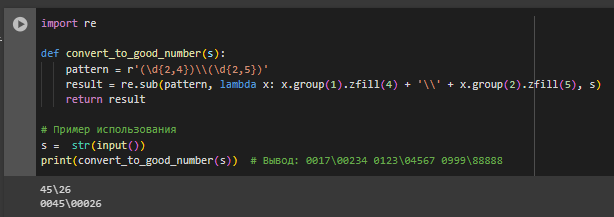
Задания Python

1. функция, которая принимает на вход строку и для каждого особенного номера, встречающегося в строке, выводит соответствующий хороший номер



import re

def convert\_to\_good\_number(s):

    pattern = r'(\d{2,4})\\(\d{2,5})'

    result = re.sub(pattern, lambda x: x.group(1).zfill(4) + '\\' + x.group(2).zfill(5), s)

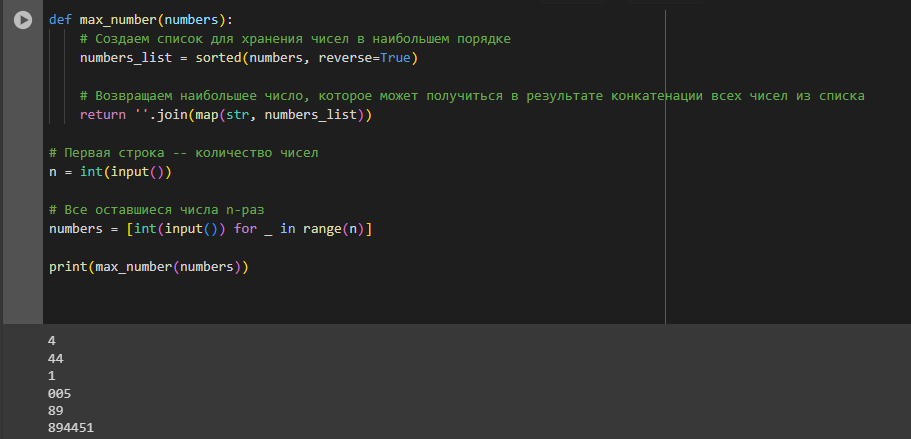
    return result

# Пример использования

s =  str(input())

print(convert\_to\_good\_number(s))

3. Функция, которая принимает на вход список строк, состоящих из цифр, и возвращает максимальное возможное число, которое может получиться в результате конкатенации всех строк из этого списка.

0

def max\_number(numbers):

    # Создаем список для хранения чисел в наибольшем порядке

    numbers\_list = sorted(numbers, reverse=True)

    # Возвращаем наибольшее число, которое может получиться в результате конкатенации всех чисел из списка

    return ''.join(map(str, numbers\_list))

# Первая строка -- количество чисел

n = int(input())

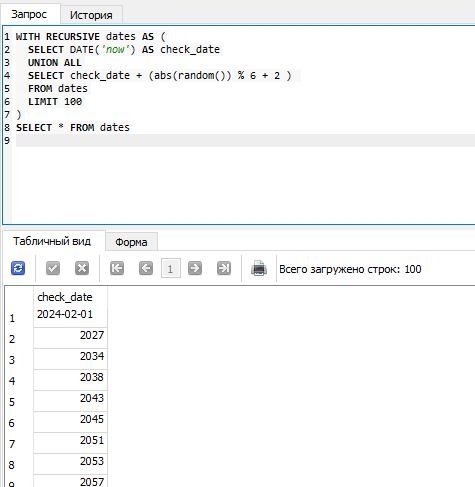
# Все оставшиеся числа n-раз

numbers = [int(input()) for \_ in range(n)]

print(max\_number(numbers))

Задания SQL

1. Оператор выбора, который выдаст 100 дат, начиная с текущей, при этом каждая дата отличается от предыдущей на 2-7 дней.



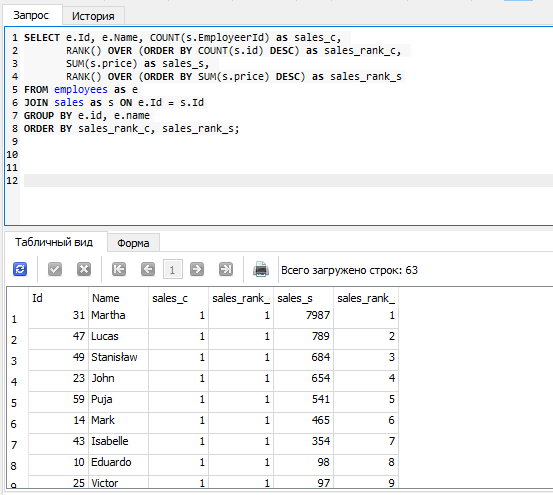
2. запрос, который вернёт количество и сумму продаж для каждого продавца, а также ранжирует продавцов по количеству продаж и по сумме продаж.

**sales\_c** - количество продаж,

**sales\_rank\_c** - ранг по количеству продаж,

**sales\_s** - сумма продаж,

**sales\_rank\_s** - ранг по сумме продаж.



3. Оператор выбора, который для каждого аккаунта выведет периоды постоянства остатков. Результат запроса должен содержать столбцы **acc** – номер аккаунта, **dt\_from** - начало периода,

**dt\_to** - конец периода, **balance** – остаток на счёте в данном периоде.

Дата конца последнего периода – 01.01.3000

