

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины «Основы программной инженерии»

Выполнил:
Матвеев Александр Иванович
1 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Работа со строками в языке Python

Цель работы: приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Ход работы.

1. Создание нового репозитория с лицензией MIT.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * SashkaHacker / **Repository name *** laba3
✔ laba3 is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-happiness](#) ?

Description (optional)
Выполнение лабораторной работы №3 по дисциплине: основы программной инженерии

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: Python
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

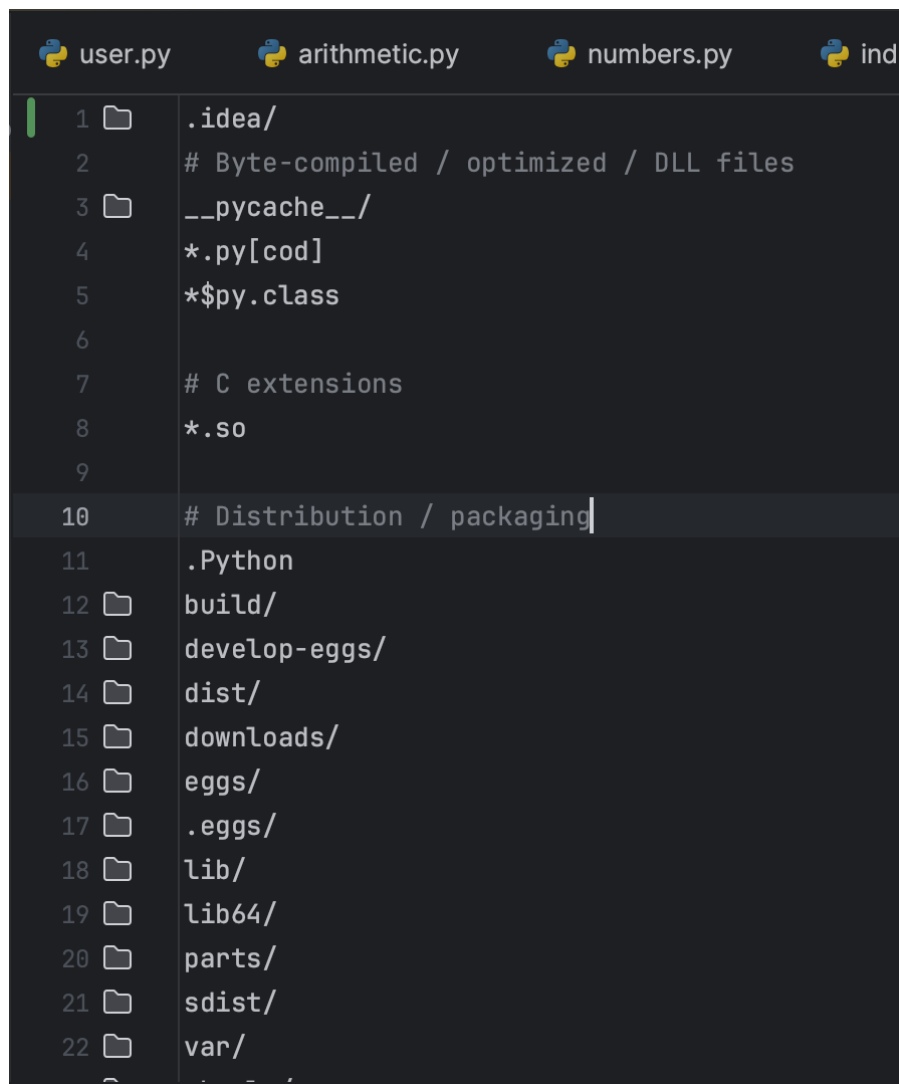
Рисунок 1 – Создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
Sashka@DESKTOP-U4RPSBI MINGW64 ~/Documents/GitHub/lab6 (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/SashkaHacker/lab6/pull/new/develop
remote:
To https://github.com/SashkaHacker/lab6.git
 * [new branch]      develop -> develop
```

Рисунок 2 – Клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.



The screenshot shows a code editor with a dark theme. At the top, there are four tabs: 'user.py', 'arithmetic.py', 'numbers.py', and 'ind'. The active tab is 'user.py'. The editor displays the content of the '.gitignore' file, which is as follows:

```
1  .idea/
2  # Byte-compiled / optimized / DLL files
3  __pycache__/
4  *.py[cod]
5  *$py.class
6
7  # C extensions
8  *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
```

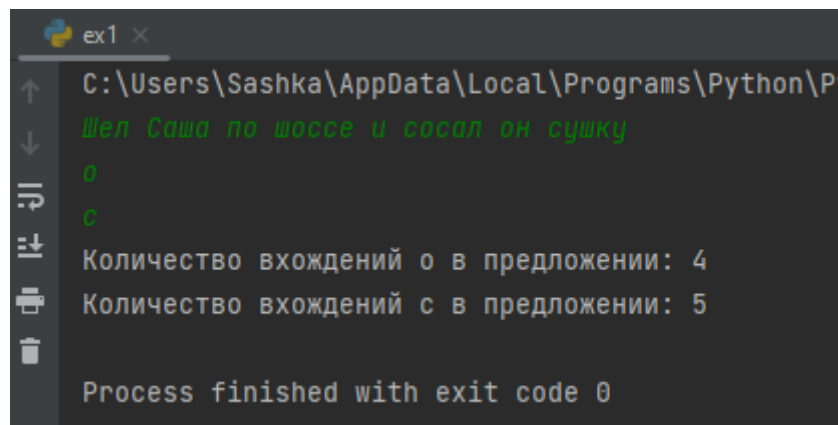
Рисунок 3 – Файл .gitignore

4. Выполнение задания №1.

Условие: Дано предложение. Составить программу, которая выводит все вхождения в предложение двух заданных символов

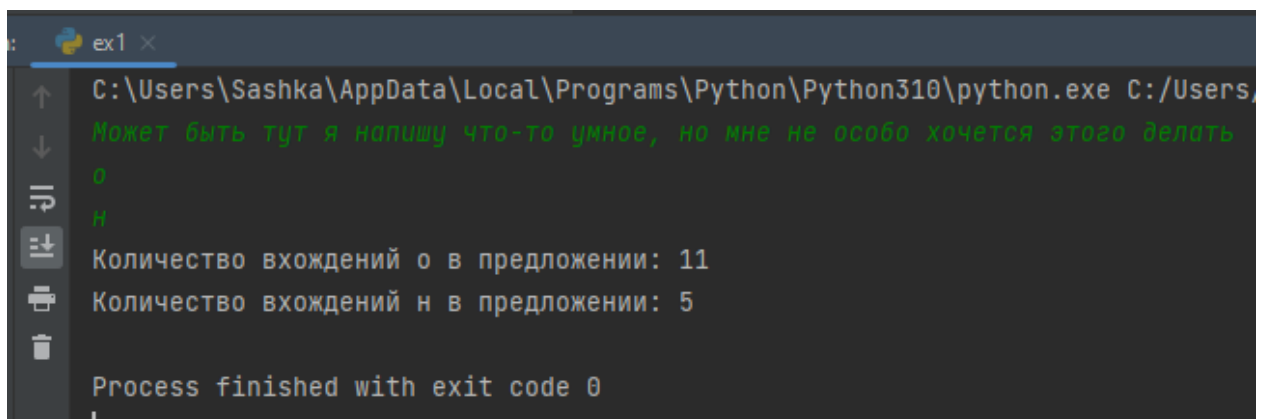
```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      # Вариант 11
5
6      ▶  if __name__ == "__main__":
7          propose = input()
8          a, b = input(), input()
9          print(f"Количество вхождений {a} в предложении: {propose.count(a)}")
10         print(f"Количество вхождений {b} в предложении: {propose.count(b)}")
11
```

Рисунок 4 – Код программы



```
ex1 x
C:\Users\Sashka\AppData\Local\Programs\Python\Py
Шел Саша по шоссе и сосал он сушку
о
с
Количество вхождений о в предложении: 4
Количество вхождений с в предложении: 5
Process finished with exit code 0
```

Рисунок 5 – Пример выполнения программы

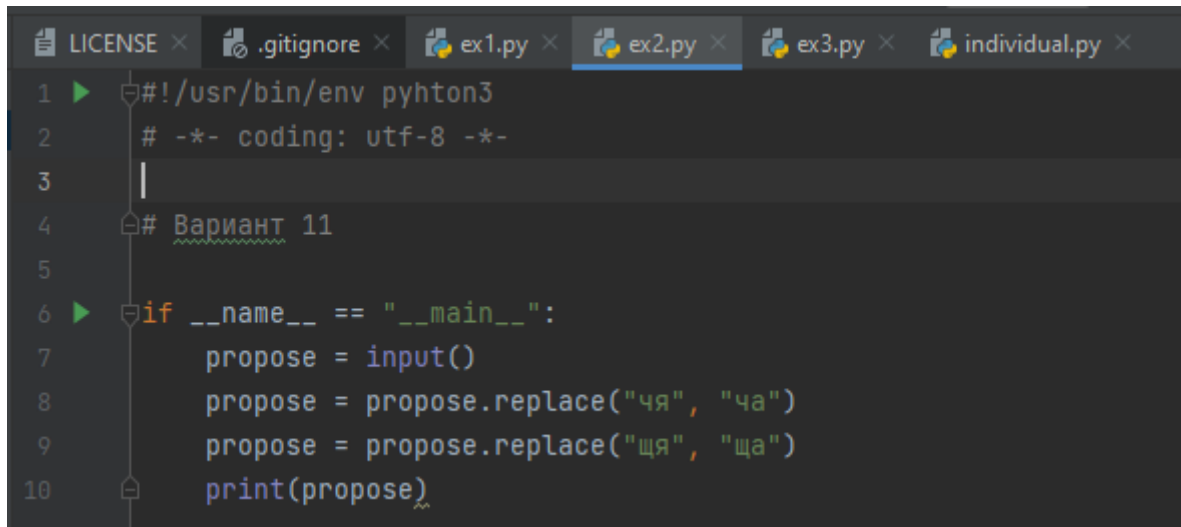


```
ex1 x
C:\Users\Sashka\AppData\Local\Programs\Python\Python310\python.exe C:/Users,
Может быть тут я напишу что-то умное, но мне не особо хочется этого делать
о
н
Количество вхождений о в предложении: 11
Количество вхождений н в предложении: 5
Process finished with exit code 0
```

Рисунок 6 – Пример выполнения программы

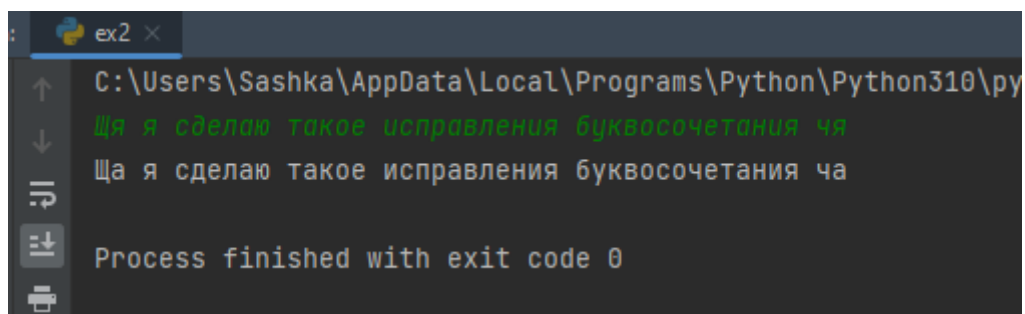
5. Выполнение задания №2.

Условие: Дана последовательность слов. Проверить, правильно ли в ней записаны буквосочетания ча и ща. Исправить ошибки.



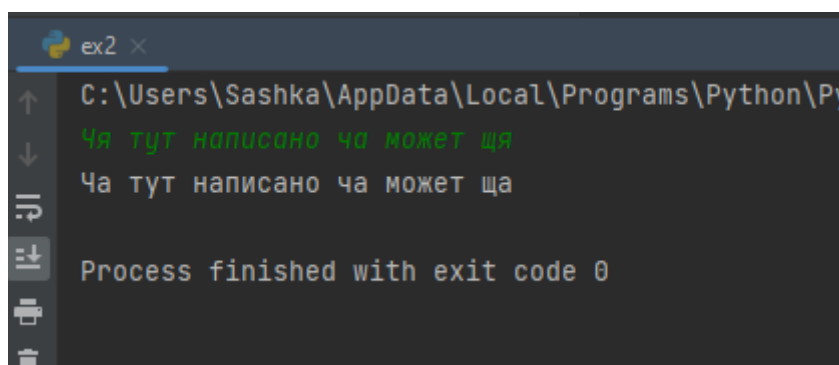
```
LICENSE x .gitignore x ex1.py x ex2.py x ex3.py x individual.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Вариант 11
5
6  if __name__ == "__main__":
7      propose = input()
8      propose = propose.replace("чя", "ча")
9      propose = propose.replace("щя", "ща")
10     print(propose)
```

Рисунок 7 – Код программы



```
ex2 x
C:\Users\Sashka\AppData\Local\Programs\Python\Python310\py
Щя я сделаю такое исправления буквосочетания чя
Ща я сделаю такое исправления буквосочетания ча
Process finished with exit code 0
```

Рисунок 8 – Пример выполнения программы

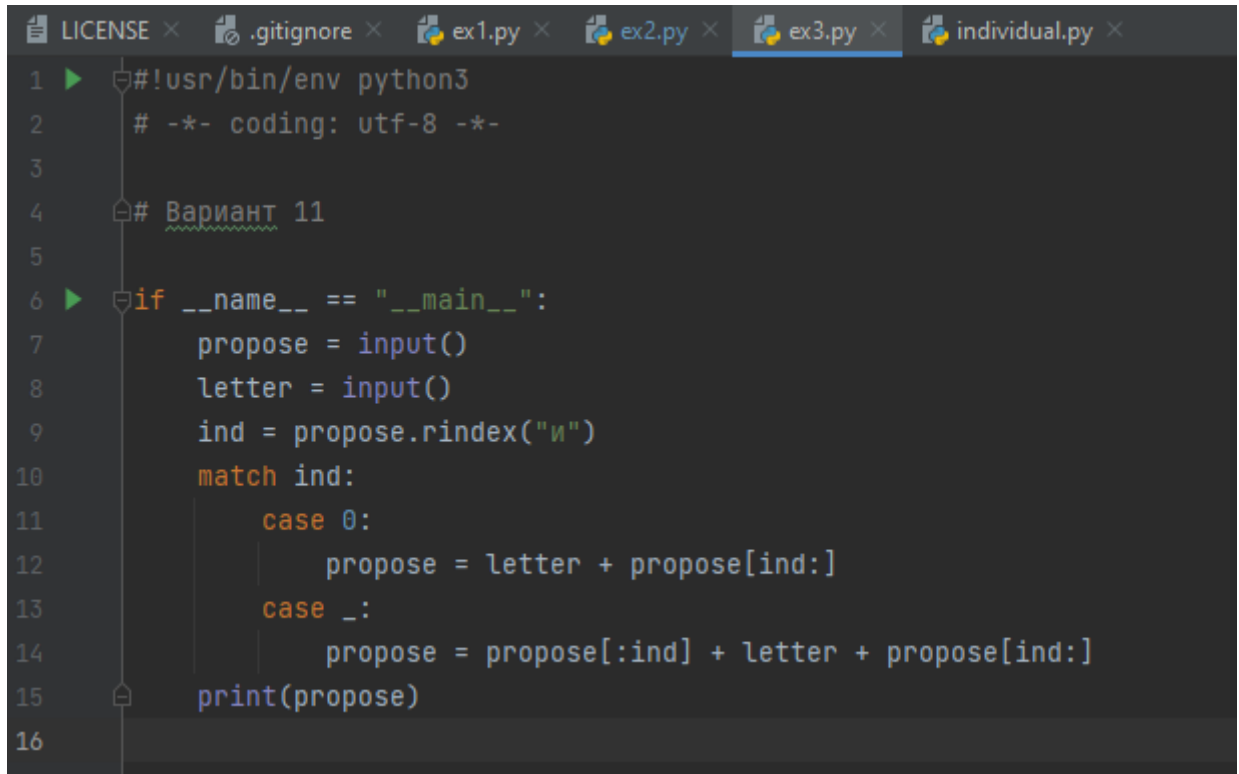


```
ex2 x
C:\Users\Sashka\AppData\Local\Programs\Python\Py
Чя тут написано ча может щя
Ча тут написано ча может ща
Process finished with exit code 0
```

Рисунок 9 – Пример выполнения программы

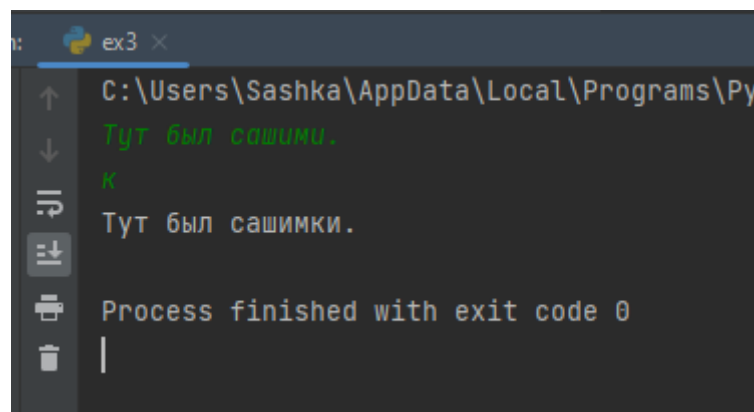
6. Выполнение задания №3.

Условие: Дано предложение, оканчивающее символом «.». Вставить заданную букву перед последней буквой и.



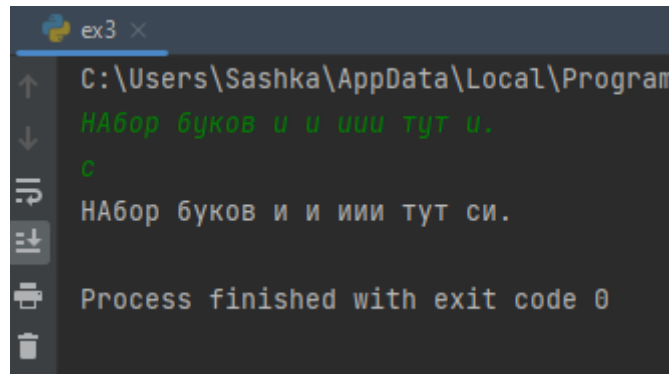
```
1  ▶ #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      # Вариант 11
5
6  ▶  if __name__ == "__main__":
7      propose = input()
8      letter = input()
9      ind = propose.rindex(".")
10     match ind:
11     case 0:
12         propose = letter + propose[ind:]
13     case _:
14         propose = propose[:ind] + letter + propose[ind:]
15     print(propose)
16
```

Рисунок 10 – Код программы



```
ex3 x
C:\Users\Sashka\AppData\Local\Programs\Python\Python310\python.exe
Тут был сашими.
к
Тут был сашимки.
Process finished with exit code 0
|
```

Рисунок 11 – Пример выполнения программы

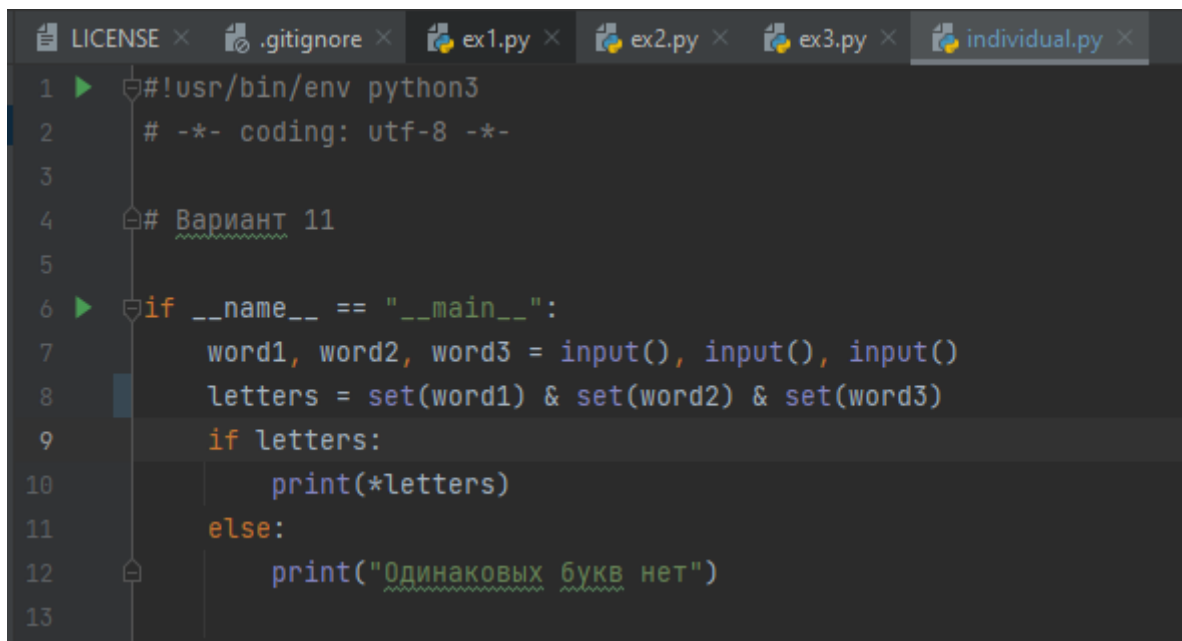


```
ex3 x
C:\Users\Sashka\AppData\Local\Program
НАбор букв и и ии тут и.
с
НАбор букв и и ии тут си.
Process finished with exit code 0
```

Рисунок 11 – Пример выполнения программы

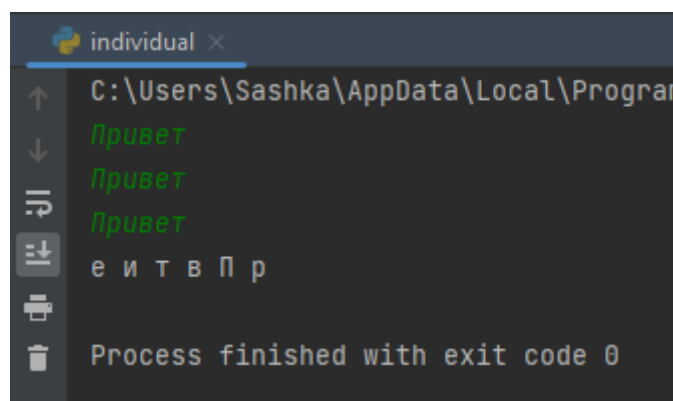
7. Выполнение индивидуального задания.

Условие: Даны три слова. Напечатать их общие буквы. Повторяющиеся буквы каждого слова не рассматривать.



```
LICENSE x .gitignore x ex1.py x ex2.py x ex3.py x individual.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Вариант 11
5
6 if __name__ == "__main__":
7     word1, word2, word3 = input(), input(), input()
8     letters = set(word1) & set(word2) & set(word3)
9     if letters:
10         print(*letters)
11     else:
12         print("Одинаковых букв нет")
13
```

Рисунок 12 – Код программы



```
individual x
C:\Users\Sashka\AppData\Local\Program
Привет
Привет
Привет
е и т в П р
Process finished with exit code 0
```

Рисунок 13 – Пример выполнения программы

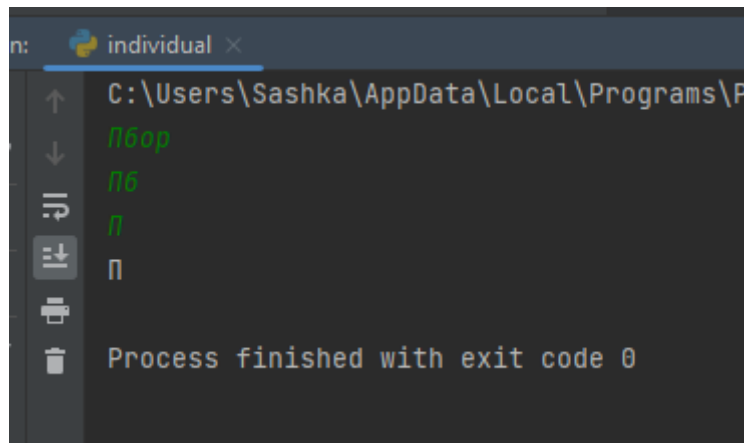


Рисунок 14 – Пример выполнения программы

Контрольные вопросы:

1. Строки в Python — это упорядоченная последовательность символов, используемых для хранения и представления текстовой информации.
2. Строковые литералы в Python можно задать с помощью одинарных, двойных или тройных кавычек. Также можно использовать обратный слеш для переноса строки.
3. В Python существуют различные операции и функции для работы со строками, включая конкатенацию, сравнение, поиск, замену, форматирование и другие.
4. Индексирование строк в Python начинается с 0. Первый символ имеет индекс 0, второй — индекс 1 и так далее.
5. Работа со срезами строк в Python позволяет получить подстроку, указав начальный и конечный индексы.
6. Строки в Python относятся к неизменяемому типу данных. Это означает, что после создания строки её нельзя изменить.
7. Можно использовать метод `istitle()`, который проверяет, начинается ли каждое слово в строке с заглавной буквы.
8. Для проверки вхождения одной строки в другую можно использовать метод `in` или `str.contains()`.

9. Для нахождения индекса первого вхождения подстроки в строку можно использовать метод `str.index()` или `str.find()`.

10. Для подсчёта количества символов в строке используется функция `len()`.

11. Для подсчёта количества определённого символа в строке используется метод `str.count()`.

12. F-строки — это способ форматирования строк в Python, который позволяет вставлять выражения внутри строки.

13. Можно использовать метод `str.find(sub[, start[, end]])`, где `start` и `end` — это индексы заданной части строки.

14. Метод `format()` позволяет вставить содержимое переменной в строку.

15. Метод `str.isdigit()` позволяет проверить, содержит ли строка только цифры.

16. Метод `str.split(sep)` позволяет разделить строку по заданному символу.

17. Метод `str.islower()` позволяет проверить, состоит ли строка только из строчных букв.

18. Можно проверить первый символ строки на строчность с помощью выражения `str[0].islower()`.

19. В Python нельзя прибавить целое число к строке напрямую, но можно преобразовать число в строку и затем выполнить конкатенацию.

20. Для «переворачивания» строки можно использовать срезы: `str[::-1]`.

21. Метод `str.join(iterable)` позволяет объединить список строк в одну строку.

22. Методы `str.upper()` и `str.lower()` приводят всю строку к верхнему или нижнему регистру соответственно.

23. Можно использовать индексацию и методы преобразования регистра: `str[0].upper() + str[1:-1] + str[-1].upper()`.

24. Метод `str.isupper()` позволяет проверить, состоит ли строка только из прописных букв.
25. Метод `splitlines()` полезен для разделения текста на строки по символам новой строки.
26. Метод `str.replace(old, new)` позволяет заменить все вхождения подстроки в строке.
27. Методы `str.startswith(prefix)` и `str.endswith(suffix)` позволяют проверить, начинается или заканчивается ли строка заданной последовательностью символов.
28. Метод `str.isspace()` позволяет проверить, состоит ли строка только из пробелов.
29. Если умножить строку на 3 в Python, то получится новая строка, в которой исходная строка повторяется три раза.
30. Можно использовать метод `str.title()`, который приводит первую букву каждого слова в строке к верхнему регистру.
31. Метод `partition(sep)` разделяет строку на три части по первому вхождению разделителя: часть перед разделителем, сам разделитель и часть после разделителя.
32. Метод `rfind()` используется для поиска последнего вхождения подстроки в строку.