

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 №2.3**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:  
Худяков Максим Дмитриевич  
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., доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

## РАБОТА СО СТРОКАМИ В ЯЗЫКЕ PYTHON

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

### Ход работы:

**Задание 1.** Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.

```
C:\GIT>git clone https://github.com/maxim-hoodyakov/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.
```

Рисунок 1. Клонирование репозитория

**Задание 2.** Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\GIT\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:/GIT/lab2.3/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

**Задание 3.** Создал проект PyCharm в папке репозитория. Приступил к работе с примером №1. Добавил новый файл primer1.py.

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

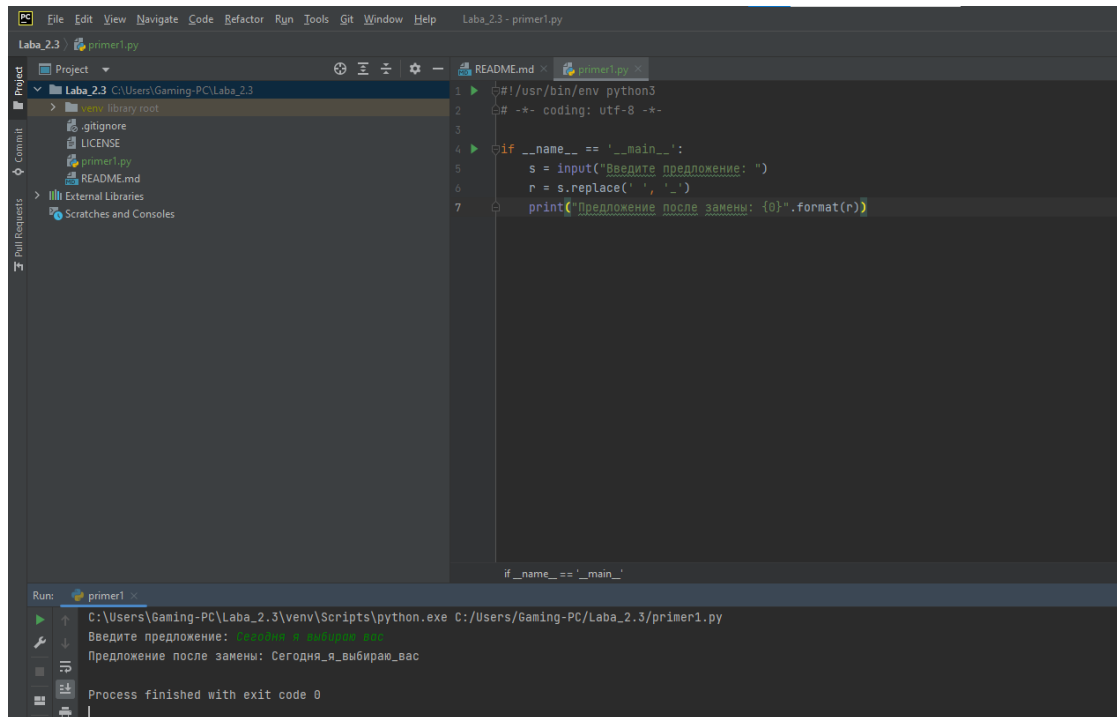


Рисунок 3. Реализация первого примера

**Задание 4.** Создал новый файл под названием primer2.py. Приступил к работе с примером №2.

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

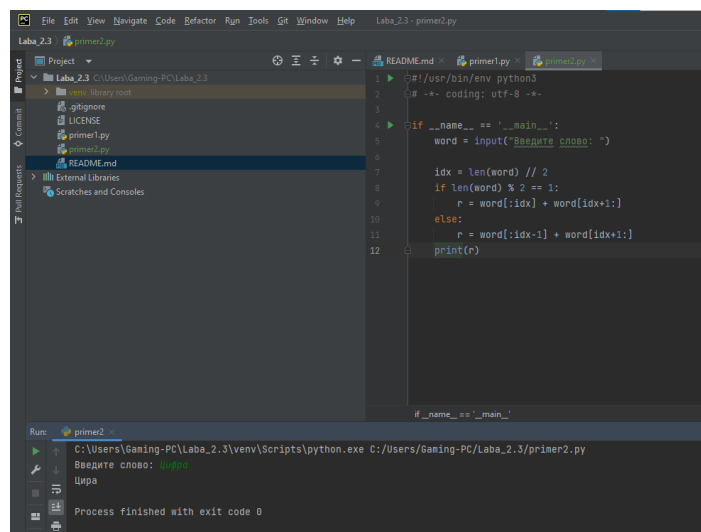


Рисунок 4. Реализация второго примера

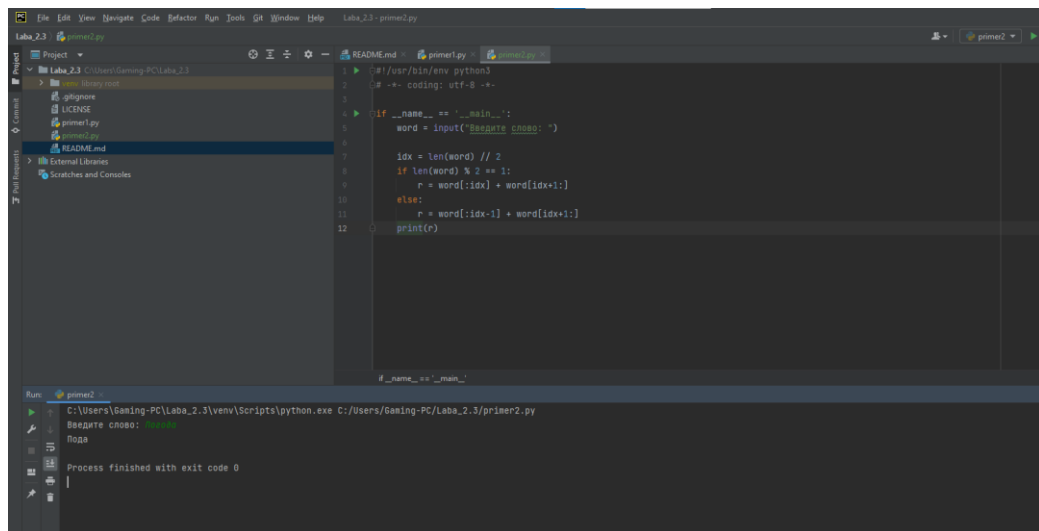


Рисунок 5. Второй пример с другими данными с клавиатуры

**Задание 4.** Создал новый файл под названием primer3.py. Приступил к работе 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.

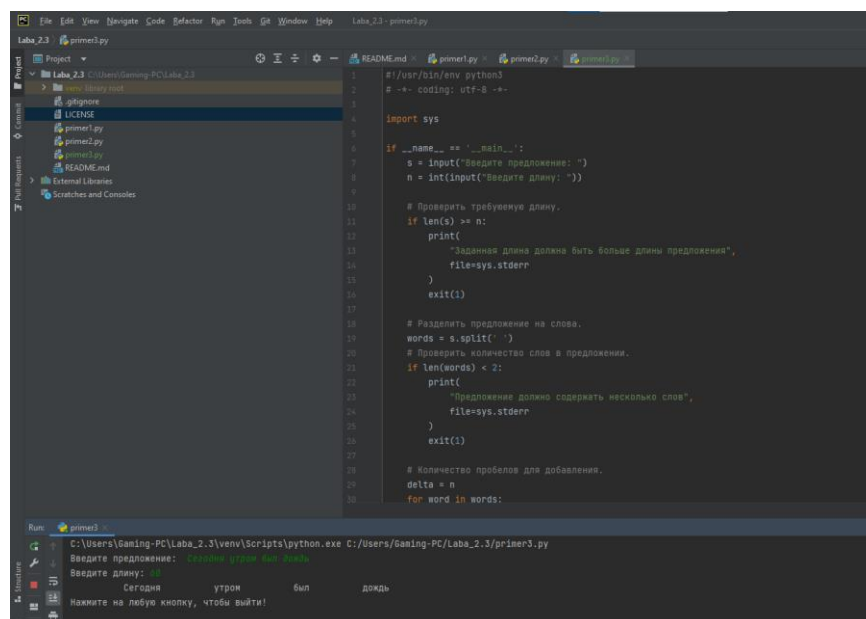


Рисунок 6. Реализация третьего примера

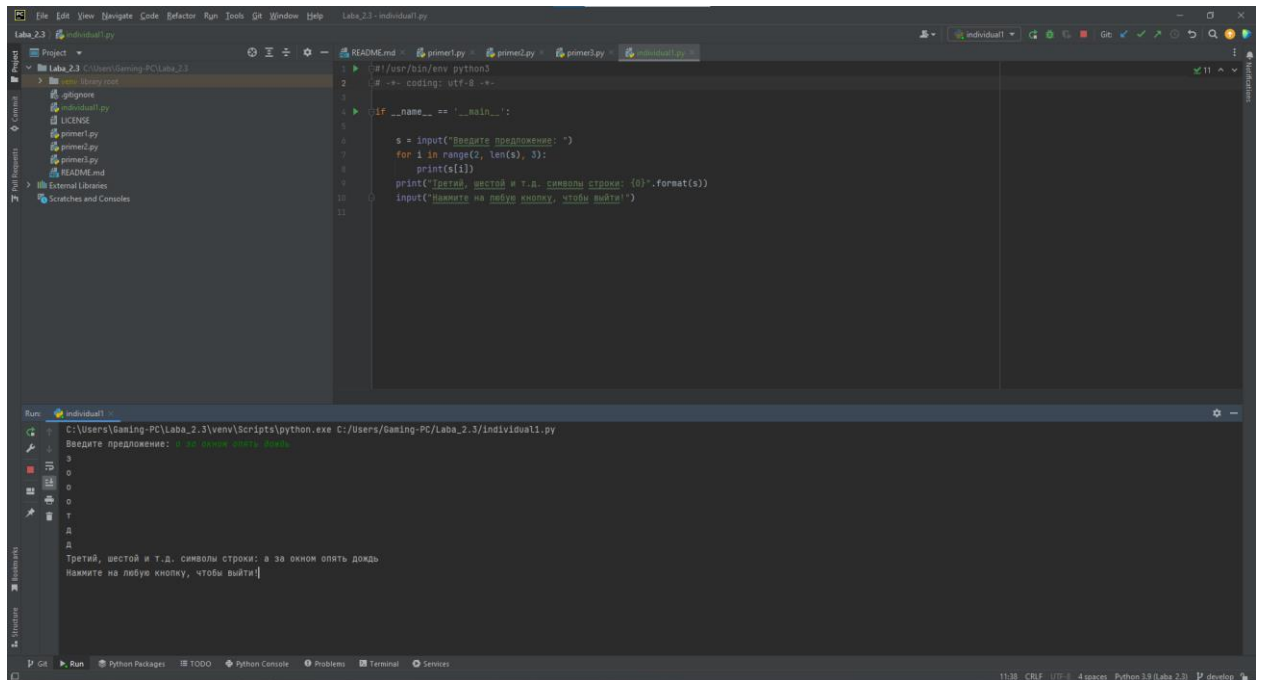
## Задание 5.

### Индивидуальное задание 1

#### Вариант 9

Создал новый файл под названием individual1.py

**Условие задания:** дано предложение. Вывести «столбиком» его третий, шестой и т. д. символы.



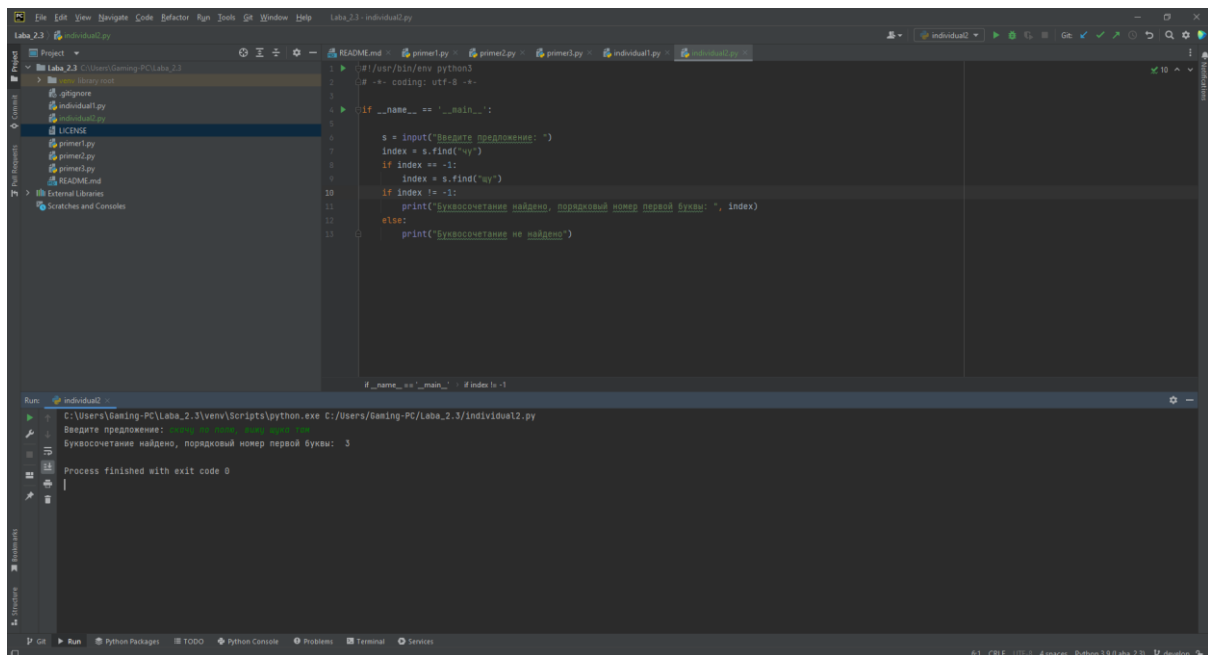


Рисунок 8. Программа выполнения ИДЗ 2

## Индивидуальное задание 3

### Вариант 9

Создал новый файл под названием individual3.py

**Условие задания:** дано слово, оканчивающееся символом «.». Составить программу, которая вставляет некоторую заданную букву после буквы с заданным номером.

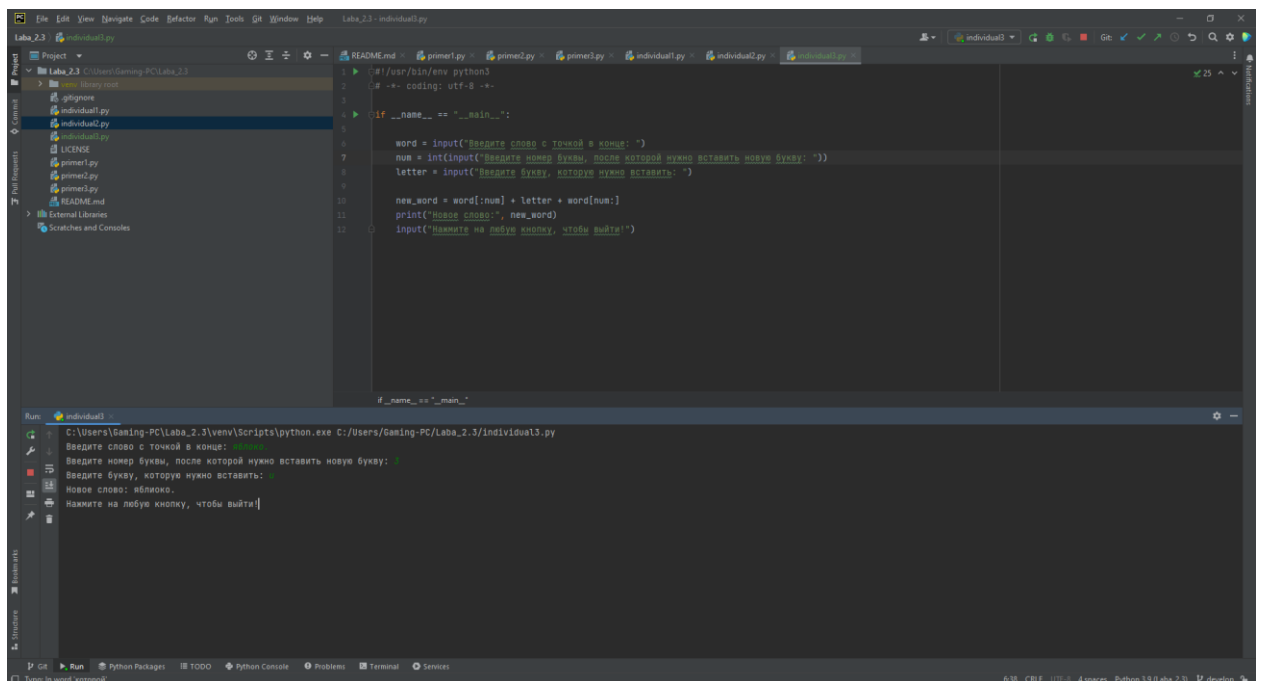


Рисунок 9. Программа выполнения ИДЗ 3

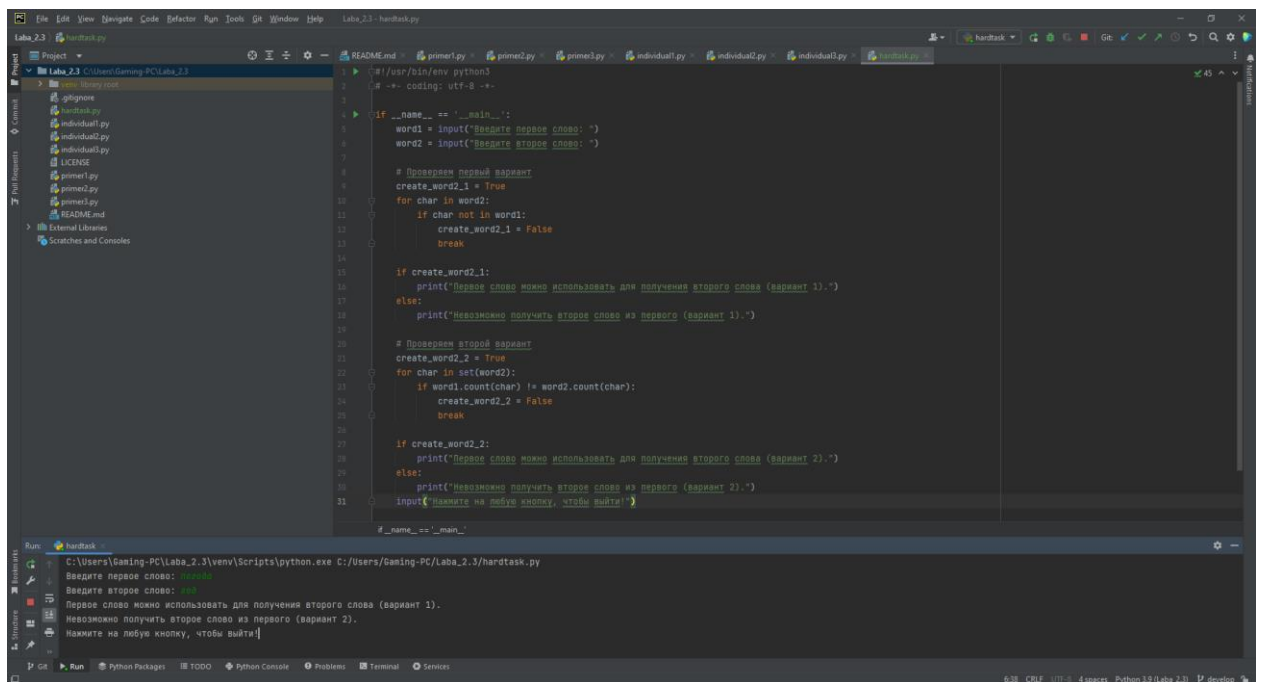
## Индивидуальное задание повышенной сложности

### Вариант 9

Создал новый файл под названием `hardtask.py`

**Условие задания:** даны два слова. Определить, можно ли из букв первого из них получить второе. Рассмотреть два варианта:

1. Повторяющиеся буквы второго слова могут в первом слове не повторяться;
2. Каждая буква второго слова должна входить в первое слово столько же раз, сколько и во второе.



```
1 # coding: utf-8 -*-
2
3 if __name__ == '__main__':
4     word1 = input("Введите первое слово: ")
5     word2 = input("Введите второе слово: ")
6
7     # Проверяем первый вариант
8     create_word2_1 = True
9     for char in word2:
10         if char not in word1:
11             create_word2_1 = False
12             break
13
14     if create_word2_1:
15         print("Первое слово можно использовать для получения второго слова (вариант 1).")
16     else:
17         print("Невозможно получить второе слово из первого (вариант 1).")
18
19     # Проверяем второй вариант
20     create_word2_2 = True
21     for char in set(word2):
22         if word1.count(char) != word2.count(char):
23             create_word2_2 = False
24             break
25
26     if create_word2_2:
27         print("Первое слово можно использовать для получения второго слова (вариант 2).")
28     else:
29         print("Невозможно получить второе слово из первого (вариант 2).")
30     input("Нажмите на любую кнопку, чтобы выйти!")
31
32 if __name__ == '__main__':
```

The screenshot shows a code editor with a file named `hardtask.py`. The code implements two algorithms to check if one word can be formed from the letters of another. The first algorithm (Variant 1) checks if all characters of the second word are present in the first word. The second algorithm (Variant 2) checks if the frequency of each character in the second word is less than or equal to its frequency in the first word. The code uses `input()` for user input and `print()` for output. The `if __name__ == '__main__':` guard is used to ensure the code runs only when the file is executed directly.

Рисунок 10. Программа выполнения задания повышенной сложности

### Задание 6.

После выполнения работы на ветке `develop`, слил ее с веткой `main` и отправил изменения на удаленный сервер.

```

C:\GIT\lab2.3>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\GIT\lab2.3>git branch
  develop
* main

C:\GIT\lab2.3>git merge develop
Updating 9753cbf..00fa62d
Fast-forward
 .idea/.gitignore           | 8 ++++
 .idea/inspectionProfiles/Project_Default.xml | 6 +++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++
 .idea/lab2.3.iml          | 10 +++++
 .idea/misc.xml             | 4 ++
 .idea/modules.xml          | 8 ++++
 .idea/vcs.xml              | 6 +++
 hardtask.py                | 30 ++++++
 individual1.py             | 8 ++++
 individual2.py             | 14 ++++++
 individual3.py             | 11 +++++
 primer1.py                 | 7 ++++
 primer2.py                 | 12 ++++++
 primer3.py                 | 56 ++++++
 14 files changed, 186 insertions(+)

```

Рисунок 11. Слияние ветки develop и main

Ссылка: <https://github.com/maxim-hoodyakov/lab2.3>

### Ответы на 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. Fстрока может быть отформатирована почти так же, как метод `str.format ()`.

#### **13. Как найти подстроку в заданной части строки?**

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.

#### **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 `islower()`. 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:** приобрел навыки по работе со строками при написании программ с помощью языка программирования Python3.