**МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Лабораторная работа 5**

Работа со строками в языке Python3

|  |
| --- |
| Выполнил студент группы ИВТ-б-о-20-1 |
| Симанский М.Ю « »\_\_\_\_\_\_\_\_\_20\_\_г. |
| Подпись студента\_\_\_\_\_\_\_\_\_\_\_ |
| Работа защищена « »\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. |
| Проверил Воронкин Р.А. \_\_\_\_\_\_\_\_  (подпись) |

**Цель работы:** приобретение навыков по работе со строками при написании программ с помощью языка программирования Python3.

Ссылка на репозиторий: https://github.com/MaksimSimanskiy/lab5.git

**Задание 1**

20. Дано предложение. Определить, сколько в нем одинаковых соседних букв.

**Код**

k = input("Введите предложение: ")

b=0

for i in range(len(k)-1):

if k[i] == k[i+1]: b += 1;

print('Одинаковых соседних букв = ',b)

**UML-диаграмма**

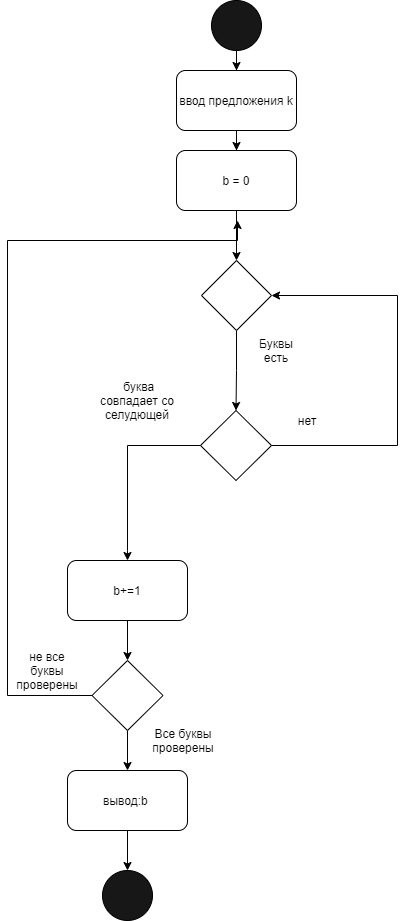


Рисунок 1 – UML-диаграмма

**Результат**

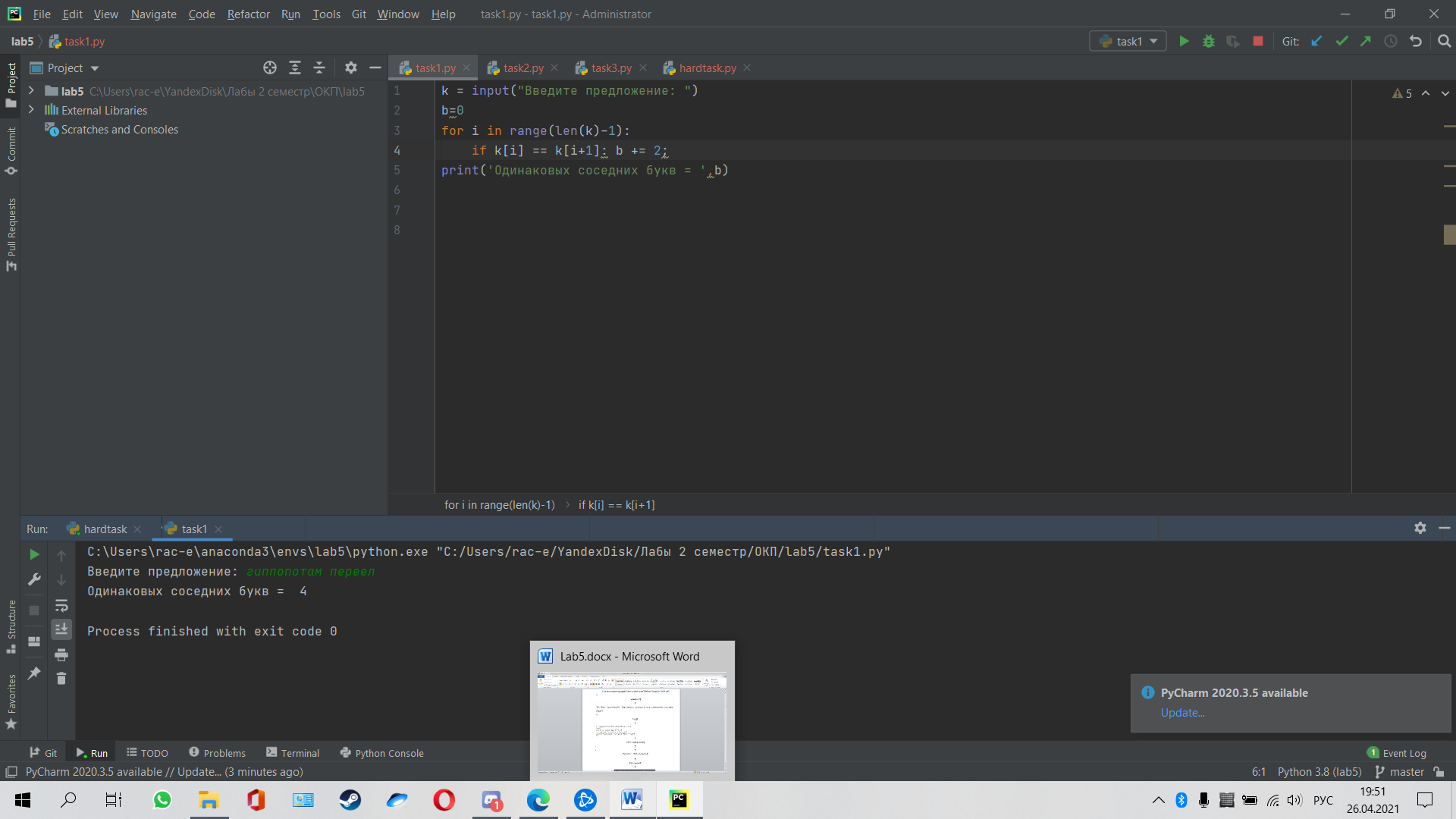


Рисунок 2 – Результат работы

**Задание 2**

12. Дано предложение. Напечатать все символы, расположенные между первой и второй запятыми. Если второй запятой нет, то должны быть напечатаны все символы, расположенные после единственной имеющейся запятой.

**Код**

k = input("Введите предложение: ")

z = k.find(',')

if( z == -1):

print('Нет запятых')

exit(1)

f = k.find(',', z+1)

if(f != -1):

print(k[z+1:f])

elif(f == -1):

print(k[z+1:])

**UML-диаграмма**

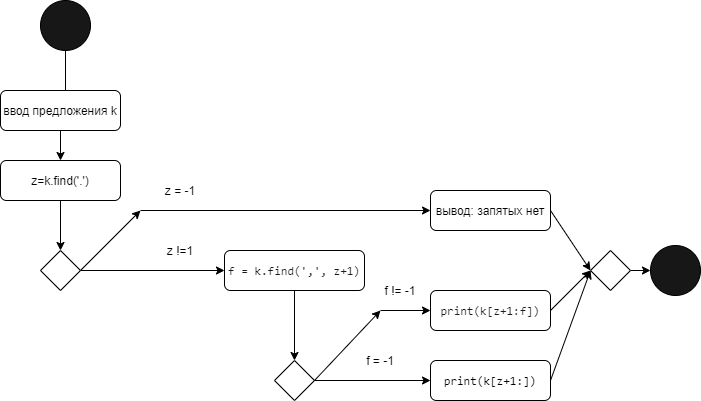
****

Рисунок 3 – UML-диаграмма

**Результат**

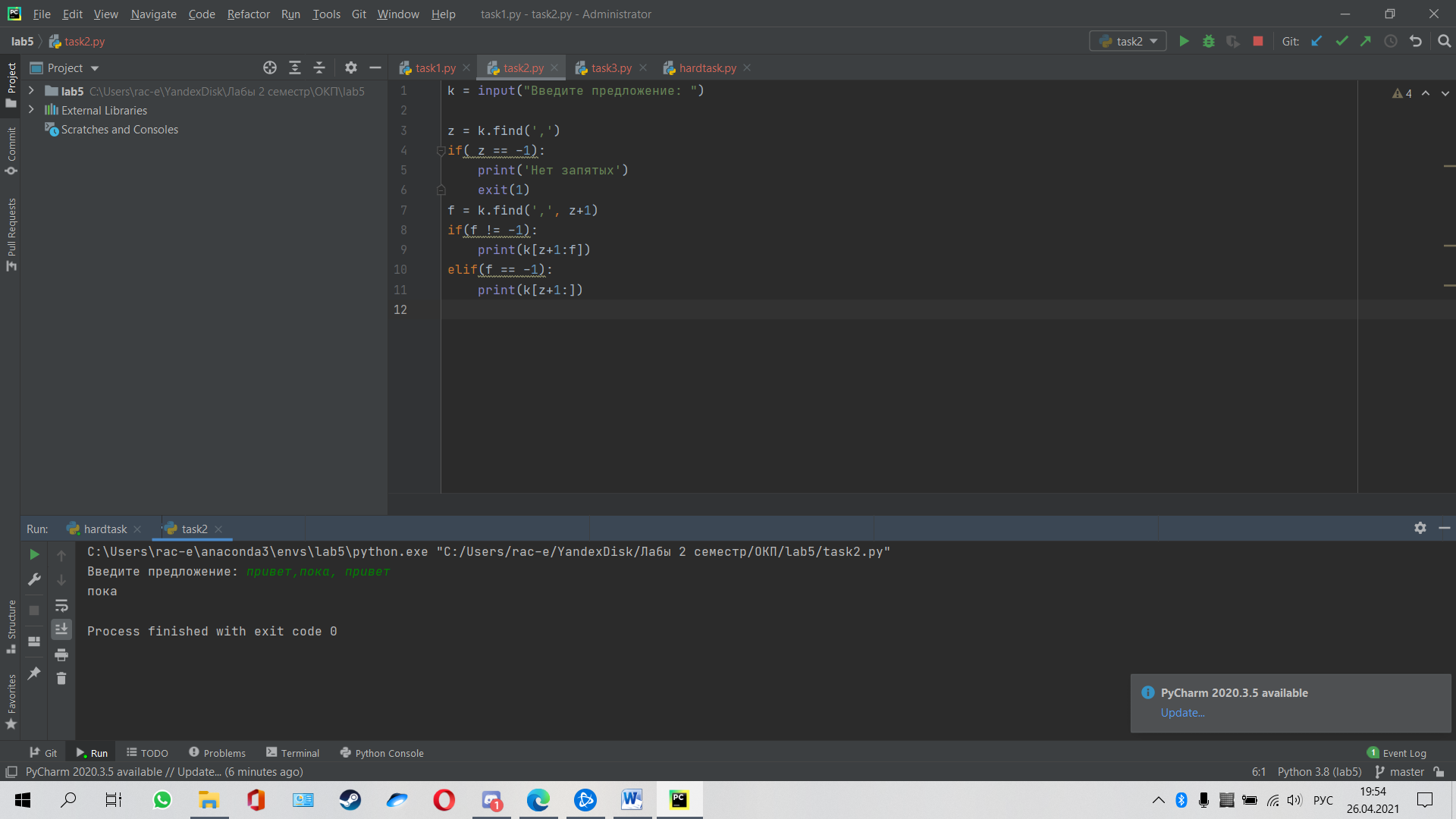


Рисунок 4 – Результат работы

**Задание 3**

26. Дана строка, состоящая только из букв. Заменить все буквы «a» на буквы «б» и наоборот, как заглавные, так и строчные. Например, при вводе строки «абвАБВ» должен получиться результат «бавБАВ».

**Код**

k = input("Введите слово: ")

a = list(k)

x = '';

s = []

for i in range(len(a)):

if(a[i] == 'а' and a[i].isupper() == False):a[i] = "б"

elif (a[i] == 'б' and a[i].isupper() == False): a[i] = "а"

if (a[i] == 'А' and a[i].isupper() == True): a[i] = "Б"

elif (a[i] == 'Б'and a[i].isupper() == True): a[i] = "А"

x = x.join(a)

print("Результат - ", x)

**UML-диаграмма**

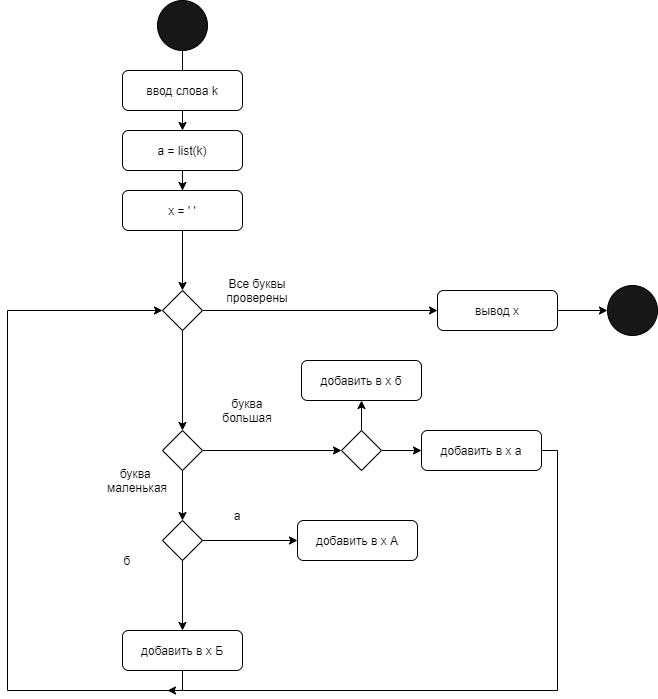


Рисунок 5 – UML- диаграмма

**Результат**

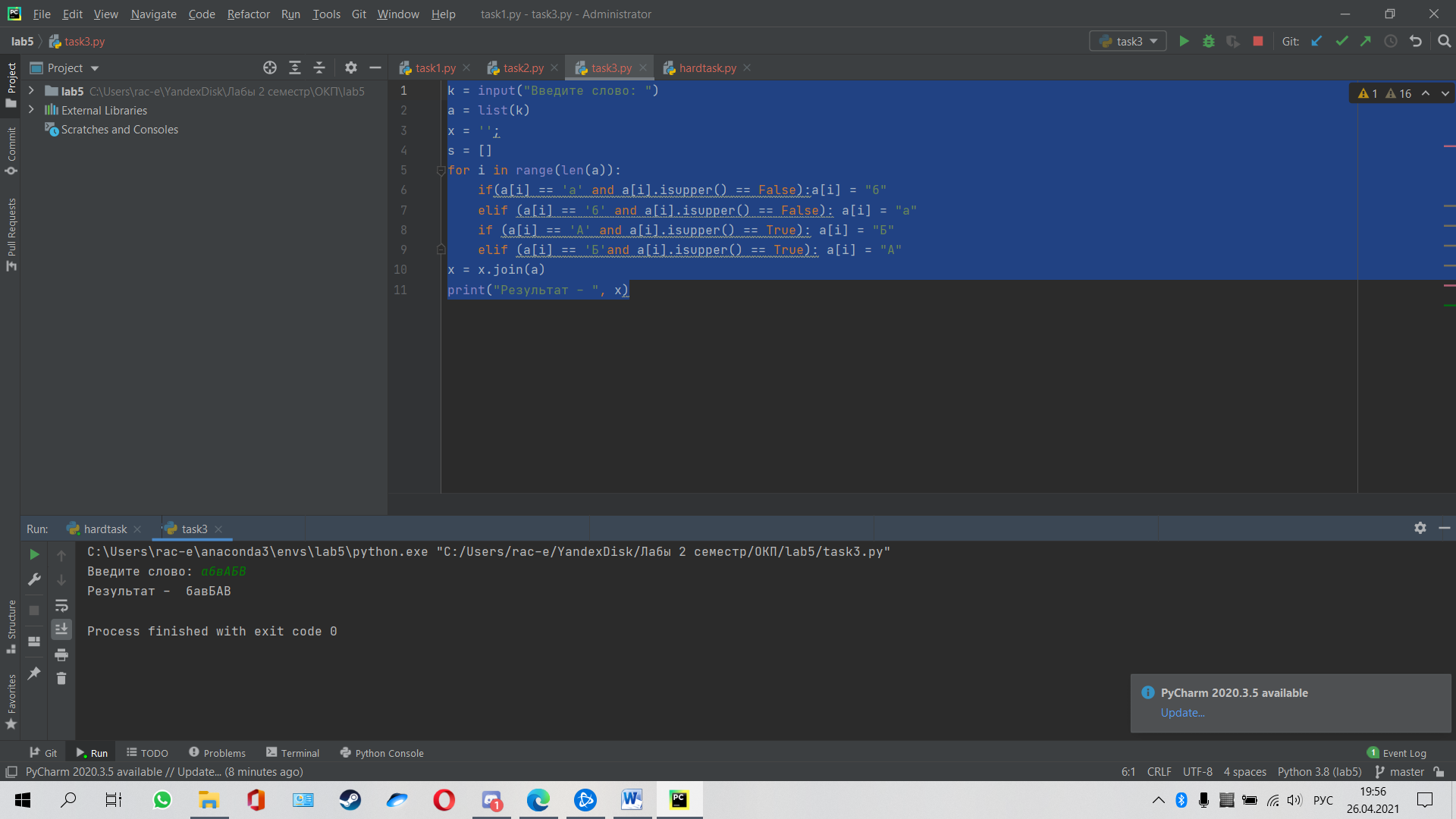


Рисунок 6 – Результат работы

**Задание повышенной сложности**

18. Дано предложение. Найти какое-нибудь его слово, начинающееся на букву к.

**Код**

s = input("Введите предложение: ")

g = s.split(' ')

for i in range(len(g)):

if( g[i].find('к') == 0):

print(g[i])

break

**UML-диаграмма**

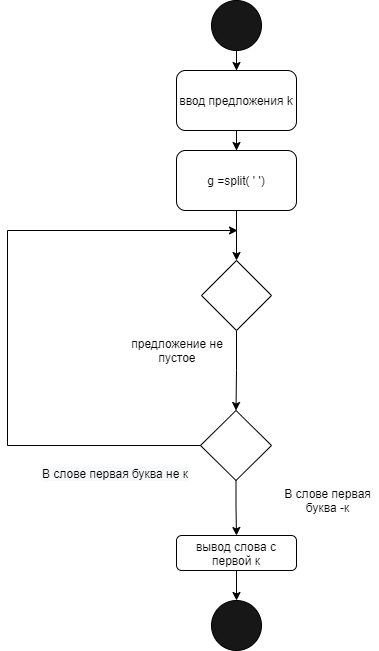
****

Рисунок 7 – UML-диаграмма

**Результат**

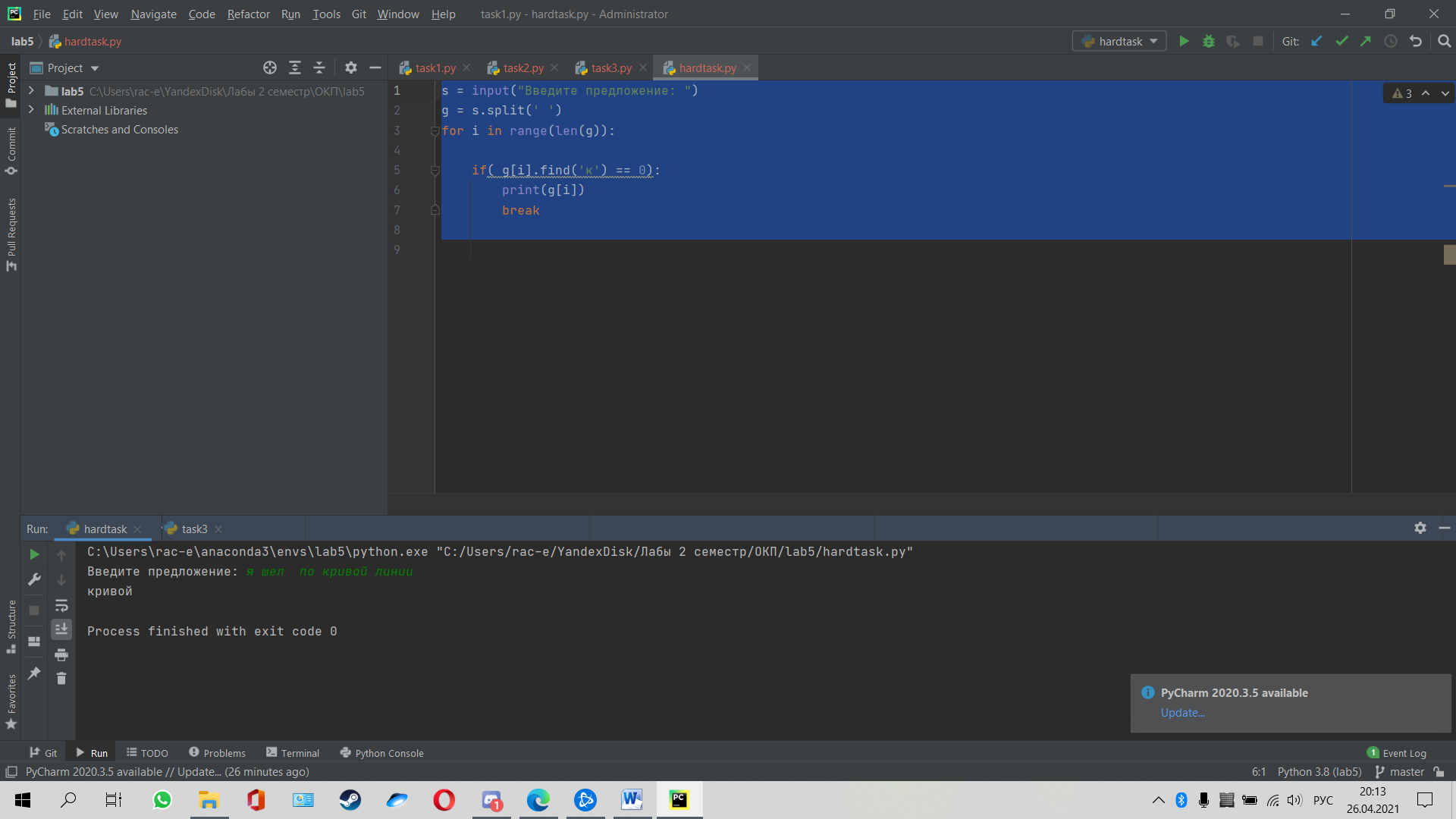


Рисунок 8 – Результат работы

**Ответы на контрольные вопросы**

1. Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации.
2. S = 'spam"s' S = "spam's"
3. Оператор сложения строк + Оператор умножения строк \*Оператор принадлежности подстроки in
4. В Python строки являются упорядоченными последовательностями символьных данных и могут быть проиндексированы. Доступ к отдельным символам в строке можно получить, указав имя строки, за которым следует число в квадратных скобках [] .
5. Python также допускает возможность извлечения подстроки из строки, известную как ‘‘string slice’’. Если s это строка, выражение формы s[m:n] возвращает часть s , начинающуюся с позиции m , и до позиции n ,
6. Строка указывает на ячейку памяти – если строка меняется, то и ячейка памяти должна быть другой.
7. if i == i.upper():
8. Оператор in возвращает True , если подстрока входит в строку, и False , если нет
9. Str.find()
10. len()
11. метод count()
12. В Python версии 3.6 был представлен новый способ форматирования строк. Эта функция официально названа литералом отформатированной строки, но обычно упоминается как f- строки (f-string).
13. Метод find
14. Использовать { имя\_переменной}
15. string.isdigit ()
16. String.split( символ)
17. islower()
18. String[0].isupper()
19. В некоторых языках это возможно, но Python при попытке выполнения подобной операции будет выдана ошибка
20. Для того чтобы «перевернуть» строку, её можно разбить, представив в виде списка символов, «перевернуть» список, и, объединив его элементы, сформировать новую строку.

string.join(reversed("hello world")).

1. Метод join() умеет объединять элементы списков в строки, разделяя отдельные строки с использованием заданного символа
2. Mожно воспользоваться методами upper() и lower(), которые, соответственно, приводят все символы строк к верхнему и нижнему регистрам.
3. animal = 'fish' animal[0].upper() + animal[1:-1] + animal[-1].upper() *#=> 'FisH'*
4. isupper() возвращает True только в том случае, если вся строка состоит из прописных букв.
5. Метод splitlines() разделяет строки по символам разрыва строки \n
6. replace().
7. startswith() и endswith()
8. isspace()
9. Будет создана новая строка, представляющая собой исходную строку, повторённую три раза.
10. title()
11. Метод partition() разбивает строку по заданной подстроке. После этого результат возвращается в виде кортежа.
12. Метод rfind() похож на метод find(), но он, в отличие от find(), просматривает строку не слева направо, а справа налево, возвращая индекс первого найденного вхождения искомой подстроки.