

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Работа со строками в языке Python»**

**Отчет по лабораторной работе № 2.3  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

**Create a new repository**

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

**Owner \*** Yana-Kh / **Repository name \*** Lab-6-OPJ ✓

Great repository names are short and memorable. Need inspiration? How about **symmetrical-computing-machine?**

**Description (optional)**

☒ **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:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

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

License: MIT License ▼

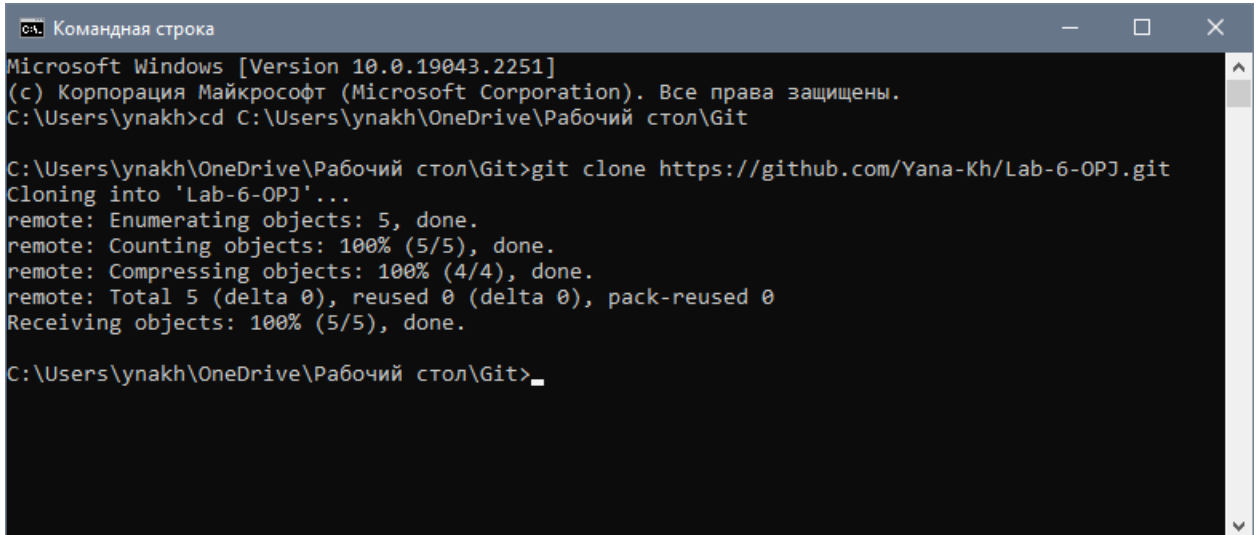
This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория

### 3. Выполните клонирование созданного репозитория.



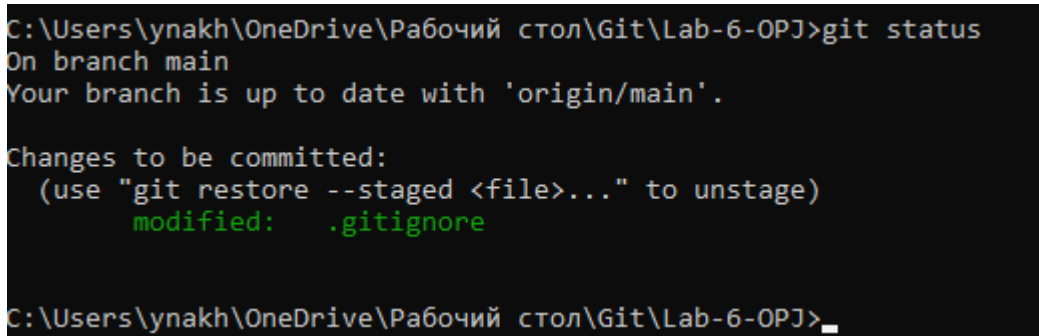
```
cmd. Командная строка
Microsoft Windows [Version 10.0.19043.2251]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git

C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-6-OPJ.git
Cloning into 'Lab-6-OPJ'...
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\ynakh\OneDrive\Рабочий стол\Git>
```

Рисунок 2 – Клонирование репозитория

### 4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



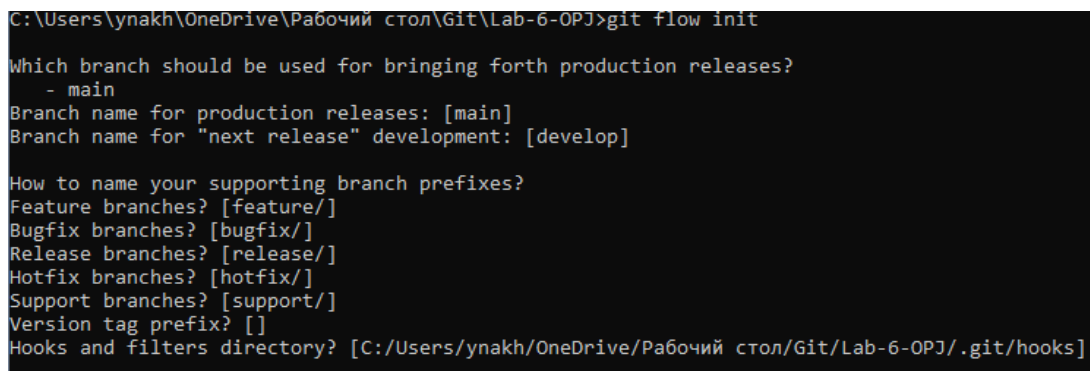
```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-6-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-6-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

### 5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-6-OPJ>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:/Users/ynakh/OneDrive/Рабочий стол/Git/Lab-6-OPJ/.git/hooks]
```

Рисунок 4 – организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

Рисунок 5 – Создание проекта PyCharm в папке репозитория

7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

Пример 1. Дано предложение. Все пробелы в нем заменить символом «  ».

Код:

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

if __name__ == '__main__':
    s = input("Введите предложение: ")
    r = s.replace(' ', '_')
    print(f"Предложение после замены: {r}")
```

```
"C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-6-OPJ\venv
Введите предложение: Привет всем ПИЖам!
Предложение после замены: Привет_всем_ПИЖам!

Process finished with exit code 0
```

### Рисунок 6 – Результат работы программы

```
Введите предложение: Какой красивый закат !
Предложение после замены: Какой_красивый_закат__!
Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

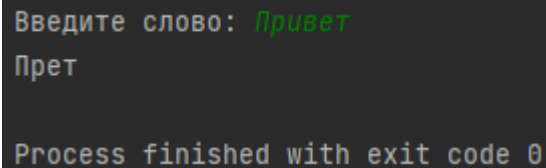
Пример 2. Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.

Код:

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

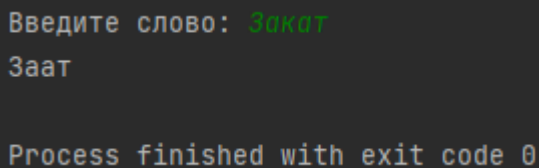
if __name__ == '__main__':
    word = input("Введите слово: ")

    idx = len(word) // 2
    if len(word) % 2 == 1:
        # Длина слова нечетная.
        r = word[:idx] + word[idx+1:]
    else:
        # Длина слова четная.
        r = word[:idx-1] + word[idx+1:]
    print(r)
```



```
Введите слово: Привет
Прет
Process finished with exit code 0
```

Рисунок 8 – Результат работы программы



```
Введите слово: Закат
Заат
Process finished with exit code 0
```

Рисунок 9 – Результат работы программы

Пример 3. 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.

Код:

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

```

import sys

if __name__ == '__main__':
    s = input("Введите предложение: ")
    n = int(input("Введите длину: "))

    # Проверить требуемую длину.
    if len(s) >= n:
        print(
            "Заданная длина должна быть больше длины предложения",
            file=sys.stderr
        )
        exit(1)

    # Разделить предложение на слова.
    words = s.split(' ')
    # Проверить количество слов в предложении.
    if len(words) < 2:
        print(
            "Предложение должно содержать несколько слов",
            file=sys.stderr
        )
        exit(1)

    # Количество пробелов для добавления.
    delta = n
    for word in words:
        delta -= len(word)

    # Количество пробелов на каждое слово.
    w, r = delta // (len(words) - 1), delta % (len(words) - 1)

    # Сформировать список для хранения слов и пробелов.
    lst = []

    # Пронумеровать все слова в списке и перебрать их.
    for i, word in enumerate(words):
        lst.append(word)

        # Если слово не является последним, добавить пробелы.
        if i < len(words) - 1:
            # Определить количество пробелов.
            width = w
            if r > 0:
                width += 1
                r -= 1

            # Добавить заданное количество пробелов в список.
            if width > 0:
                lst.append(' ' * width)

    # Вывести новое предложение, объединив все элементы списка lst.
    print(''.join(lst))

```

```
Введите предложение: Привет мир
Введите длину: 18
Привет      мир

Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

```
Введите предложение: Привет
Введите длину: 7
Предложение должно содержать несколько слов

Process finished with exit code 1
```

Рисунок 11 – Результат работы программы

8. Выполните индивидуальные задания, согласно своему варианту. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

#### Задание 1

32. Дано предложение. Напечатать все его символы, предшествующие первой запятой. Рассмотреть два случая:

- известно, что в предложении запятые имеются;
- в предложении запятых может не быть.

Код:

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

if __name__ == '__main__':
    s = input("Введите предложение: ")
    for i in s:
        if i == ",":
            print("Появилась запятая")
            exit(1)
        else:
            print(i)
    print("Запятых нет")
```

```
Введите предложение: Привет, солнце!  
П  
р  
и  
в  
е  
т  
Появилась запятая  
  
Process finished with exit code 1
```

Рисунок 12 – Результат работы программы

```
Введите предложение: Пароль  
П  
а  
р  
о  
л  
ь  
Запятых нет  
  
Process finished with exit code 0
```

Рисунок 13 – Результат работы программы

## Задание 2

4. Дана последовательность символов, в начале которой имеется некоторое количество одинаковых символов. Определить это количество. Рассмотреть два случая:

- известно, что не все символы последовательности одинаковые;
- все символы последовательности могут быть одинаковыми.

Код:

```
if __name__ == '__main__':  
    s = input("Введите предложение: ")  
    n = 0  
    for i in s:  
        if i == s[0]:  
            n += 1  
        else:  
            print("Последовательность прервана")
```



```
print(f"Число вхождений {n}")
exit(1)
print("Последовательность состоит только из одного символа")
print(f"Число вхождений {n}")
```

```
Введите предложение: fffpg
Последовательность прервана
Число вхождений 3

Process finished with exit code 1
```

Рисунок 14 – Результат работы программы

```
Введите предложение: uuuuuu
Последовательность состоит только из одного символа
Число вхождений 5

Process finished with exit code 0
```

Рисунок 15 – Результат работы программы

### Задание 3

32. Дан текст, представляющий собой десятичную запись целого числа. Вычислить сумму цифр этого числа.

Код:

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

import sys

if __name__ == '__main__':
    s = input("Введите число: ")
    n = 0
    if s.isdigit():
        for i in s:
            n += int(i)
    else:
        print("Обнаружены не числовые символы", file=sys.stderr)
        exit(1)
    print(f"Сумма цифр числа {s}: {n}")
```

```
Введите число: 6343
Сумма цифр числа 6343: 16

Process finished with exit code 0
```

Рисунок 16 – Результат работы программы

```
Введите число: 456g
Обнаружены не числовые символы

Process finished with exit code 1
```

Рисунок 17 – Результат работы программы

### Задание повышенной сложности

32. Дан текст. Проверить, правильно ли в нем расставлены круглые скобки (т. е. находится ли справа от каждой открывающей скобки соответствующая ей закрывающая скобка, а слева от каждой закрывающей – соответствующая ей открывающая). Предполагается, что внутри каждой пары скобок нет других скобок.

- Ответом должны служить слова да или нет.
- В случае неправильности расстановки скобок: если имеются лишние правые (закрывающие) скобки, то выдать сообщение с указанием позиции первой такой скобки; если имеются лишние левые (открывающие) скобки, то выдать сообщение с указанием количества таких скобок. Если скобки расставлены правильно, то сообщить об этом.

Код:

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

if __name__ == '__main__':
    s = input("Введите строку: ")
    n, m = 0, 0
    index = 0
    print("Правильно ли расставлены круглые скобки?")
    for i in s:
        index += 1
        if i == "(":
            n += 1
        if i == ")":
            m += 1
        if n < m:
            print("нет")
            print(f"Обнаружена лишняя правая скобка: {index}")
            exit(1)
    if n > m:
        print(f"нет\nОбнаружено {n-m} лишних левых скобок")
```

```
if n == m:  
    print("да\nСкобки расставлены правильно")
```

```
Введите строку: (())  
Правильно ли расставлены круглые скобки?  
да  
Скобки расставлены правильно  
  
Process finished with exit code 0
```

Рисунок 18 – Результат работы программы

```
Введите строку: (((  
Правильно ли расставлены круглые скобки?  
нет  
Обнаружено 3 лишних левых скобок  
  
Process finished with exit code 0
```

Рисунок 19 – Результат работы программы

```
Введите строку: ()fe(fef)f  
Правильно ли расставлены круглые скобки?  
нет  
Обнаружена лишняя правая скобка: 11  
  
Process finished with exit code 1  
|
```

Рисунок 20 – Результат работы программы

## Вопросы для защиты работы

### 1. Что такое строки в языке Python?

Строки в языке Python – это упорядоченные последовательности символов, используемые для хранения и представления текстовой информации.

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

Существует несколько литералов строк – это строки в апострофах(') и в кавычках("), что по сути одно и то же, и строки в тройных апострофах.

```
S = 'привет'
```

```
S = " привет"
```

Причина наличия первых двух вариантов в том, чтобы позволить вставлять в литералы строк символы кавычек или апострофов, не используя экранирование.

Последний способ позволяет для записывать многострочные блоки текста. Например:

```
>>> c = """это очень большая  
... строка, многострочный  
... блок текста"""
```

### 3. Какие операции и функции существуют для строк?

Оператор сложения строк + – соединяет несколько строк в одну, например:

```
S = «Я люблю »
```

```
B = «роллы!»
```

```
S + B = Я люблю роллы!
```

Оператор умножения строк \* – повторяет последовательность символов заданное количество раз, например:

```
S = «ма»; S * 2 = мама
```

Оператор принадлежности подстроки `in` – возвращает `true` или `false`, если подстрока входит в строку или нет соответственно.

Функция `chr()` – Преобразует целое число в символ (ASCII or Unicode)

Функция `ord()` – Преобразует символ в целое число (ASCII or Unicode)

Функция `len()` – Возвращает длину строки

Функция `str()` – Изменяет тип объекта на `string`

#### 4. Как осуществляется индексирование строк?

Индексация строк начинается с нуля. Обращение к символу по индексу осуществляется путем записи `string[i]`, где `string` – имя строки, `i` – индекс, соответствующий порядковому номеру символа.

Попытка обращения по индексу большему чем длина строки приводит к ошибке.

Индексы могут быть как положительными, так и отрицательными (`-1` = последний символ, `-2` = предпоследний и т.д.)

Нет индекса, который применим к пустой строке.

#### 5. Как осуществляется работа со срезами для строк?

Срез – это извлечение подстроки из строки.

Если `s` это строка, выражение формы `s[m:n:p]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая последнюю позицию. Последняя переменная означает шаг, означающий сколько символов следует пропустить после извлечения каждого символа в срезе.

Если не указан `m` – срез с начала, т.е. начиная с нулевого элемента.

Если не указан `n` – срез до конца строки

Если не указан `p` – шаг не осуществляется

Если `p < 0` – совершается шаг в обратном направлении

#### 6. Почему строки Python относятся к неизменяемому типу данных?

Фактически работа с неизменяемыми типами данных осуществляется следующим образом: сначала создается объект с определенным значением, берется его адрес и присваивается переменной. При попытке изменить заданное ранее значение создается новый объект, в котором хранится новое значение, берется его адрес и этот адрес присваивается переменной.

В Python нельзя изменить некоторый одиночный символ в строке, например, через `s[2] = 'z'`, не говоря уже о том, чтобы вставить символ внутрь строки.

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

Для проверки того, что каждое слово начинается с заглавной буквы необходимо воспользоваться командой `string.istitle()`, которое возвращает `True` когда `s` не пустая строка и первый алфавитный символ каждого слова в верхнем регистре, а все остальные буквенные символы в каждом слове строчные.

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

Для этого можно воспользоваться `s.count()` возвращает количество точных вхождений подстроки в `s`.

9. Как найти индекс первого вхождения подстроки в строку?

Для этого необходимо воспользоваться командой `string.find(<sub>[, <start>[, <end>]])`, которая возвращает первый индекс в `s` который соответствует началу строки `<sub>` или `-1`, если указанная подстановка не найдена.

10. Как подсчитать количество символов в строке?

Для этого необходимо воспользоваться командой `len() + 1`

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

Для этого можно воспользоваться `string.count(<sub>[, <start> [, <end> ]])`

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

f-строки – это один из способов форматирования текста в Python. Одной из отличительных особенностей f-строк, является интерполяция переменной. Вы можете указать имя переменной непосредственно в f-строковом литерале (f'string'), и python заменит имя соответствующим значением.

13. Как найти подстроку в заданной части строки?

Для этого необходимо воспользоваться командой `string.find(<sub>[, <start>[, <end>]])`, указав в start и end индексы первого и последнего символа (последнего +1, т.к. он не включается) желаемого интервала.

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

Вводится строка, места, куда необходимо вставить переменную обозначаются как {[индекс переменной]}, за этим следует «.format()», в скобках – переменные через запятую. Если переменная одна {} можно оставить пустыми.

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

Для этого необходимо воспользоваться командой `string.isdigit()`

16. Как разделить строку по заданному символу?

Для этого необходимо воспользоваться командой `split()` (в скобках символ или последовательность символов)

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

Для этого необходимо воспользоваться командой `string.islower()`

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

Для этого необходимо воспользоваться вышеупомянутой командой для первого символа строки `string[0].islower()`

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

Нет

20. Как «перевернуть» строку?

Можно воспользоваться срезом: `имя_строки[::-1]`

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

Можно воспользоваться `'-'.join([список])`

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

`s.upper()` – к верхнему

`s.lower()` – к нижнему

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

С помощью вышеуказанных команд с обращением к индексам `[0] + [1:-1] (срез середины) + [-1]`

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

Можно воспользоваться `s.isupper()`



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

Он разделяет строки по символам разрыва строки, таким как «\n», «\r» и другие.

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

Можно воспользоваться `s.replace('что заменить', 'на что заменить')`

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

`s.startswith('...')` – начинается

`s.endswith('...')` – заканчивается

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

Можно воспользоваться `s.isspace()`

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

Она повторится три раза, например «лар» \* 3 = «ларларлар»

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

Можно воспользоваться `s.title()`

31. Как пользоваться методом `partition()` ?

Делит строку на основе разделителя. Отделяет от `s` подстроку длиной от начала до первого вхождения. Возвращаемое значение представляет собой кортеж из трех частей: часть до, разделитель, часть после

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

Метод `rfind()` похож на метод `find()`, но он, в отличие от `find()`, ищет с конца.

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