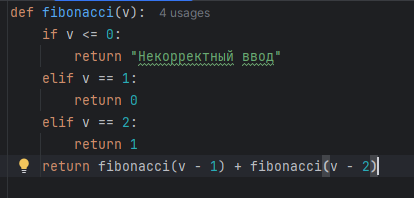


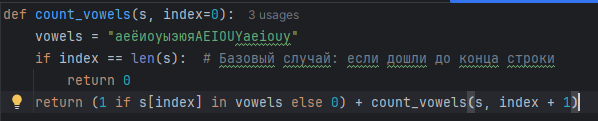
Создана функция factorial(z), которая принимает одно целое число z. Определена базовая ситуация: если z равно 0 или 1, функция возвращает 1. Для других значений z функция вызывает саму себя (factorial(z-1)) и умножает результат на z, реализуя принцип рекурсии.



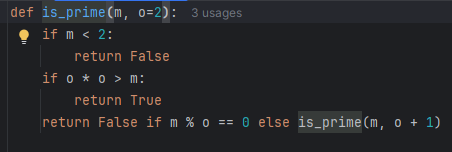
Создана функция fibonacci(n), которая принимает одно целое число n. Определены базовые случаи: Если n = 1, возвращается 0 (первый элемент ряда Фибоначчи).

Если n = 2, возвращается 1 (второй элемент ряда Фибоначчи).

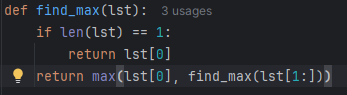
Для всех других значений n функция вызывает саму себя дважды: fibonacci(n - 1) + fibonacci(n - 2), вычисляя сумму двух предыдущих чисел.



Создана функция count\_vowels(s, index), принимающая строку s и индекс index, указывающий на текущий символ. Определён базовый случай: если индекс index достиг конца строки, функция возвращает 0. Если текущий символ s[index] является гласной буквой (из набора "аеёиоуыэюяAEIOUYaeiouy"), прибавляется 1, иначе 0. Функция вызывает себя с index + 1, двигаясь по строке.



Если n < 2, то число не является простым. Если d \* d > n, то число уже нельзя разделить на меньшие множители, значит, оно простое. Если n % d == 0, то число делится без остатка, следовательно, оно не является простым. В остальных случаях вызываем is\_prime(n, d + 1), увеличивая делитель d.



Базовый случай: если в списке один элемент, он и есть максимум. Рекурсивный случай: берём первый элемент и сравниваем его с максимумом оставшейся части списка (lst[1:]).