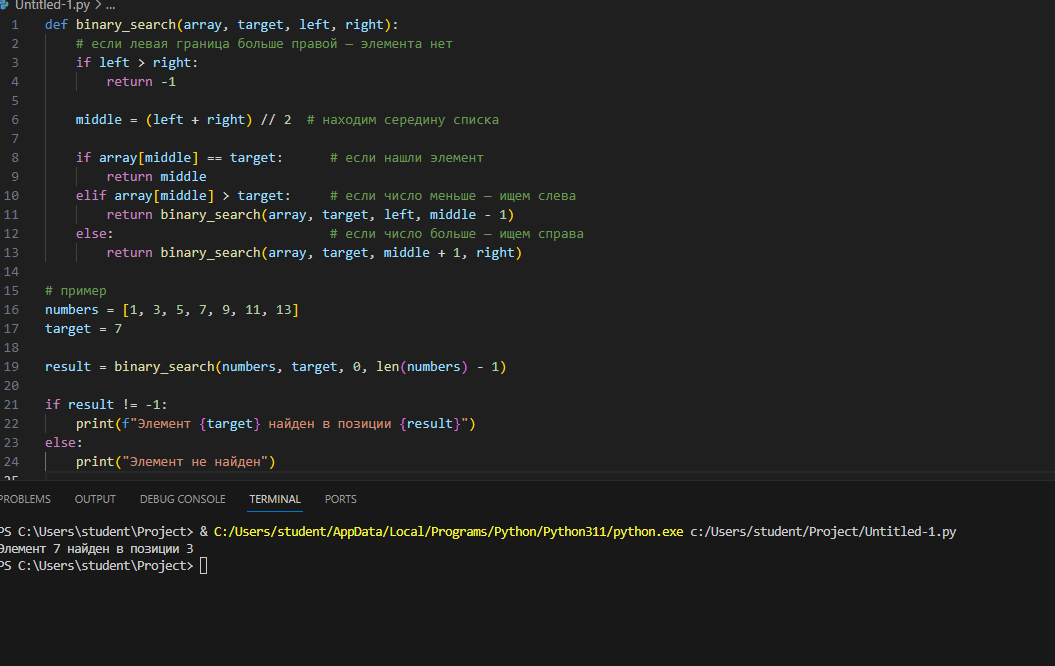
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Функция binary\_search ищет число **в середине списка**.

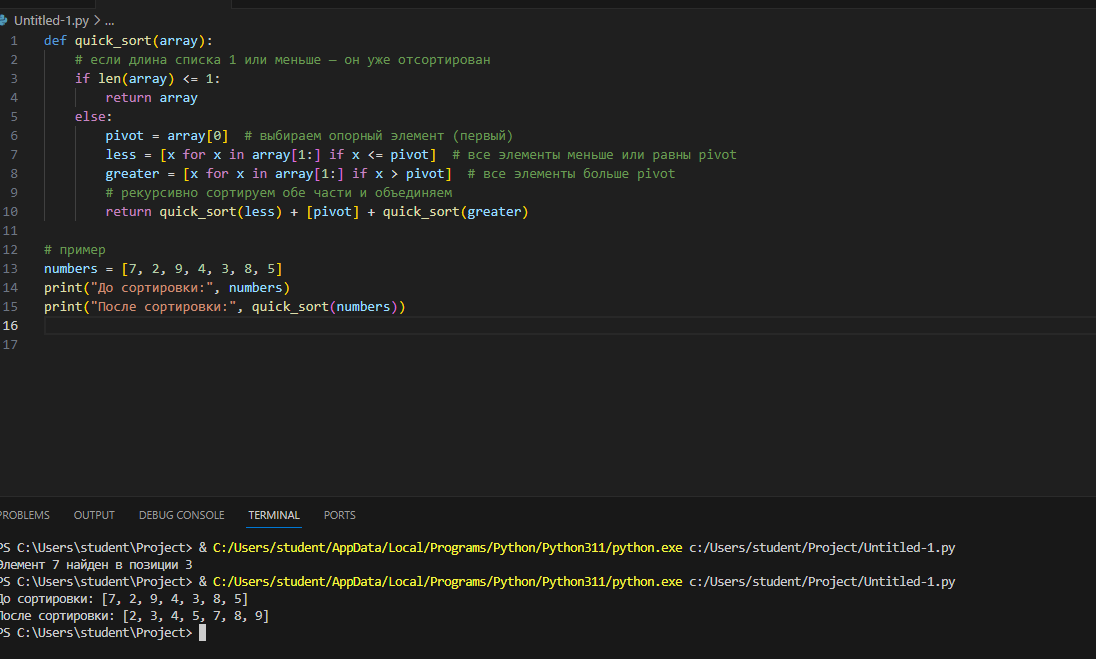
 Если это не то число — рекурсивно ищет **в левой или правой половине**.

 Так список каждый раз уменьшается в 2 раза.

 Когда элемент найден — возвращается его **индекс**.

 Если не найден — возвращается -1.

*   
    
    
    
    
    
    
    
    
    
    
    
    
    
    
  Функция **quick\_sort** сортирует список **с помощью рекурсии**.
* Мы выбираем один элемент — **опорный (pivot)**.
* Делим список на две части:
  + less — все числа **меньше** опорного,
  + greater — все числа **больше** опорного.
* Потом рекурсивно сортируем обе части и **соединяем** их обратно.

  
  
  
  
  
**1. Что такое рекурсия и как она реализуется в программах?**  
Рекурсия — это когда функция вызывает саму себя, пока не выполнится условие остановки (базовое условие).

**2. Что представляет собой базовое условие рекурсии?**  
Это условие, при котором функция перестаёт вызывать саму себя и возвращает результат.

**3. Чем итерационный алгоритм отличается от рекурсивного?**  
Итерация использует циклы, а рекурсия — повторные вызовы самой функции.

**4. В каких случаях рекурсивные алгоритмы могут быть неэффективны?**  
Когда слишком много вызовов — это занимает много памяти и может вызвать переполнение стека.

**5. Как можно оптимизировать рекурсивные функции?**  
Использовать мемоизацию, хвостовую рекурсию или заменить рекурсию циклом.