

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:
Липчанский Дмитрий Сергеевич
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:

1. Создал новый репозиторий и клонировал его на свой компьютер.

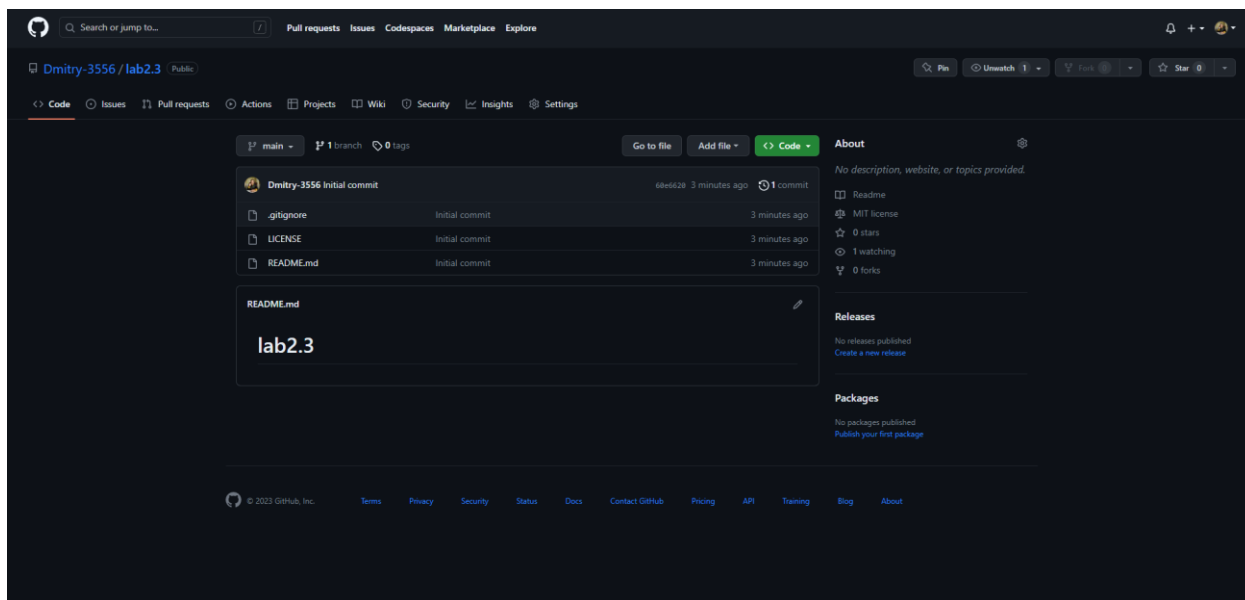


Рисунок 1 – Новый репозиторий

2. В ходе данной лабораторной работы работал с моделью ветвления git-flow.

```
C:\Work>git clone https://github.com/Dmitry-3556/lab2.3.git
Cloning into 'lab2.3'...
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:\Work>cd C:\Work\lab2.3

C:\Work\lab2.3>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Work\lab2.3>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/]
Version tag prefix? []
Hooks and filters directory? [C:/Work/lab2.3/.git/hooks]

C:\Work\lab2.3>
```

Рисунок 2 – Клонирование и модель ветвления git-flow

3. Пример №1.

Добавил новый файл ex1.py.

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

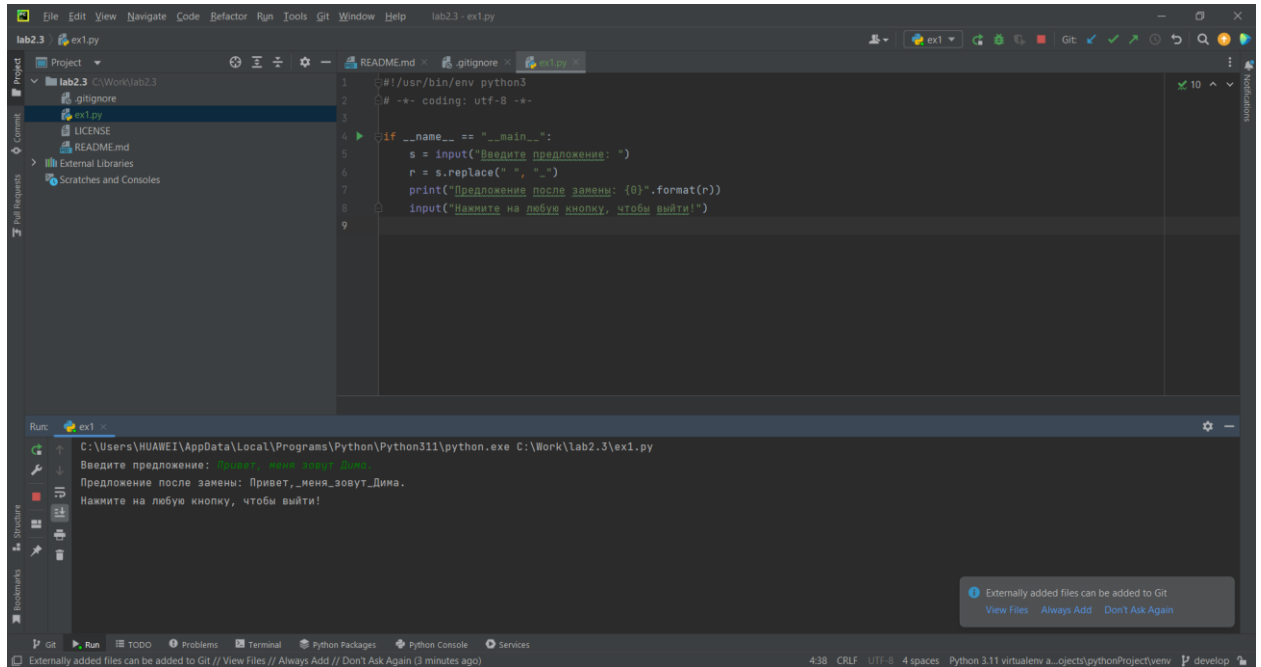


Рисунок 3 – Программа и ее результат

4. Создал новый файл под названием *ex2.py*

Пример №2.

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

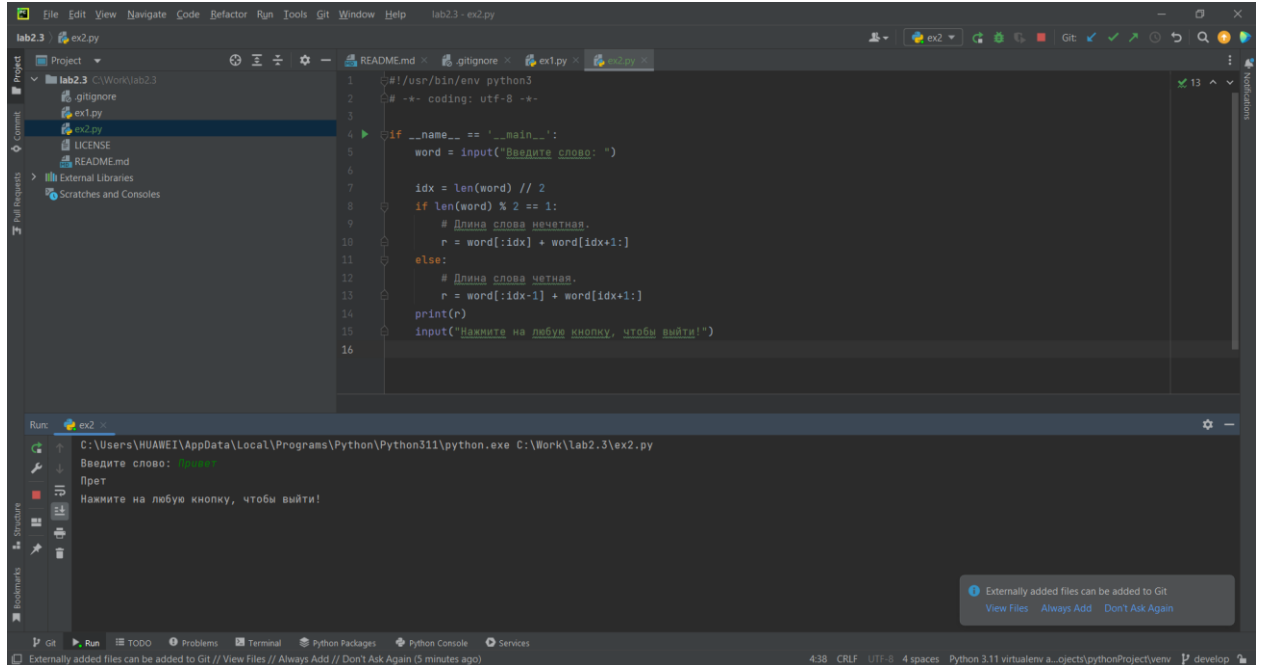


Рисунок 4 – Программа и ее результат с четной длиной

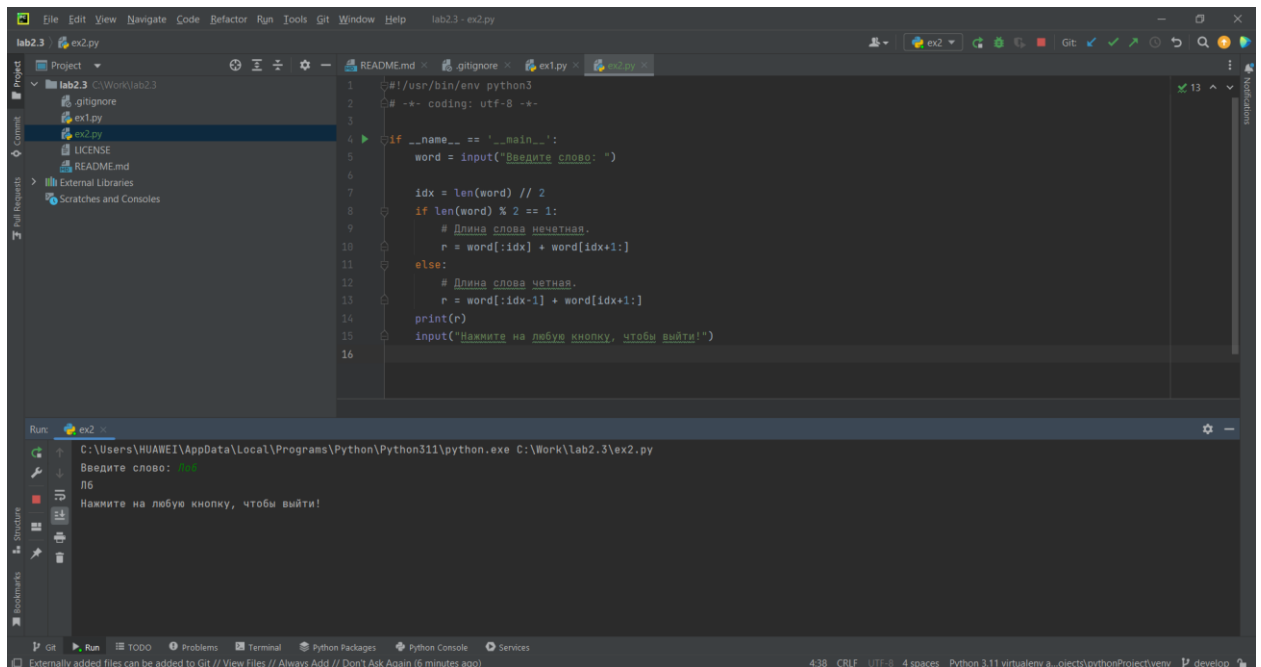
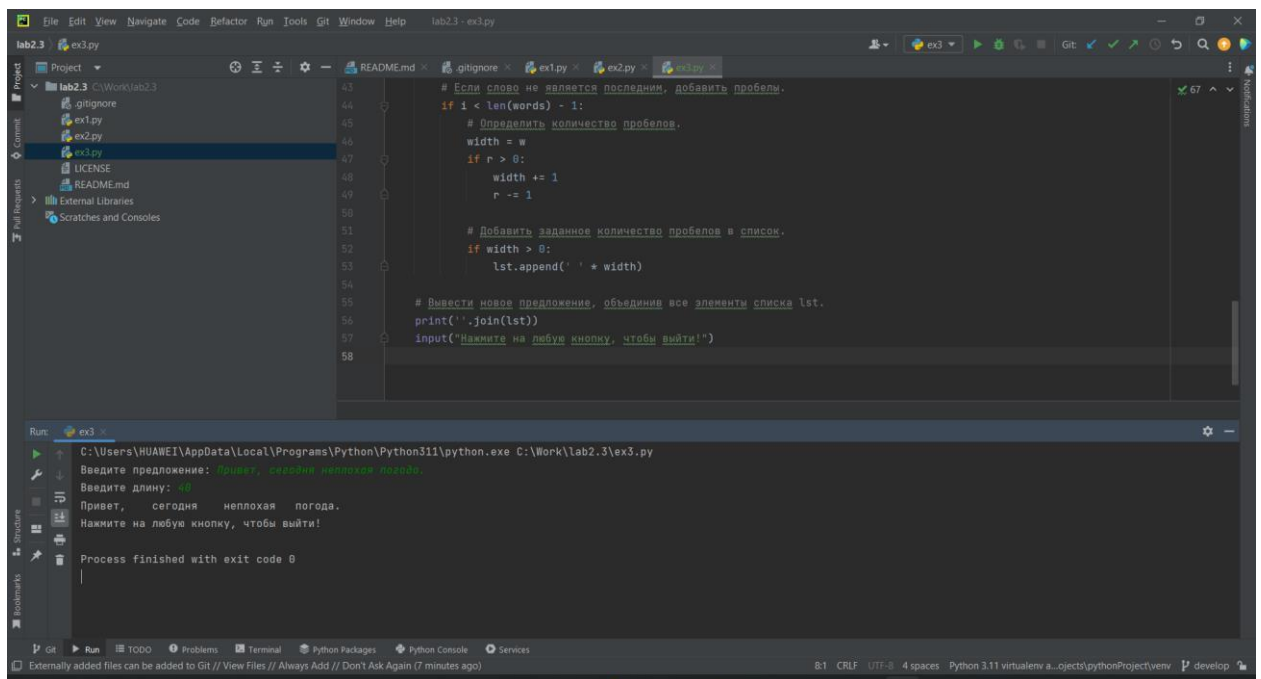


Рисунок 5 – Программа и ее результат с нечетной длиной

5. Создал новый файл под названием *ex3.py*

Пример №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 a file named `ex3.py`. The code is a Python script that takes a sentence and a target length as input and inserts spaces between words to reach the target length. The script uses a loop to calculate the number of spaces needed and inserts them one by one. The output window shows the execution of the script with the input sentence "Привет, сегодня неплохая погода." and the target length 40, resulting in the output "Привет, сегодня неплохая погода." and a message to press a button to exit.

```
43 # Если слово не является последним, добавить пробелы.
44 if i < len(words) - 1:
45     # Определить количество пробелов.
46     width = w
47     if r > 0:
48         width += 1
49         r -= 1
50
51     # Добавить заданное количество пробелов в список.
52     if width > 0:
53         lst.append(' ' * width)
54
55 # Вывести новое предложение, объединив все элементы списка lst.
56 print(''.join(lst))
57 input("Нажмите на любую кнопку, чтобы выйти!")
58
```

Run: ex3

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Work\lab2.3\ex3.py

Введите предложение: Привет, сегодня неплохая погода.

Введите длину: 40

Привет, сегодня неплохая погода.

Нажмите на любую кнопку, чтобы выйти!

Process finished with exit code 0

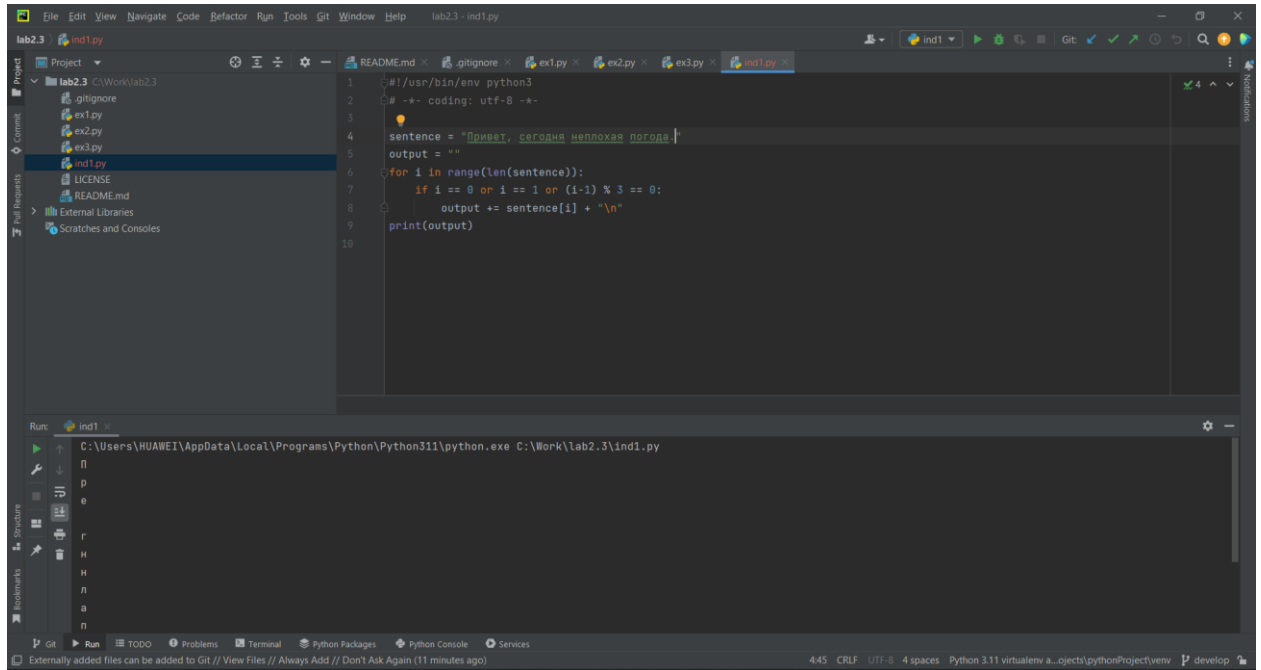
Рисунок 6 – Программа и ее результат

6. Индивидуальное задание №1.

Создал новый файл под названием *ind1.py*

Вариант 14 (по списку группы).

Условие задания: дано предложение. Вывести «столбиком» его первый, второй, пятый, шестой, девятый, десятый и т. д. символы.



The screenshot shows a code editor with a file named `ind1.py`. The code is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 sentence = "Привет, сегодня неплохая погода"
5 output = ""
6 for i in range(len(sentence)):
7     if i == 0 or i == 1 or (i-1) % 3 == 0:
8         output += sentence[i] + "\n"
9 print(output)
```

Below the editor, the Run console shows the output of the script:

```
П
р
е
  
г
  
н
  
н
  
н
  
а
  
н
```

Рисунок 7 – Программа и ее результат

7. Индивидуальное задание №2.

Создал новый файл под названием *ind2.py*

Вариант 14

Условие задания:

14. Дана строка, в которой есть слово или. Определить, сколько раз оно встречается.

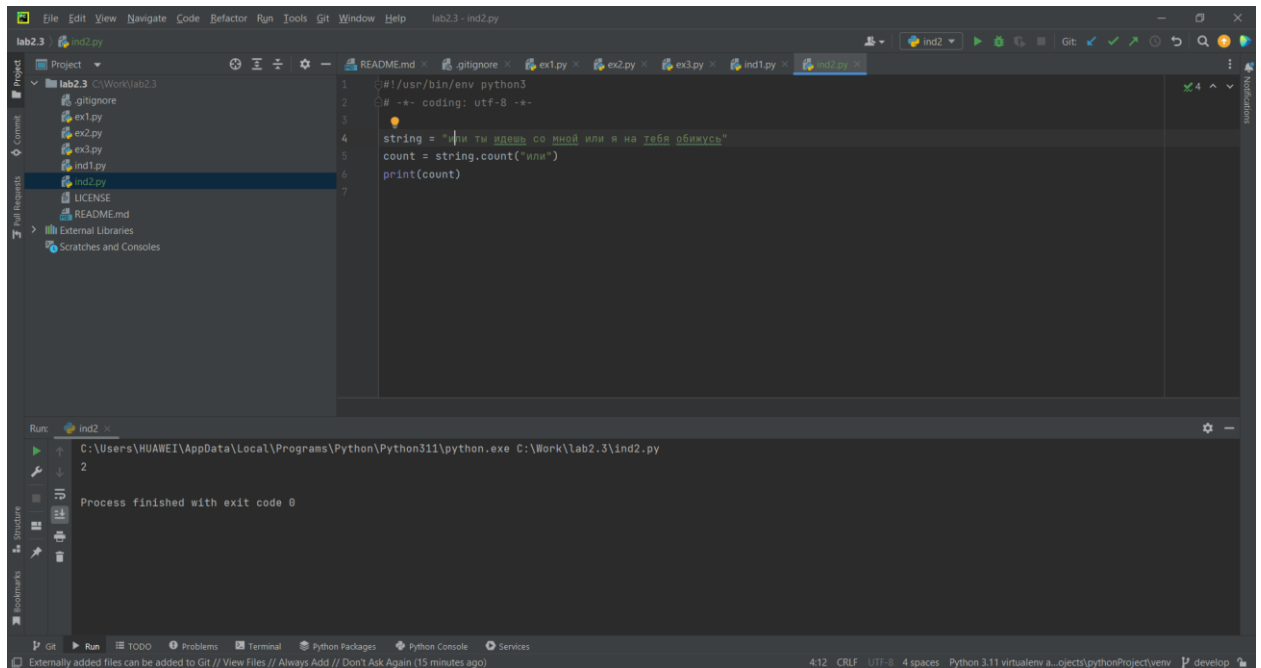


Рисунок 8 – Программа и ее результат

8. Индивидуального задание №3.

Создал новый файл под названием *ind3.py*

Вариант 14.

Условие задания:

14. Дано слово. Переставить его первую букву на место последней. При этом вторую, третью, ..., последнюю буквы сдвинуть влево на одну позицию.

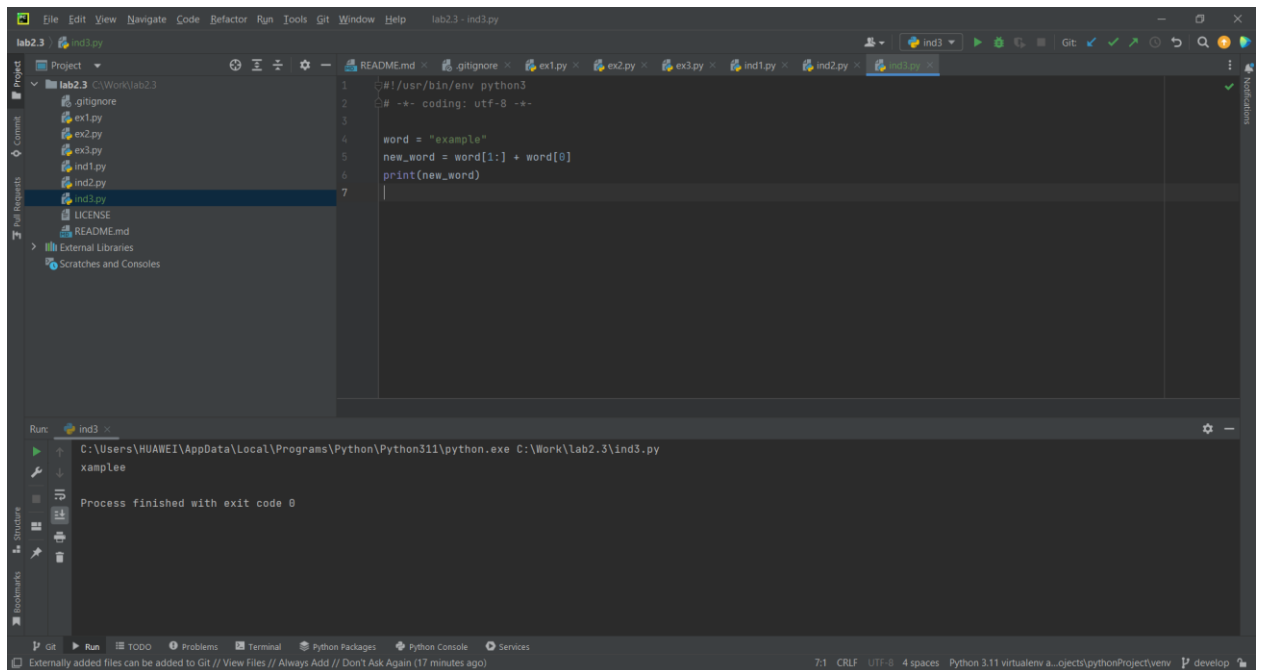


Рисунок 9 – Программа и ее результат

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

```
C:\Work\lab2.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

C:\Work\lab2.3>git merge develop
Merge made by the 'ort' strategy.
 ex1.py | 8 ++++++
 ex2.py | 15 ++++++
 ex3.py | 57 ++++++
 ind1.py | 9 ++++++
 ind2.py | 6 ++++++
 ind3.py | 6 ++++++
 6 files changed, 101 insertions(+)
 create mode 100644 ex1.py
 create mode 100644 ex2.py
 create mode 100644 ex3.py
 create mode 100644 ind1.py
 create mode 100644 ind2.py
 create mode 100644 ind3.py

C:\Work\lab2.3>
```

Рисунок 10 – Слияние веток

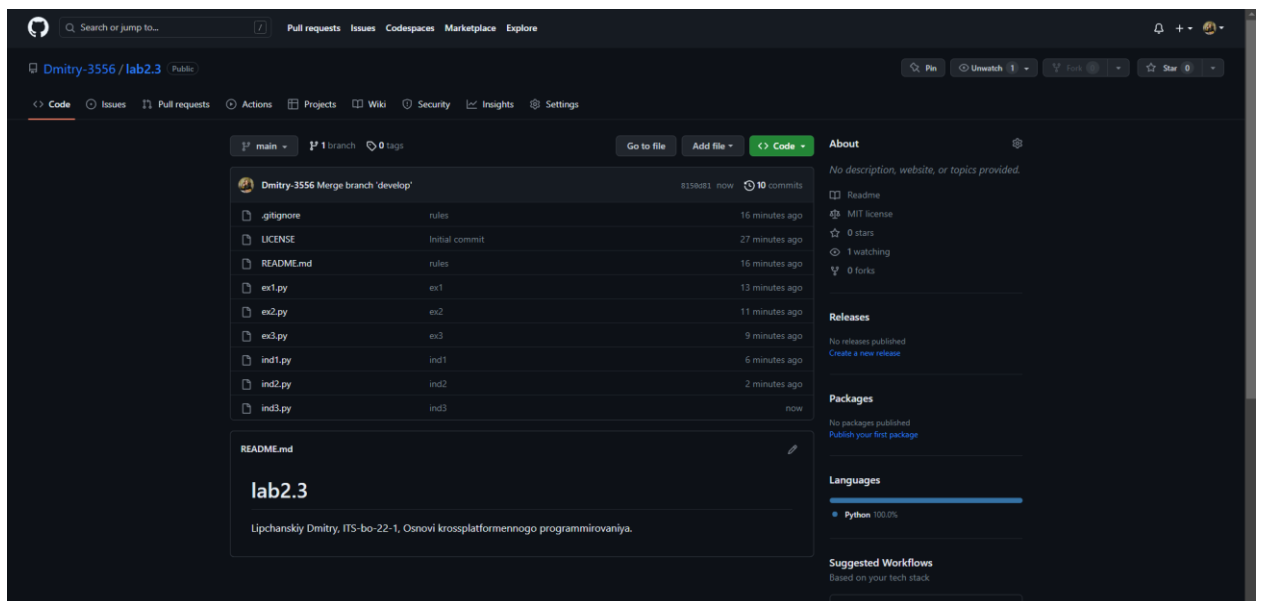


Рисунок 11 – Удаленный сервер

Ссылка: <https://github.com/Dmitry-3556/lab2.3>

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.