

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 РАБОТЕ №2.3
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Кондратенко Даниил Витальевич
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(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.

Порядок выполнения работы:

Задание 1.

Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

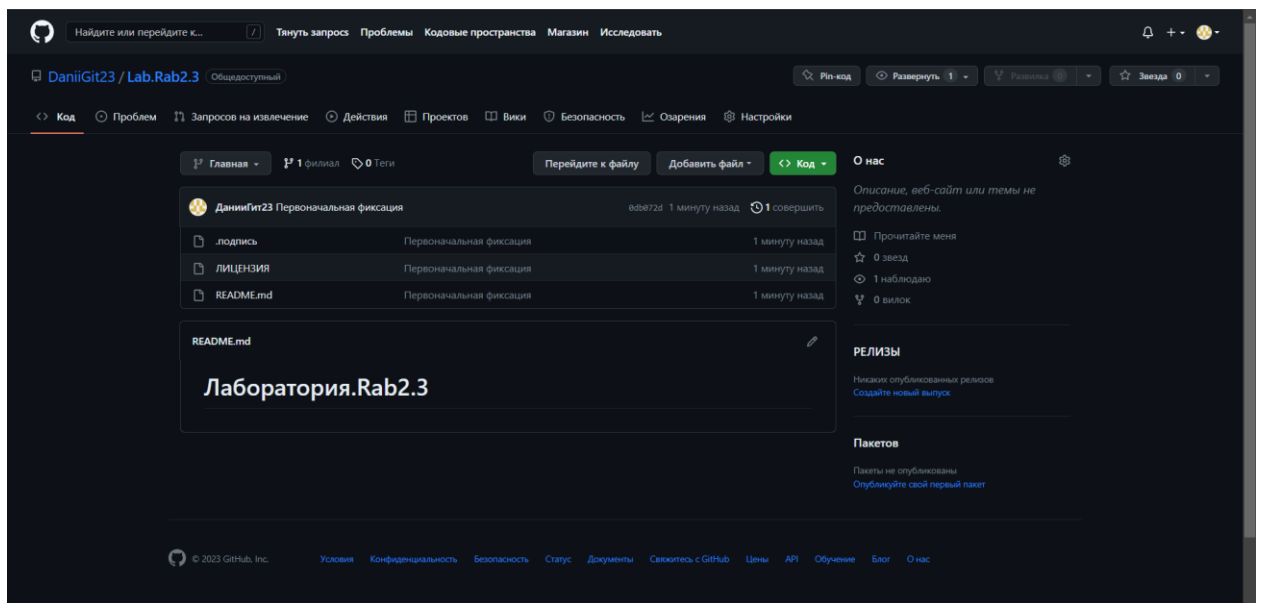
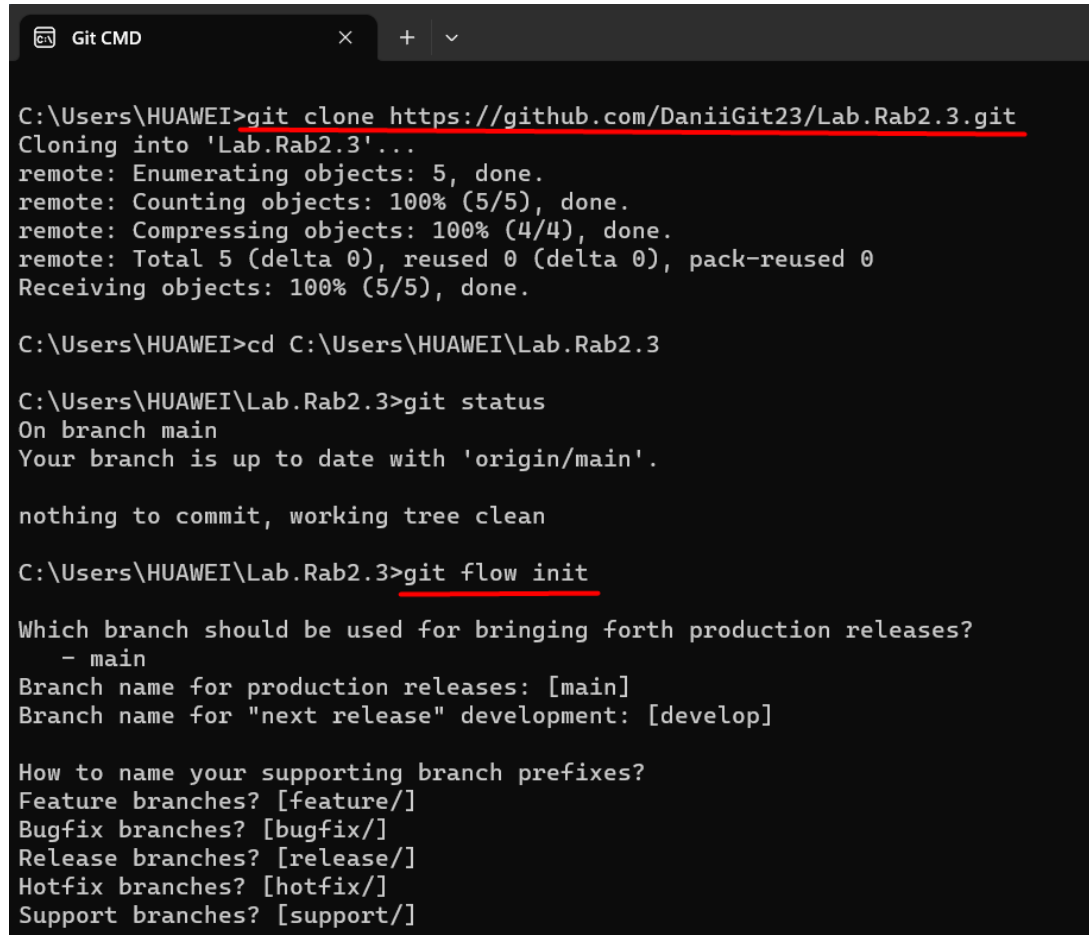


Рисунок 1 – Новый репозиторий

Задание 2.

Проклонировал свой репозиторий на свой компьютер.

Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop.



```
Git CMD
C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/Lab.Rab2.3.git
Cloning into 'Lab.Rab2.3'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab2.3

C:\Users\HUAWEI\Lab.Rab2.3>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\HUAWEI\Lab.Rab2.3>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
```

Рисунок 2 – Клонирование и модель ветвления git-flow

Реализовывал примеры и индивидуальные задания на основе ветки develop, без создания дополнительной ветки feature/(название ветки) по указанию преподавателя.

Задание 3.

Создал проект PyCharm в папке репозитория.

Работа с примером №1.

Добавил новый файл *zamena.py*.

Условие примера: Дано предложение. Все пробелы в нем заменить СИМВОЛОМ «_».

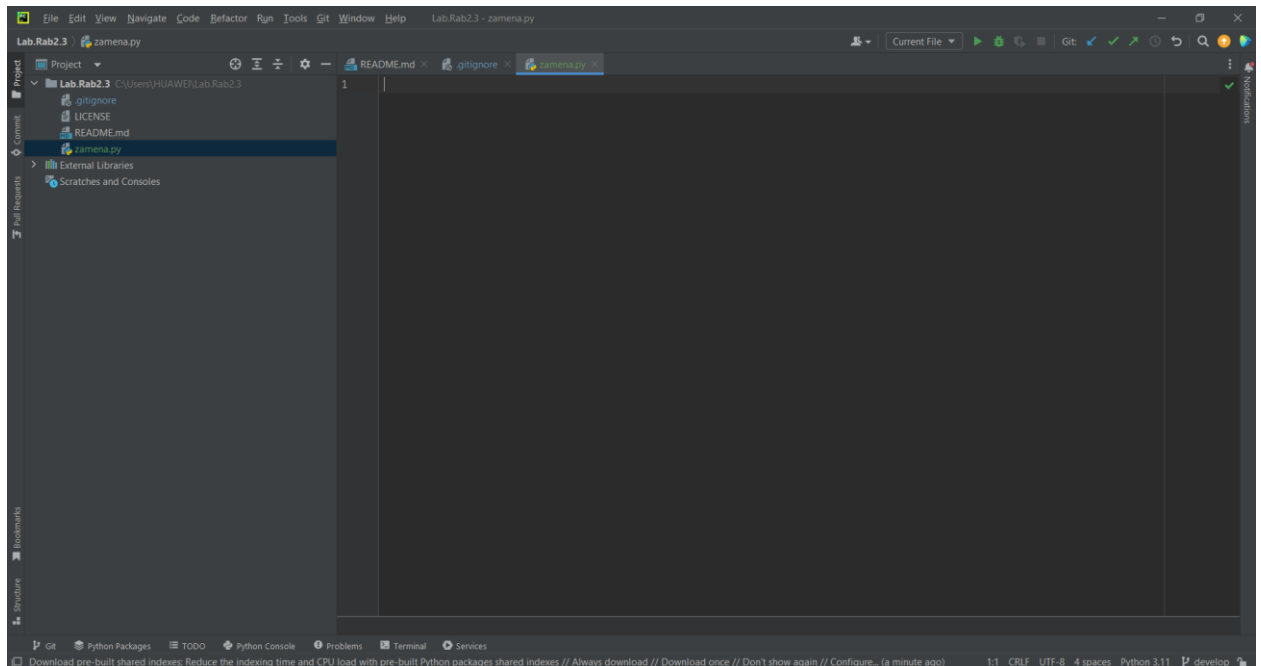


Рисунок 2 – Проект PyCharm и новый файл *zamena.py*

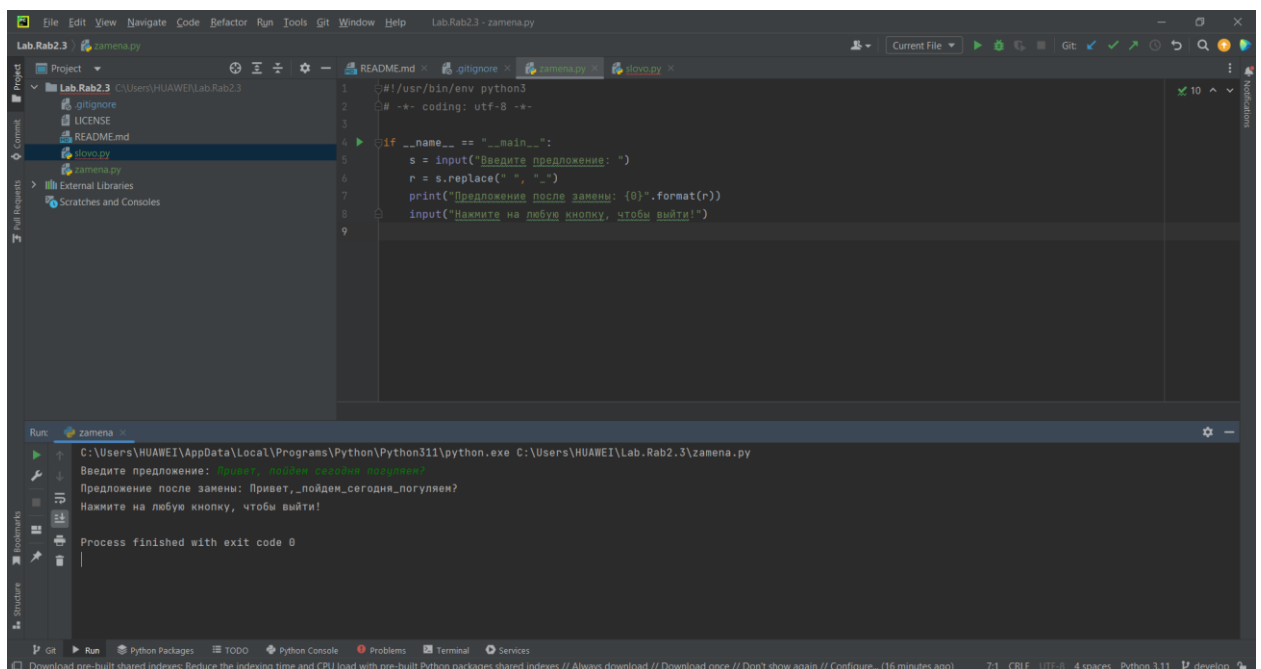


Рисунок 3 – Программа и ее результат

Задание 4.

Создал новый файл по названию *slovo.py*

Работа с примером №2.

Условие примера: Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.

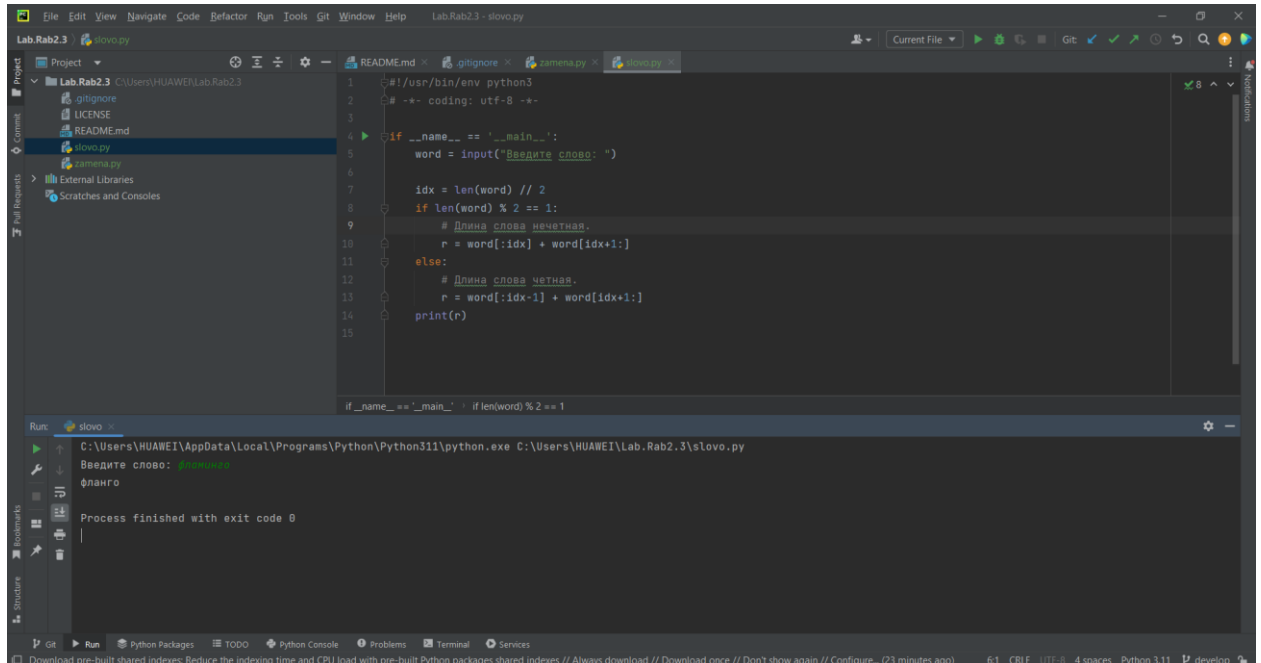


Рисунок 4 – Программа и ее результат (с четной длиной)

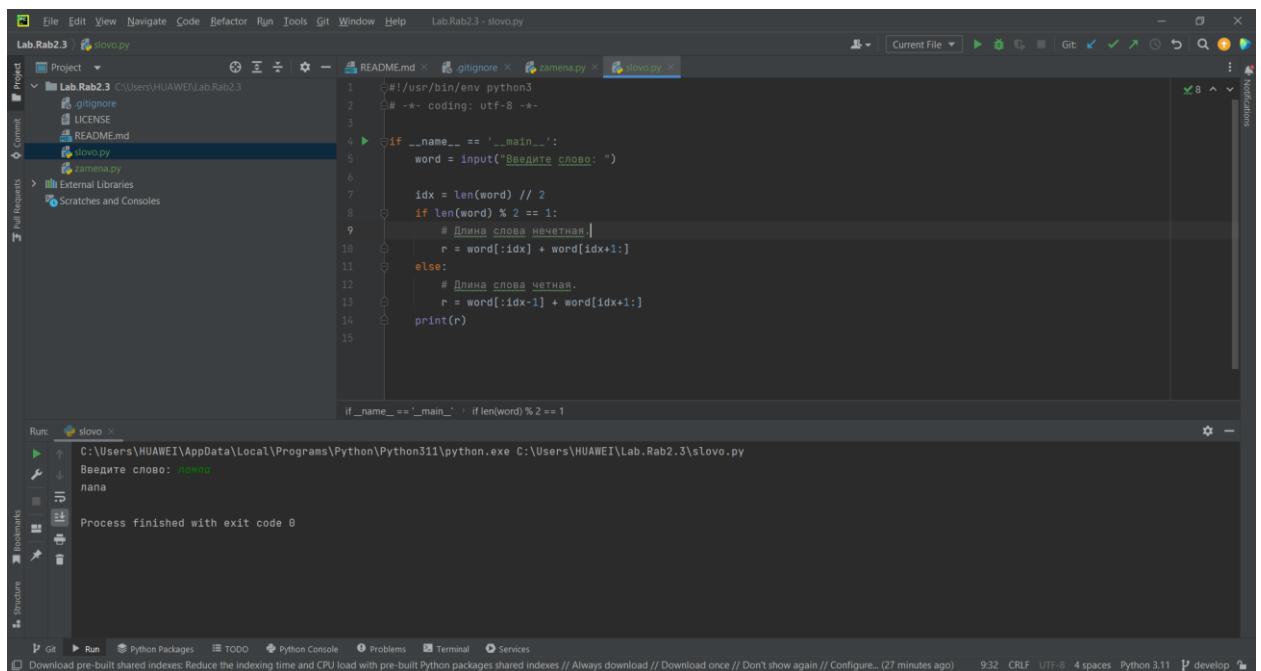


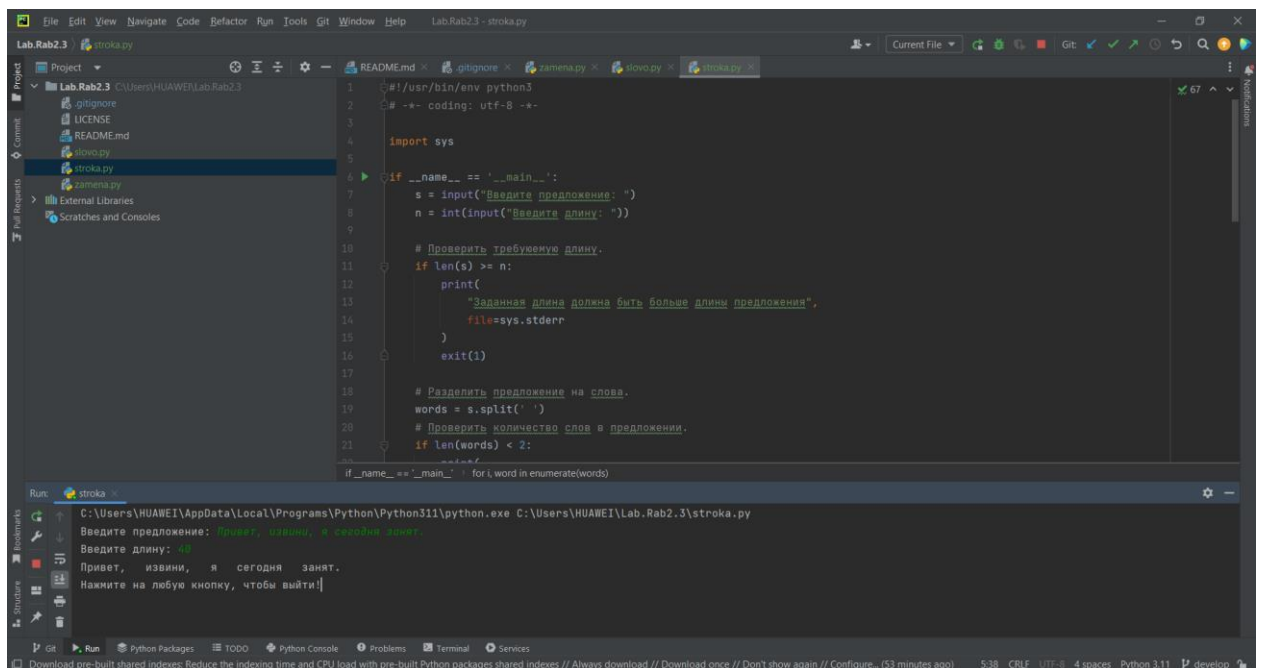
Рисунок 5 – Программа и ее результат (с нечетной длиной)

Задание 5.

Создал новый файл по названию *stroka.py*

Работа с примером №3.

Условие примера: Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки стала равна заданной длине (предполагается, что требуемая длина не меньше исходной). Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными словами должно отличаться не более чем на 1.



The screenshot shows a code editor with the file `stroka.py` open. The code is as follows:

```
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
10     # Проверить требуемую длину.
11     if len(s) >= n:
12         print(
13             "Заданная длина должна быть больше длины предложения",
14             file=sys.stderr,
15         )
16         exit(1)
17
18     # Разделить предложение на слова.
19     words = s.split(' ')
20     # Проверить количество слов в предложении.
21     if len(words) < 2:
22         print("В предложении должно быть не менее 2 слов",
23               file=sys.stderr,
24         )
25         exit(1)
26
27     if __name__ == '__main__':
28         for i, word in enumerate(words):
29             # ... (code continues but is partially obscured)
```

The Run console at the bottom shows the execution of the program:

```
C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab2.3\stroka.py
Введите предложение: Привет, извини, я сегодня занят.
Введите длину: 40
Привет, извини, я сегодня занят.
Нажмите на любую кнопку, чтобы выйти
```

Рисунок 6 – Программа и ее результат

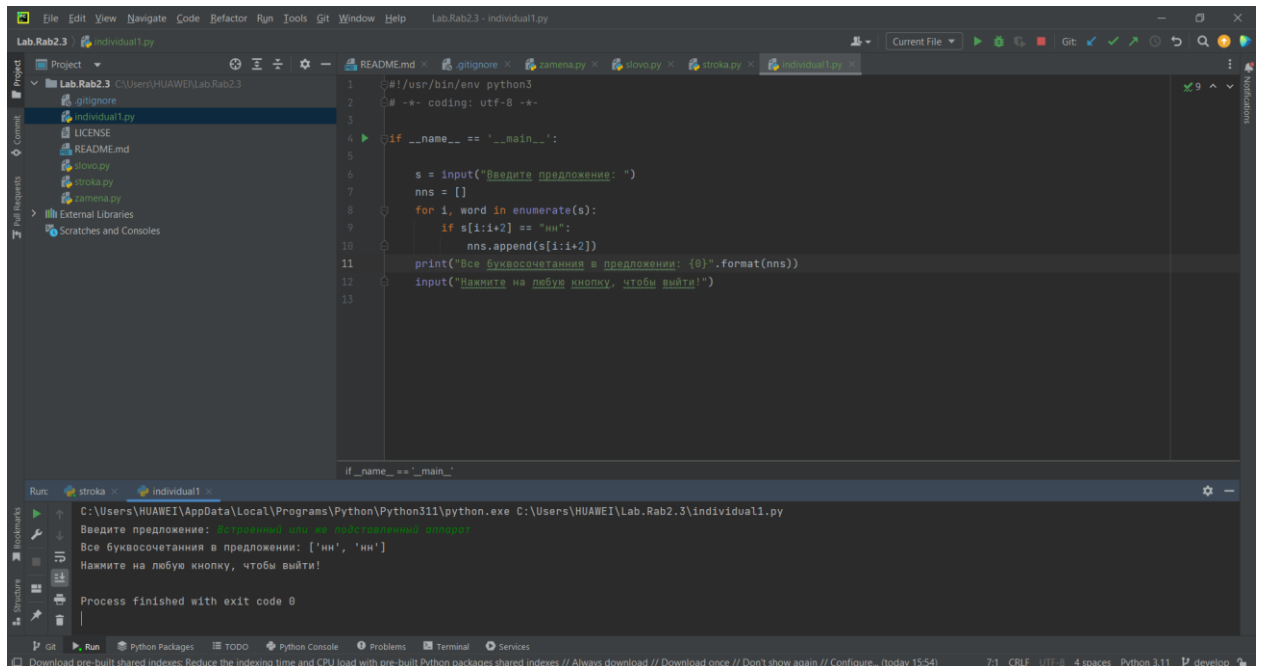
Задание 6.

Выполнение индивидуального задание №1.

Создал новый файл под названием *individual1.py*

Вариант 12 (по списку группы).

Условие задания: дано предложение. Вывести все имеющиеся в нем буквосочетания нн.



The screenshot shows a code editor with a file named `individual1.py`. The code is a Python script that prompts the user to enter a sentence and then prints all occurrences of the substring "nn" within that sentence. The script uses a loop to iterate over the sentence and a list to store the found substrings.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     nns = []
7     for i, word in enumerate(s):
8         if s[i:i+2] == "nn":
9             nns.append(s[i:i+2])
10    print("Все буквосочетания в предложении: {}".format(nns))
11    input("Нажмите на любую кнопку, чтобы выйти!")
12
13 if __name__ == '__main__':
```

The output of the script is shown in the Run window at the bottom. It displays the input sentence, the found substrings, and the prompt to press any key to exit.

```
Run: stroka > individual1
C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab2.3\individual1.py
Введите предложение: В предложении есть nn. Это предложение.
Все буквосочетания в предложении: ['nn', 'nn']
Нажмите на любую кнопку, чтобы выйти!
Process finished with exit code 0
```

Рисунок 7 – Программа и ее результат

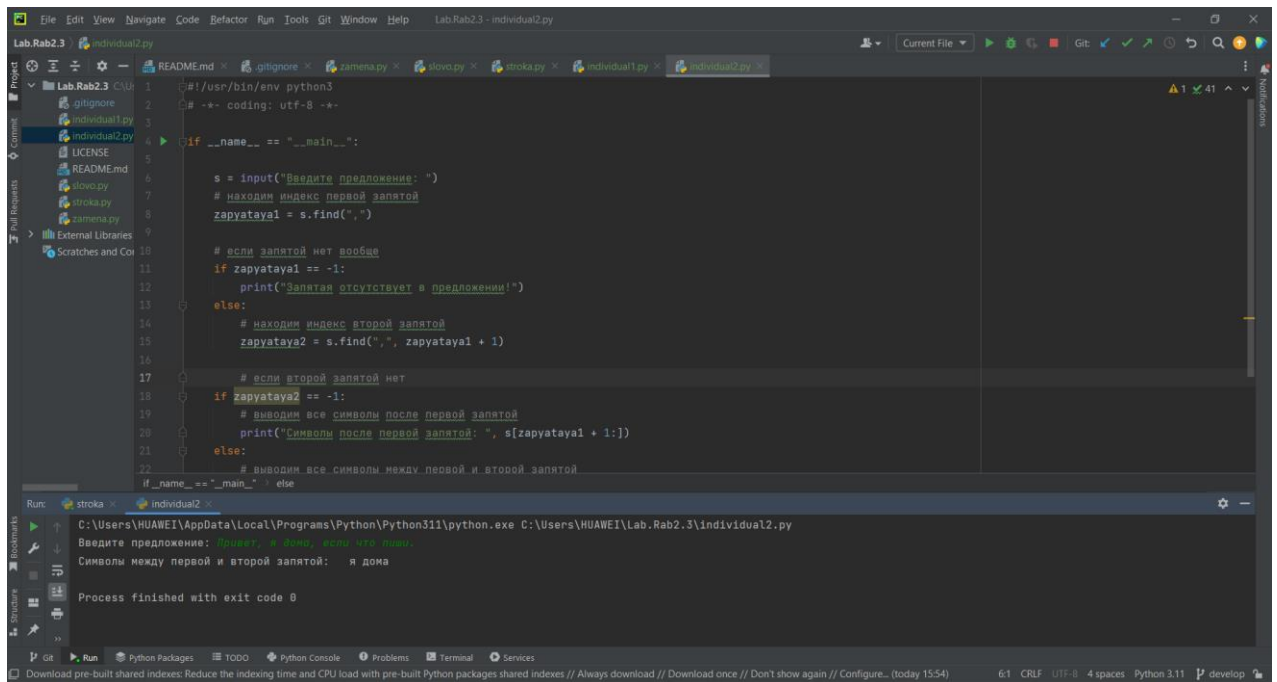
Задание 7.

Выполнение индивидуального задание №2.

Создал новый файл под названием *individual2.py*

Вариант 12 (по списку группы).

Условие задания: 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.



The screenshot shows a code editor with a file named `individual2.py`. The code is a Python script that takes a user input and prints symbols between the first and second commas, or after the first comma if there is no second comma.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    s = input("Введите предложение: ")
    # находим индекс первой запятой
    zapyataya1 = s.find(",")

    # если запятой нет вообще
    if zapyataya1 == -1:
        print("Запятая отсутствует в предложении!")
    else:
        # находим индекс второй запятой
        zapyataya2 = s.find(",", zapyataya1 + 1)

        # если второй запятой нет
        if zapyataya2 == -1:
            # выводим все символы после первой запятой
            print("Символы после первой запятой: ", s[zapyataya1 + 1:])
        else:
            # выводим все символы между первой и второй запятой
            print("Символы между первой и второй запятой: ", s[zapyataya1 + 1:zapyataya2])
```

The Run console shows the execution of the script. The user input is "Я люблю, если что, дома". The output is "Символы между первой и второй запятой: я дома".

Рисунок 7 – Программа и ее результат

Задание 8.

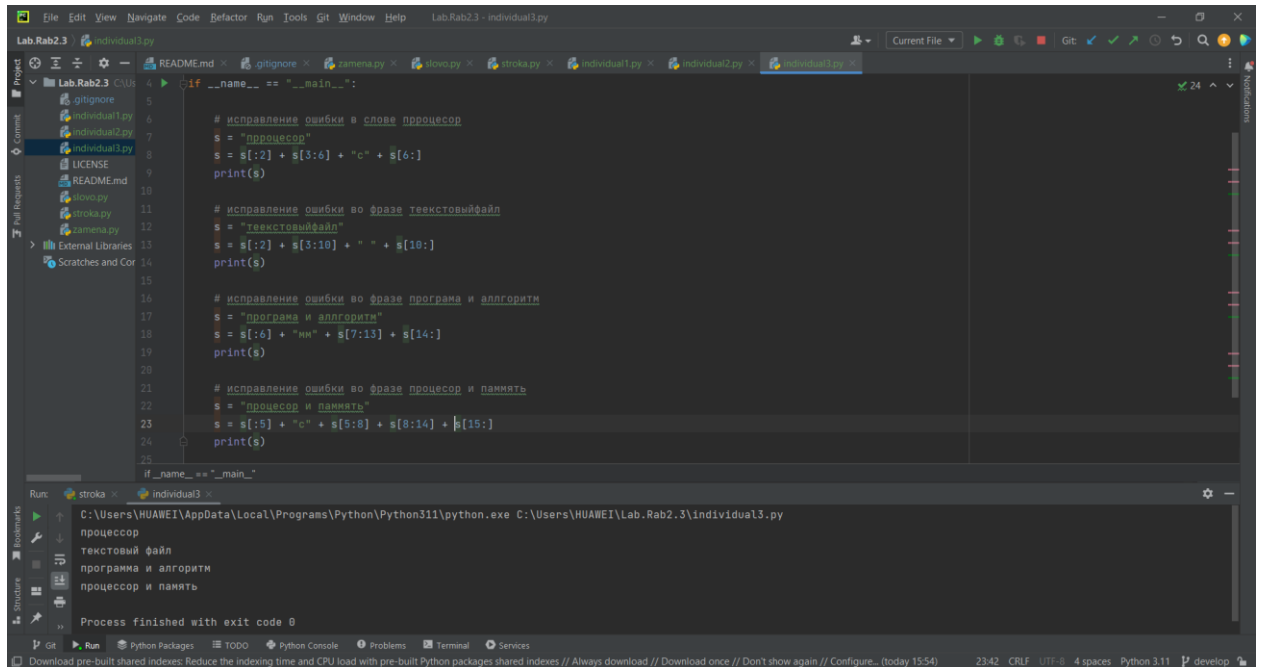
Выполнение индивидуального задание №3.

Создал новый файл под названием *individual3.py*

Вариант 12 (по списку группы).

Условие задания: Путем вставок и удаления символов исправить ошибки:

1. в слове прроцесор;
2. во фразе теекстовыйфайл;
3. во фразе програма и аллгоритм;
4. во фразе процесор и паммать.



```
Lab.Rab2.3 - individual3.py
if __name__ == "__main__":
    # исправление ошибки в слове прроцесор
    s = "прроцесор"
    s = s[:2] + s[3:6] + "с" + s[6:]
    print(s)

    # исправление ошибки во фразе теекстовыйфайл
    s = "теекстовыйфайл"
    s = s[:2] + s[3:10] + " " + s[10:]
    print(s)

    # исправление ошибки во фразе програма и аллгоритм
    s = "програма и аллгоритм"
    s = s[:6] + "нн" + s[7:13] + s[14:]
    print(s)

    # исправление ошибки во фразе процесор и паммать
    s = "процесор и паммать"
    s = s[:5] + "с" + s[5:8] + s[8:14] + s[15:]
    print(s)

if __name__ == "__main__":
```

Run: C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab2.3\individual3.py

процессор
текстовый файл
программа и алгоритм
процессор и память

Process finished with exit code 0

Рисунок 8 – Программа и ее результат

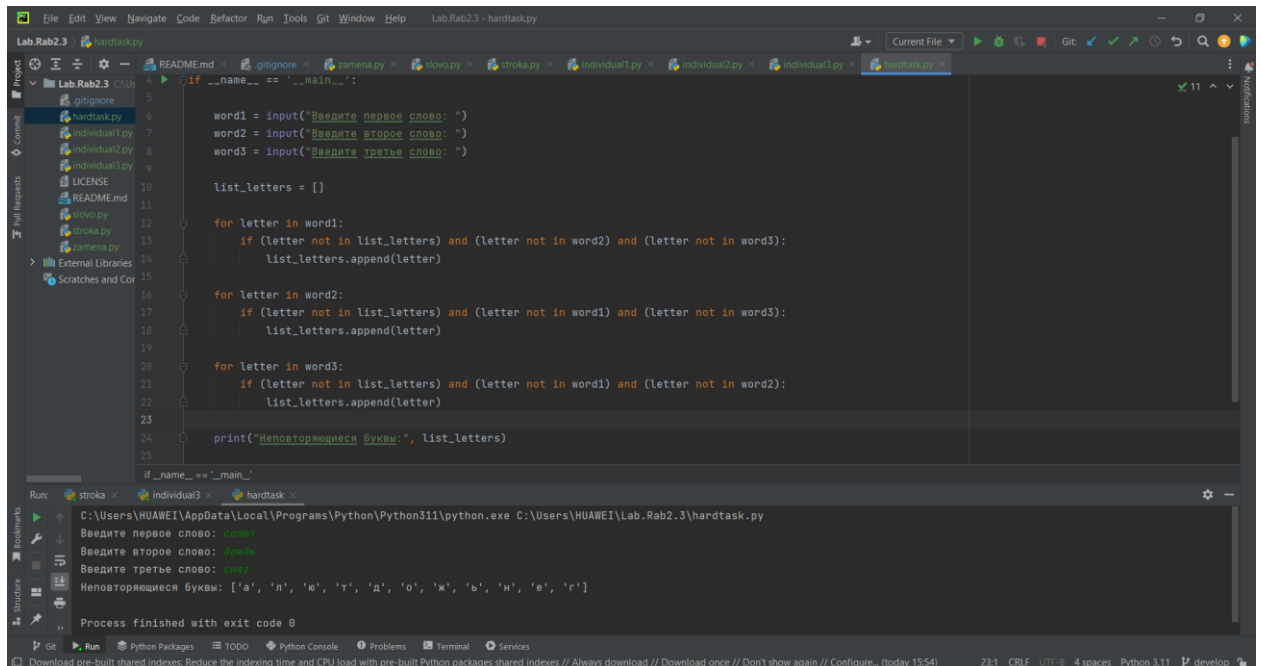
Задание 9.

Выполнение индивидуального задания повышенной сложности.

Создал новый файл под названием *hardtask.py*

Вариант 12 (по списку группы).

Условие задания: Даны три слова. Напечатать неповторяющиеся в них буквы.



```
Lab.Rab2.3 - hardtask.py
1  if __name__ == '__main__':
2
3      word1 = input("Введите первое слово: ")
4      word2 = input("Введите второе слово: ")
5      word3 = input("Введите третье слово: ")
6
7      list_letters = []
8
9      for letter in word1:
10         if (letter not in list_letters) and (letter not in word2) and (letter not in word3):
11             list_letters.append(letter)
12
13     for letter in word2:
14         if (letter not in list_letters) and (letter not in word1) and (letter not in word3):
15             list_letters.append(letter)
16
17     for letter in word3:
18         if (letter not in list_letters) and (letter not in word1) and (letter not in word2):
19             list_letters.append(letter)
20
21     print("Неповторяющиеся буквы:", list_letters)
22
23 if __name__ == '__main__':
```

```
Run: stroka - hardtask
C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab2.3\hardtask.py
Введите первое слово: слово
Введите второе слово: строка
Введите третье слово: замена
Неповторяющиеся буквы: ['с', 'л', 'о', 'в', 'е', 'н', 'а', 'з', 'м', 'п', 'р', 'к', 'г']
Process finished with exit code 0
```

Рисунок 9 – Программа и ее результат

Задание 10.

Слил ветку develop с веткой main и отправил на удаленный сервер.

```
C:\Users\HUAWEI\Lab.Rab2.3>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\HUAWEI\Lab.Rab2.3>git merge develop
Updating 0db072d..9d19209
Fast-forward
 .gitignore      | 4 ++++
 README.md       | 3 ++-
 hardtask.py     | 25 ++++++
 individual1.py  | 12 ++++++
 individual2.py  | 24 ++++++
 individual3.py  | 25 ++++++
 slovo.py        | 15 ++++++
 stroka.py       | 57 ++++++
 zamena.py       | 8 ++++++
 9 files changed, 172 insertions(+), 1 deletion(-)
 create mode 100644 hardtask.py
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 individual3.py
 create mode 100644 slovo.py
 create mode 100644 stroka.py
 create mode 100644 zamena.py
```

Рисунок 10 – Слияние веток

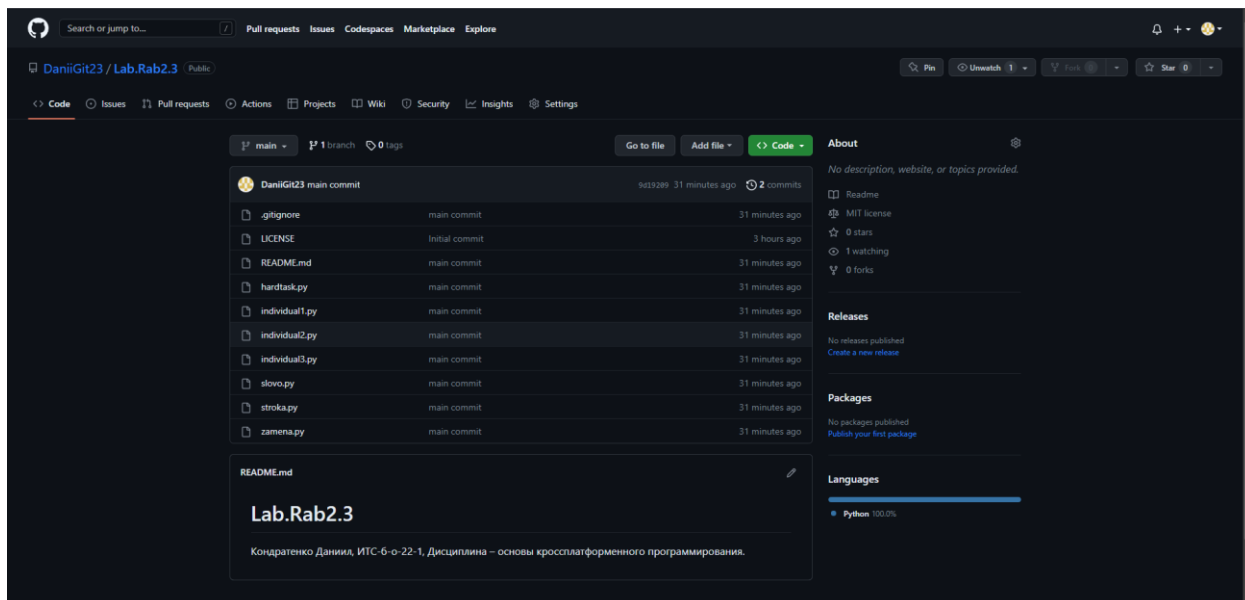


Рисунок 11 – Удаленный сервер

Ссылка на репозиторий: <https://github.com/DaniiGit23/Lab.Rab2.3.git>

Контрольные вопросы:

1. Что такое строки в языке Python?

Строки в языке Python являются типом данных, специально предназначенным для обработки текстовой информации. Строка может содержать произвольно длинный текст (ограниченный имеющейся памятью). В новых версиях Python имеются два типа строк: обычные строки (последовательность байтов) и Unicode-строки (последовательность символов).

2. Какие существуют способы задания строковых литералов в языке Python?

Литералы строк позволяют интерпретатору Python убедиться, что перед ним действительно находится строка. Такой подход называется "утиной" типизацией – если что-то плавает как утка, крикает как утка и откладывает яйца как утка, то скорее всего это действительно утка. То же самое и с литералами строк – если что-то соответствует литералам строк, то это можно считать строкой.

3. Какие операции и функции существуют для строк?

Для работы со строками существуют следующие стандартные процедуры и функции: `d:=copy (a, x, y)` функция, возвращает копию из `y` (`integer`) знаков части строки `a` (`string`), начиная с позиции `x` (`integer`). Формат: `d:=copy (a, x, y)`. Результат записывается в переменную `d`, например: `a:='бегемот'; d:=copy (a,5,3); writeln (d);` результат - `мот` `delete (a, x, y)` процедура, удаляет `y` (`integer`) знаков части строки `a` (`string`), начиная с позиции `x` (`integer`).

4. Как осуществляется индексирование строк?

Доступ к символам в строках основан на операции индексирования – после строки или имени переменной, ссылающейся на строку, в квадратных скобках указываются номера позиций необходимых символов. Так же следует понимать, что этот самый доступ, основан на смещении, т.е. расстоянии символов от левого или правого края строки. Данное расстояние измеряется целыми числами и, по сути, определяет номер позиции символов в строке – их индекс.

5. Как осуществляется работа со срезами для строк?

Есть три формы срезов:

Самая простая форма среза - взятие одного символа строки - `S[i]`, где `S` - строка, `i` - индекс.

Второй тип - срез с двумя параметрами. Т.е. `S[a:b]` возвращает подстроку, начиная с символа с индексом **a** до символа с индексом **b**, не включая его. Если опустить второй параметр (но поставить двоеточие), то срез берется до конца строки.

Срез с тремя параметрами - `S[a:b:d]`. Третий параметр задает шаг (как в случае с функцией `range`), то есть будут взяты символы с индексами `a`, `a + d`, `a + 2 * d` и т. д. Например, при задании значения третьего параметра, равному 2, в срез попадет каждый второй символ.

6. Почему строки Python относятся к неизменяемому типу данных?

Python обрабатывает изменяемые и неизменяемые объекты по-разному.

Доступ к неизменяемым объектам осуществляется быстрее, чем к изменяемым.

Изменяемые объекты отлично подойдут, если вам нужно менять размер объектов, например, list, dict и т.д.

Неизменяемые значения используются, когда вам нужно убедиться, что созданный вами объект никогда не будет меняться.

Неизменяемые объект принципиально дорого менять, поскольку для этого требуется создать копию, а изменяемые менять легко.

7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

Метод `str.istitle()` возвращает `True`, если каждое слово в строке `str` начинается с заглавной буквы и в ней есть хотя бы один символ в верхнем регистре. Возвращает `False` в противном случае. Например, заглавные буквы могут следовать только за непрописанными символами, а строчные - только за прописными.

8. Как проверить строку на вхождение в неё другой строки?

Использование `find()` для проверки наличия в строке другой подстроки. Мы также можем использовать функцию `string find()`, чтобы проверить, содержит ли строка подстроку или нет. Эта функция возвращает первую позицию индекса, в которой найдена подстрока, иначе возвращает `-1`.

9. Как найти индекс первого вхождения подстроки в строку?

Для поиска подстроки в строке в Python применяется метод `find()`, который возвращает индекс первого вхождения подстроки в строку и имеет три формы:

`find(str)` : поиск подстроки `str` ведется с начала строки до ее конца

`find(str, start)` : параметр `start` задает начальный индекс, с которого будет производиться поиск

`find(str, start, end)` : параметр `end` задает конечный индекс, до которого будет идти поиск

10. Как подсчитать количество символов в строке?

Метод `len ()` подсчитывает общее количество символов в строке. Если нам нужно подсчитать, сколько раз в строке встречается определенный символ или последовательность символов, мы можем использовать метод `str.count ()`.

11. Как подсчитать то, сколько раз определённый символ встречается в строке?

Давайте возьмем нашу строку `ss = "Sammy Shark!"` и подсчитаем, сколько раз в ней встречается символ "a": `print (ss. count (" a"))` Output 2. Мы можем поискать и другой символ: `print (ss. count (" s"))` Output 0.

12. Что такое f-строки и как ими пользоваться?

F-строчный метод. Это новый механизм форматирования строк, представленный PEP 498. Он также известен как интерполяция литеральной строки или, чаще, как F-строки (символ `f`, предшествующий строковому литералу). Основная задача этого механизма – облегчить интерполяцию. Когда мы добавляем строке букву 'F', строка сама становится f-строкой. F-строка может быть отформатирована почти так же, как метод `str.format ()`.

13. Для поиска подстроки в строке Python, используется специальный метод `find ()`. Метод `find ()` как и большинство остальных методов, работает довольно просто. Если искомый элемент найден в строке, он вернет нам индекс первого вхождения, если не найден он вернет нам -1.

14. Как вставить содержимое переменной в строку, воспользовавшись методом `format()`?

Метод `format()` позволяет добиваться результатов, сходных с теми, которые можно получить, применяя f-строки. Правда, я полагаю, что использовать `format()` не так удобно, так как все переменные приходится указывать в качестве аргументов `format()`.

15. Как узнать о том, что в строке содержатся только цифры?

Существует метод `isnumeric()`, который возвращает `True` в том случае, если все символы, входящие в строку, являются цифрами. Знаки препинания он цифрами не считает.

16. Как разделить строку по заданному символу?

Здесь нам поможет метод `split()` который разбивает строку по заданному символу или по нескольким символам.

17. Как проверить строку на то, что она составлена только из строчных букв?

Метод `islower()` возвращает `True` только в том случае, если строка составлена исключительно из строчных букв.

18. Как проверить то, что строка начинается со строчной буквы?

Сделать это можно, вызвав вышеописанный метод `islower()` для первого символа строки.

19. Можно ли в Python прибавить целое число к строке?

В некоторых языках это возможно, но Python при попытке выполнения подобной операции будет выдана ошибка `TypeError`.

20. Как «перевернуть» строку?

Для того чтобы «перевернуть» строку, её можно разбить, представив в виде списка символов, «перевернуть» список, и, объединив его элементы, сформировать новую строку.

21. Как объединить список строк в одну строку, элементы которой разделены дефисами?

Метод `join ()` умеет объединять элементы списков в строки, разделяя отдельные строки с использованием заданного символа.

22. Как привести всю строку к верхнему или нижнему регистру?

Для решения этих задач можно воспользоваться методами `upper()` и `lower()`, которые, соответственно, приводят все символы строк к верхнему и нижнему регистрам.

23. Как преобразовать первый и последний символы строки к верхнему регистру?

Тут, как и в одном из предыдущих примеров, мы будем обращаться к символам строки по индексам. Строки в Python иммутабельны, поэтому мы будем заниматься сборкой новой строки на основе существующей.

24. Как проверить строку на то, что она составлена только из прописных букв?

Имеется метод `isupper()`, который похож на уже рассмотренный `isupper()`. Но `isupper()` возвращает `True` только в том случае, если вся строка состоит из прописных букв.

25. В какой ситуации вы воспользовались бы методом `splitlines()` ?

Метод `splitlines()` разделяет строки по символам разрыва строки.

26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?

Если обойтись без экспорта модуля, позволяющего работать с регулярными выражениями, то для решения этой задачи можно воспользоваться методом `replace()`.

27. Как проверить то, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов?

В состав алфавитно-цифровых символов входят буквы и цифры. Для ответа на этот вопрос можно воспользоваться методом `isalnum()`.

28. Как узнать о том, что строка включает в себя только пробелы?

Есть метод `isspace()` который возвращает `True` только в том случае, если строка состоит исключительно из пробелов.

29. Что случится, если умножить некую строку на 3?

Будет создана новая строка, представляющая собой исходную строку, повторённую три раза.

30. Как привести к верхнему регистру первый символ каждого слова в строке?

Существует метод `title()`, приводящий к верхнему регистру первую букву каждого слова в строке.

31. Как пользоваться методом `partition()` ?

Метод `partition()` разбивает строку по заданной подстроке. После этого результат возвращается в виде кортежа. При этом подстрока, по которой осуществлялась разбивка, тоже входит в кортеж.

32. В каких ситуациях пользуются методом `rfind()` ?

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.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python3.