

**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 №6 по дисциплине основы программной
инженерии**

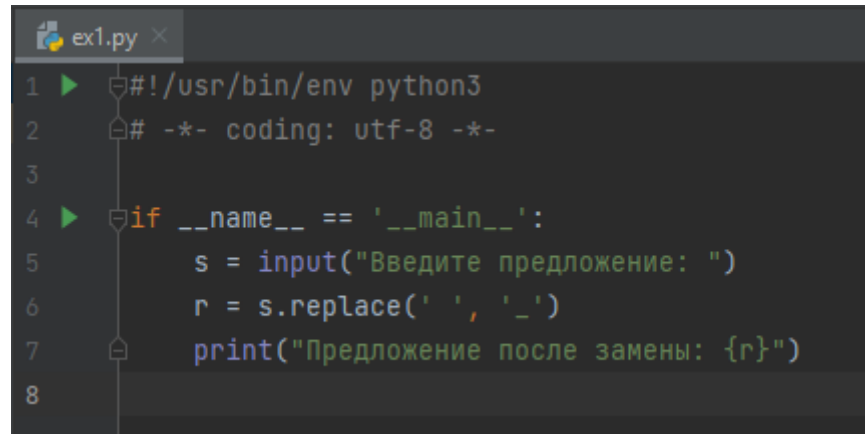
Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

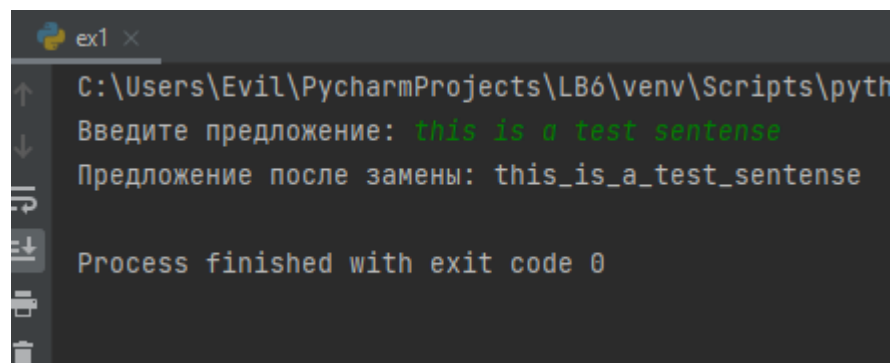
1. Работа со строками в языке Python

1.1 Пример 1 (рис. 1, 2).



```
ex1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      s = input("Введите предложение: ")
6      r = s.replace(' ', '_')
7      print("Предложение после замены: {r}")
8
```

Рисунок 1 – Код примера

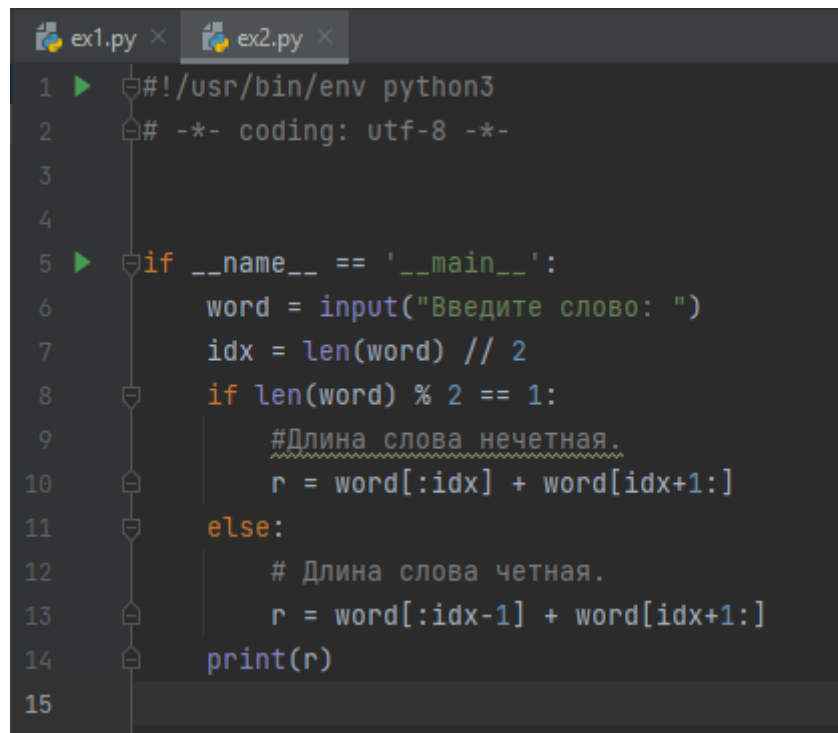


```
ex1 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe
Введите предложение: this is a test sentence
Предложение после замены: this_is_a_test_sentence

Process finished with exit code 0
```

Рисунок 2 – Работа программы

1.2 Пример 2 (рис. 3, 4, 5).

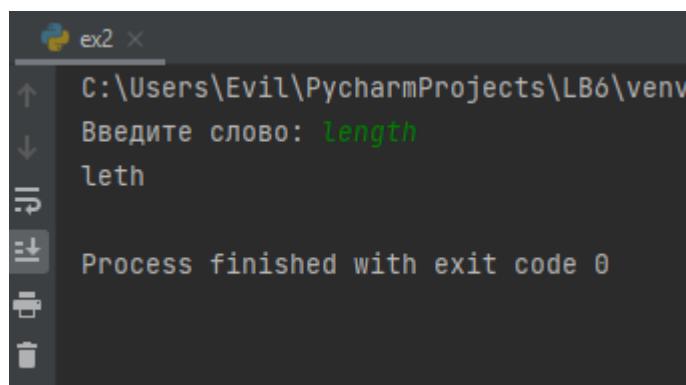


```

1  > #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  > if __name__ == '__main__':
6      word = input("Введите слово: ")
7      idx = len(word) // 2
8      if len(word) % 2 == 1:
9          #Длина слова нечетная.
10         r = word[:idx] + word[idx+1:]
11     else:
12         # Длина слова четная.
13         r = word[:idx-1] + word[idx+1:]
14     print(r)
15

```

Рисунок 3 – Код программы

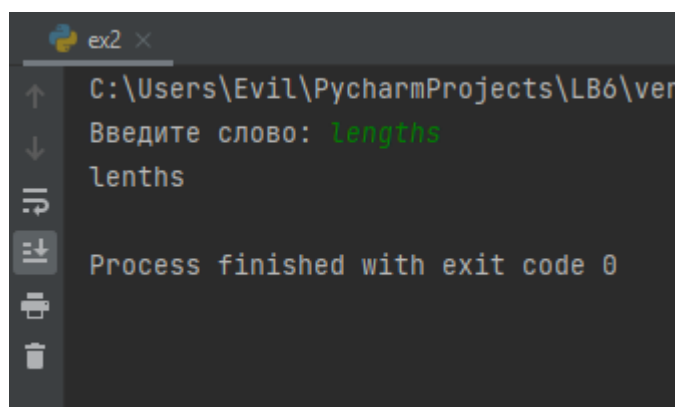


```

ex2 x
C:\Users\Evil\PycharmProjects\LB6\venv\
Введите слово: length
leth
Process finished with exit code 0

```

Рисунок 4 – Вывод программы при чётной длине слова



```

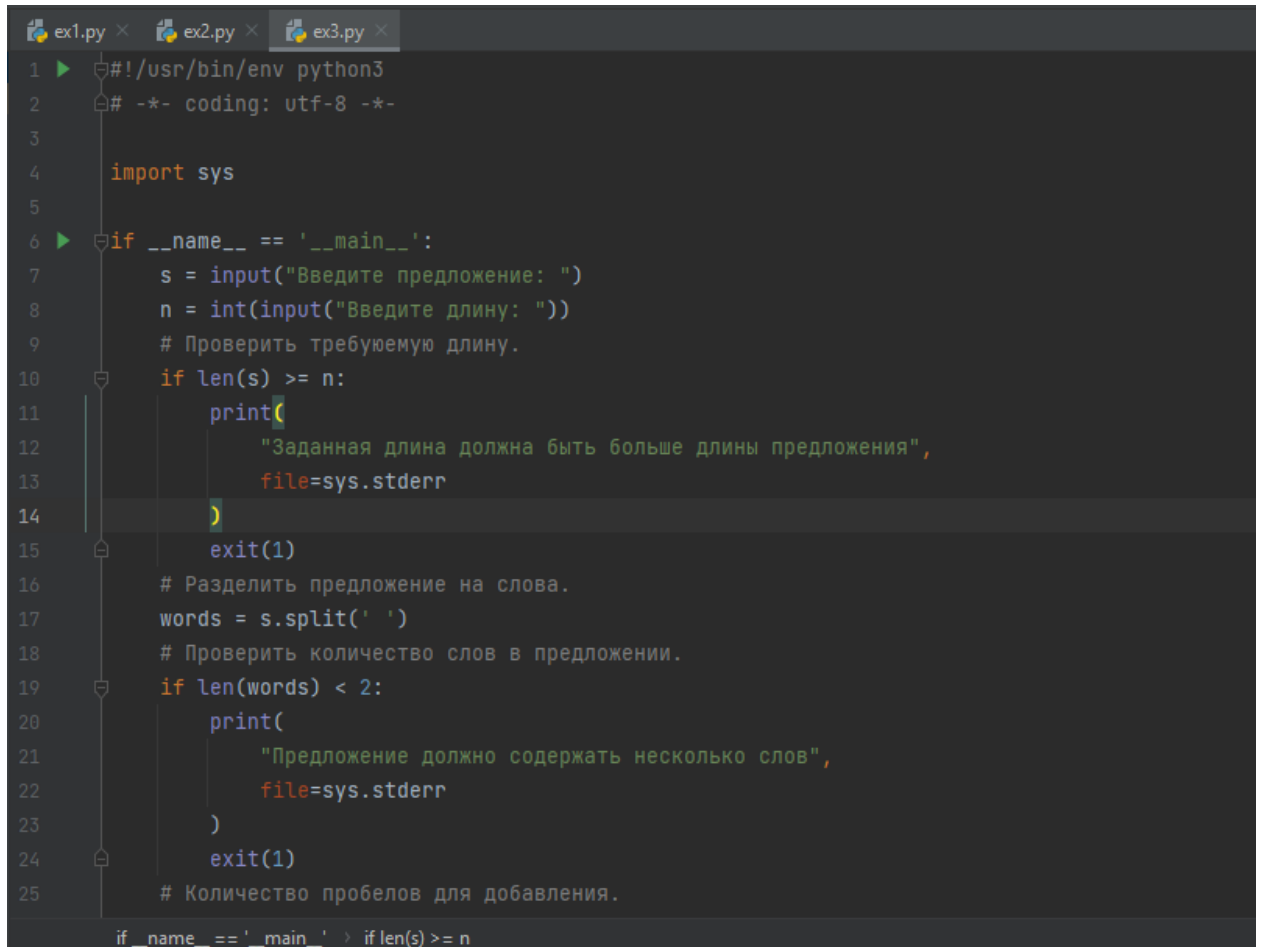
ex2 x
C:\Users\Evil\PycharmProjects\LB6\venv\
Введите слово: lengths
lenths
Process finished with exit code 0

```

Рисунок 5 – Вывод программы при нечётной длине слова

1.3 Пример 3 (рис. 6, 7, 8).

Код программы представлен на рисунках 6 и 7.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      s = input("Введите предложение: ")
8      n = int(input("Введите длину: "))
9      # Проверить требуемую длину.
10     if len(s) >= n:
11         print(
12             "Заданная длина должна быть больше длины предложения",
13             file=sys.stderr
14         )
15         exit(1)
16     # Разделить предложение на слова.
17     words = s.split(' ')
18     # Проверить количество слов в предложении.
19     if len(words) < 2:
20         print(
21             "Предложение должно содержать несколько слов",
22             file=sys.stderr
23         )
24         exit(1)
25     # Количество пробелов для добавления.
```

Рисунок 6 – Код программы

```
ex1.py x ex2.py x ex3.py x
25 # Количество пробелов для добавления.
26 delta = n
27 for word in words:
28     delta -= len(word)
29 # Количество пробелов на каждое слово.
30 w, r = delta // (len(words) - 1), delta % (len(words) - 1)
31 # Сформировать список для хранения слов и пробелов.
32 lst = []
33 # Пронумеровать все слова в списке и перебрать их.
34 for i, word in enumerate(words):
35     lst.append(word)
36 # Если слово не является последним, добавить пробелы.
37 if i < len(words) - 1:
38     # Определить количество пробелов.
39     width = w
40     if r > 0:
41         width += 1
42         r -= 1
43     # Добавить заданное количество пробелов в список.
44     if width > 0:
45         lst.append(' ' * width)
46
47 # Вывести новое предложение, объединив все элементы списка lst.
48 print(''.join(lst))
49
if __name__ == '__main__': > if len(s) >= n
```

Рисунок 7 – Продолжение

```
Run: ex3 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe
Введите предложение: this is a test sentence
Введите длину: 28
this is a test sentence

Process finished with exit code 0
```

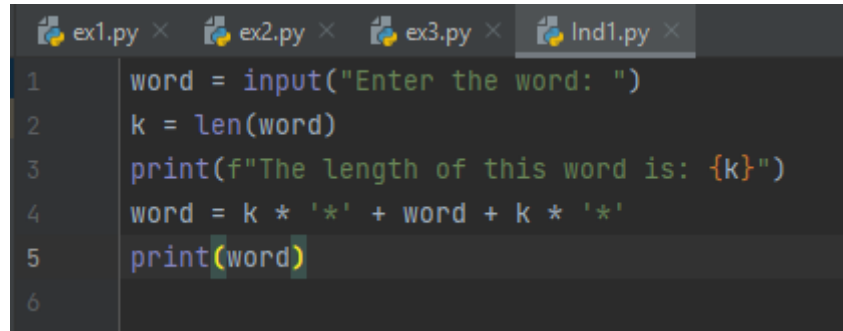
Рисунок 8 – Пример работы программы

Решение индивидуальных заданий

Вариант 5

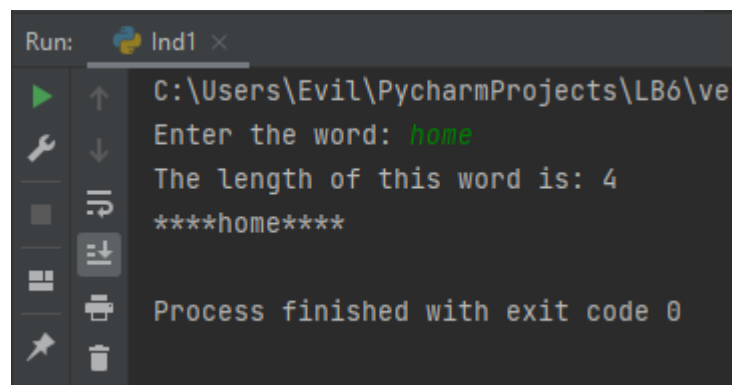
1.4 Индивидуальное задание №1 (рис. 9, 10, 11).

Условие: Дано слово. Добавить к нему в начале и конце столько звездочек, сколько букв в этом слове.



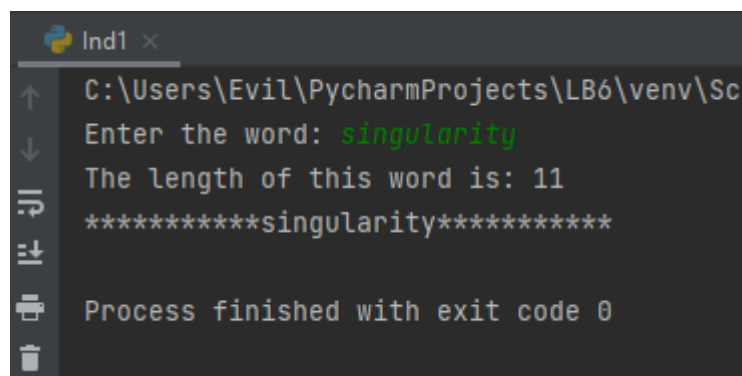
```
1 word = input("Enter the word: ")
2 k = len(word)
3 print(f"The length of this word is: {k}")
4 word = k * '*' + word + k * '*'
5 print(word)
6
```

Рисунок 9 – Код программы



```
Run: Ind1 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe
Enter the word: home
The length of this word is: 4
****home****
Process finished with exit code 0
```

Рисунок 10 – Пример работы программы для слова с длиной 4



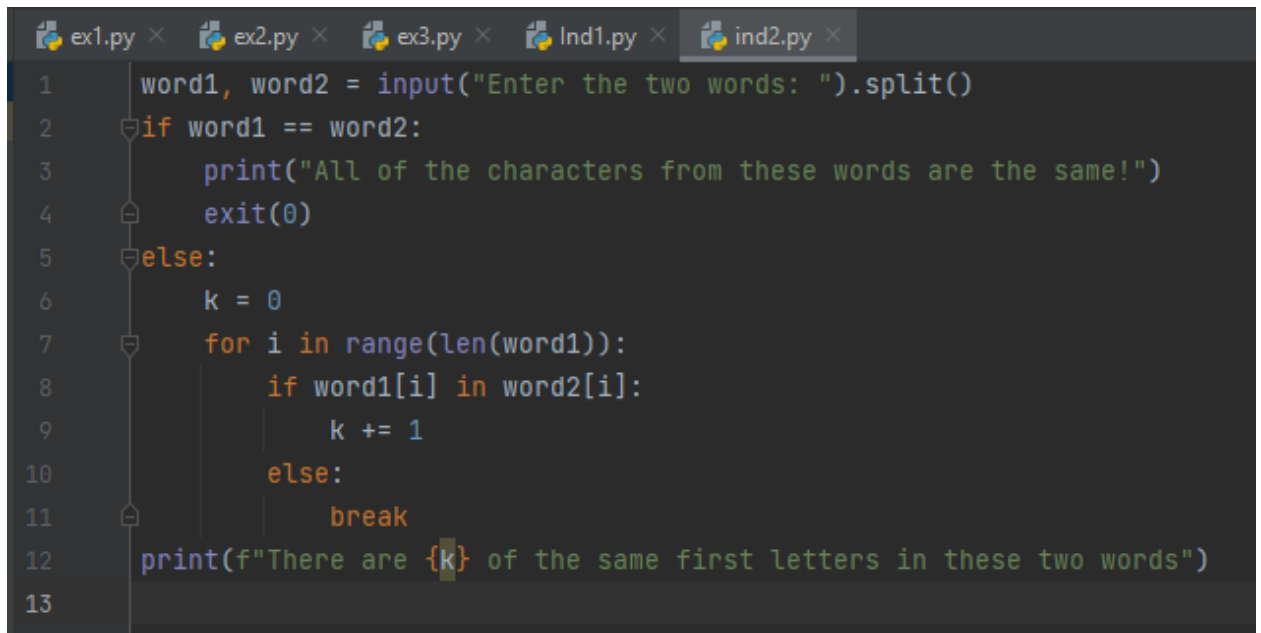
```
Ind1 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe
Enter the word: singularity
The length of this word is: 11
*****singularity*****
Process finished with exit code 0
```

Рисунок 11 – Пример работы программы для слова с длиной 11

1.5 Индивидуальное задание №2 (рис. 12, 13, 14, 15).

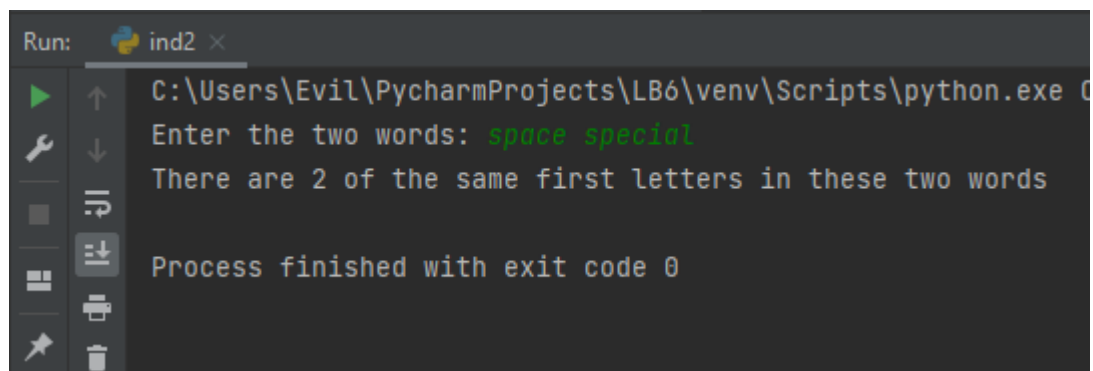
Условие: Даны два слова. Определить, сколько начальных букв первого слова совпадает с начальными буквами второго слова. Рассмотреть два случая:

- известно, что слова разные;
- слова могут быть одинаковыми.



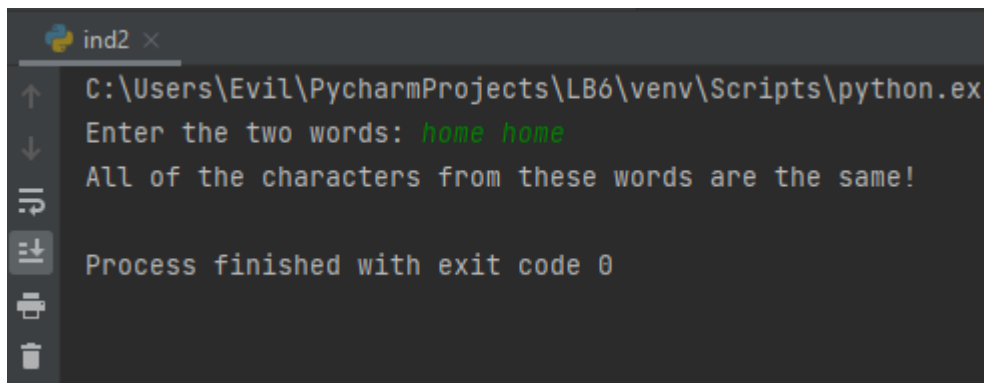
```
1 word1, word2 = input("Enter the two words: ").split()
2 if word1 == word2:
3     print("All of the characters from these words are the same!")
4     exit(0)
5 else:
6     k = 0
7     for i in range(len(word1)):
8         if word1[i] in word2[i]:
9             k += 1
10        else:
11            break
12    print(f"There are {k} of the same first letters in these two words")
13
```

Рисунок 12 – Код программы



```
Run: ind2 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe 0
Enter the two words: space special
There are 2 of the same first letters in these two words
Process finished with exit code 0
```

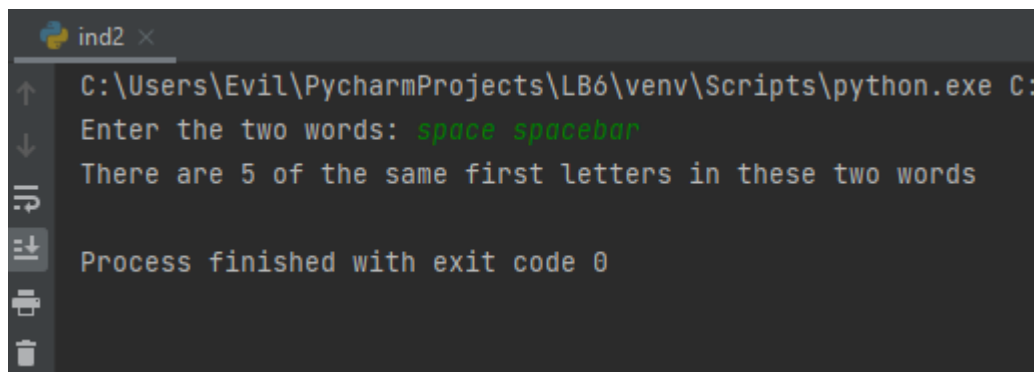
Рисунок 13 – Пример работы программы при двух совпадающих первых буквах



```
ind2 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe
Enter the two words: home home
All of the characters from these words are the same!

Process finished with exit code 0
```

Рисунок 14 – Пример работы программы для одинаковых слов



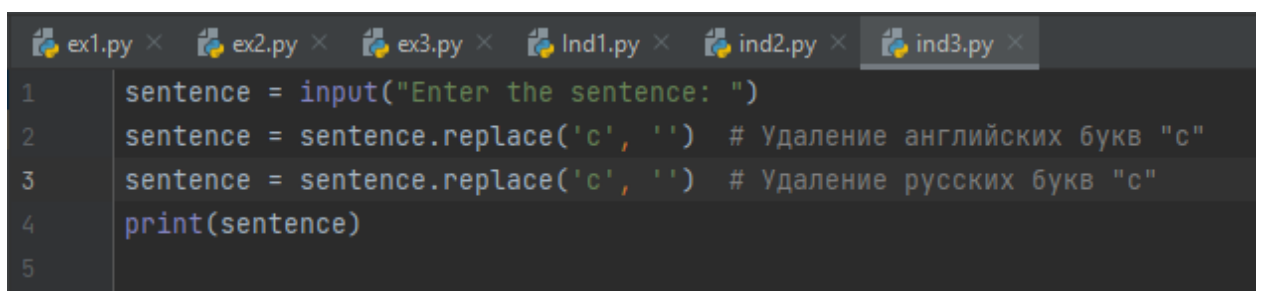
```
ind2 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe C:
Enter the two words: space spacebar
There are 5 of the same first letters in these two words

Process finished with exit code 0
```

Рисунок 15 – Еще пример работы программы

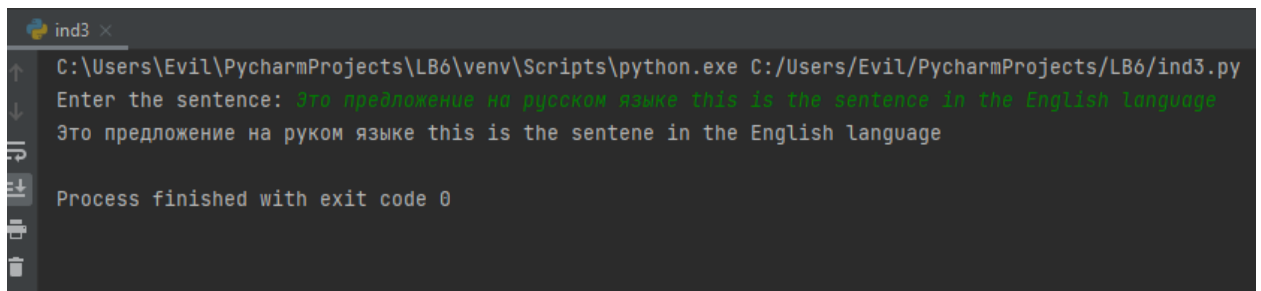
1.6 Индивидуальное задание №3 (рис. 16)

Условие: Дано предложение. Удалить из него все буквы с (как в кириллице, так и на латинице).



```
ex1.py x  ex2.py x  ex3.py x  ind1.py x  ind2.py x  ind3.py x
1  sentence = input("Enter the sentence: ")
2  sentence = sentence.replace('c', '') # Удаление английских букв "с"
3  sentence = sentence.replace('с', '') # Удаление русских букв "с"
4  print(sentence)
5
```

Рисунок 16 – Код программы



```
ind3 x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe C:/Users/Evil/PycharmProjects/LB6/ind3.py
Enter the sentence: Это предложение на русском языке this is the sentence in the English language
Это предложение на русском языке this is the sentence in the English language
Process finished with exit code 0
```

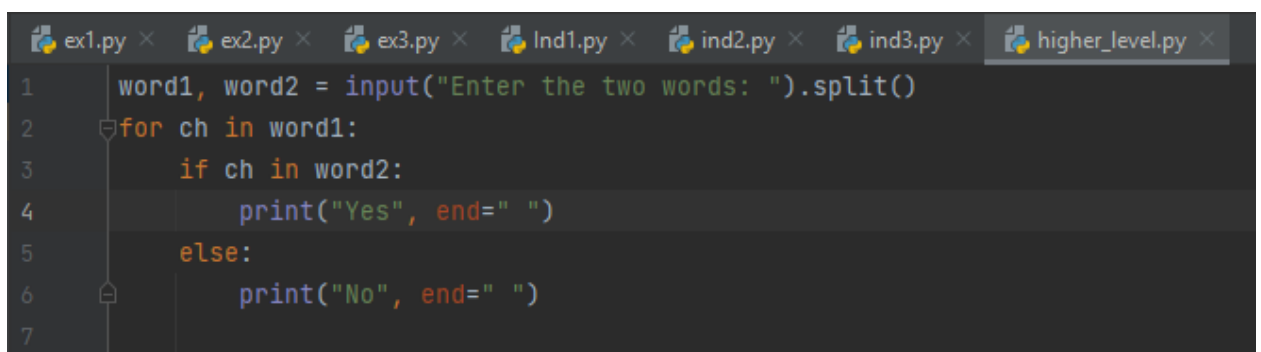
Рисунок 17 – Работа программы

Задание повышенной сложности

Вариант 5

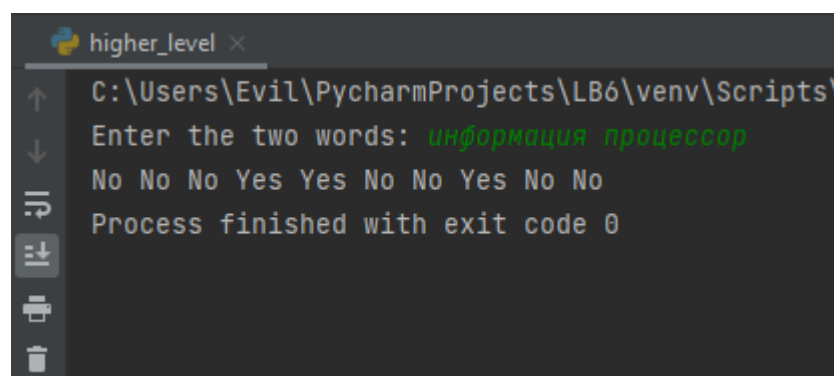
Условие: Даны два слова. Для каждой буквы первого слова (в том числе для повторяющихся в этом слове букв) определить, входит ли она во второе слово. Например, если заданные слова «информация» и «процессор», то для букв первого из них ответом должно быть: «нет нет нет да

да нет нет да нет нет».



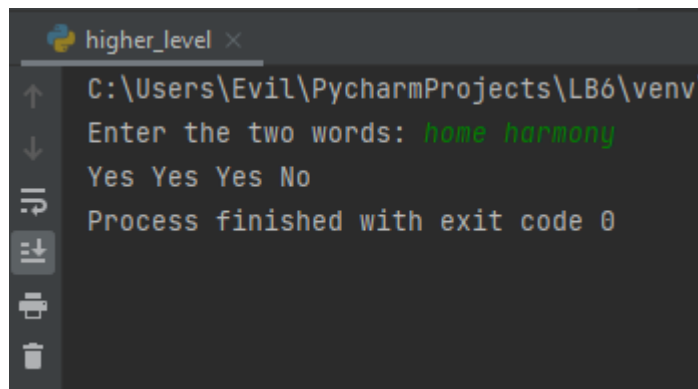
```
ex1.py x ex2.py x ex3.py x ind1.py x ind2.py x ind3.py x higher_level.py x
1 word1, word2 = input("Enter the two words: ").split()
2 for ch in word1:
3     if ch in word2:
4         print("Yes", end=" ")
5     else:
6         print("No", end=" ")
7
```

Рисунок 18 – Код программы



```
higher_level x
C:\Users\Evil\PycharmProjects\LB6\venv\Scripts\python.exe C:/Users/Evil/PycharmProjects/LB6/higher_level.py
Enter the two words: информация процессор
No No No Yes Yes No No Yes No No
Process finished with exit code 0
```

Рисунок 19 – Пример работы программы



```
higher_level x
C:\Users\Evil\PycharmProjects\LB6\venv\
Enter the two words: home harmony
Yes Yes Yes No
Process finished with exit code 0
```

Рисунок 20 – Пример работы программы

2. Ответы на контрольные вопросы

1. Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

2. Строки в апострофах и в кавычках, экранированные последовательности - служебные символы, "Сырые" строки, строки в тройных апострофах или кавычках.

3. Сложение, умножение, оператор принадлежности. Строковых функций в Python много, вот некоторые из них:

`chr()` – Преобразует целое число в символ

`ord()` – Преобразует символ в целое число

`len()` – Возвращает длину строки

`str()` – Изменяет тип объекта на string

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 []. Индексация строк начинается с нуля: у первого символа индекс 0, следующего 1 и так далее. Индекс последнего символа в python — “длина строки минус один”.

5. Если `s` это строка, выражение формы `s[m:n]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая позицию. Если пропустить первый индекс, срез начинается с начала строки. Аналогично, если опустить второй индекс `s[n:]`, срез длится от первого индекса до конца строки.

6. Более легкое представление в памяти.
7. `s.istitle()`
8. `if s1 in s2`
9. `s.find(<sub>)`.
10. `len(s)`
11. `s.count(<char>)`.
12. f-строки упрощают форматирование строк. Пример: `print(f' This is {name}, he is {age} years old')`
13. `string.find(<sub>[, <start>[, <end>]])`
14. `'Hello, { }!'.format('Vasya')`
15. `string.isdigit()`
16. `'foo.bar.baz.qux'.rsplit(sep='.')` – пример разделения
17. `string.islower()`
18. `s[0].isupper()`
19. С точки зрения математической операции нельзя, можно лишь только вывести из без разделения друг от друга
20. `s[::-1]` – при помощи среза.
21. `‘–’.join(<iterable>)`
22. К верхнему – `string.upper()`, к нижнему – `string.lower()`.
23. `s[0].upper()` `s[len(s) – 1].upper()`
24. `s.isupper()`
25. Если нужно сохранить символы, обозначающие конец слов.
26. `s.replace(‘что заменить’, ‘на что заменить’)`
27. `string.endswith(<suffix>[, <start>[, <end>]])`, `str.startswith(prefix[, start[, end]])`
28. `s.isspace()`
29. Будет получена копия исходной строки в трёхкратном размере.
30. `s.tittle()`
31. `s.partition(<sep>)` отделяет от `s` подстроку длиной от начала до первого вхождения `<sep>` .

Возвращаемое значение представляет собой кортеж из трех частей:

Часть s до <sep>

Разделитель <sep>

Часть s после <sep>

32. Когда нужен индекс последнего вхождения подстроки в строку.