

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа со строками в языке Python»

ОТЧЕТ
по лабораторной работе №6
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

09.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

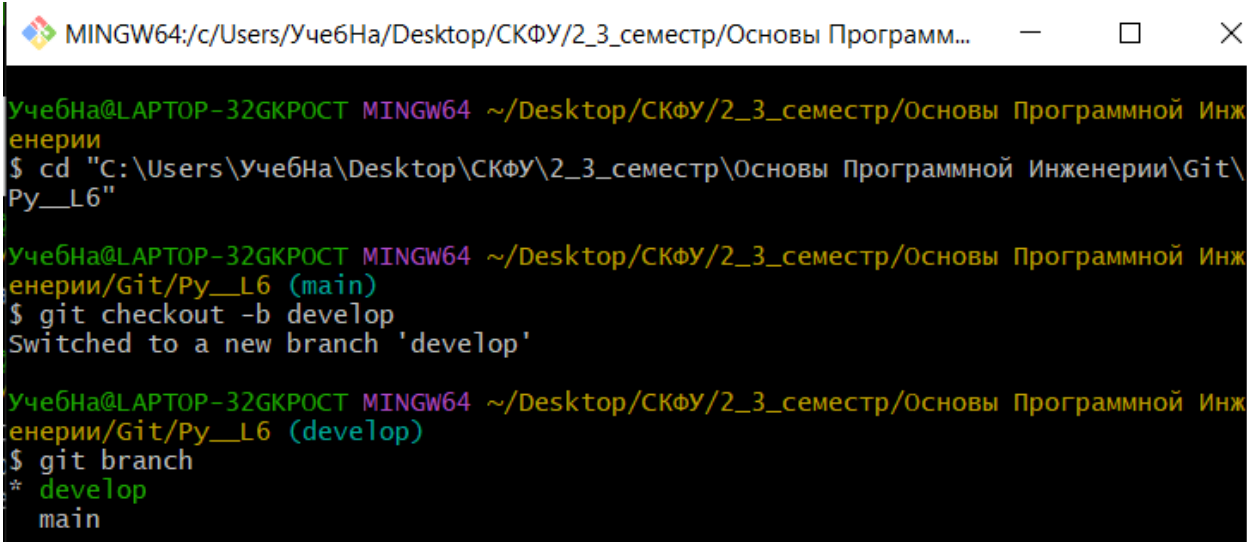
Ставрополь, 2022 г.

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

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L6.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

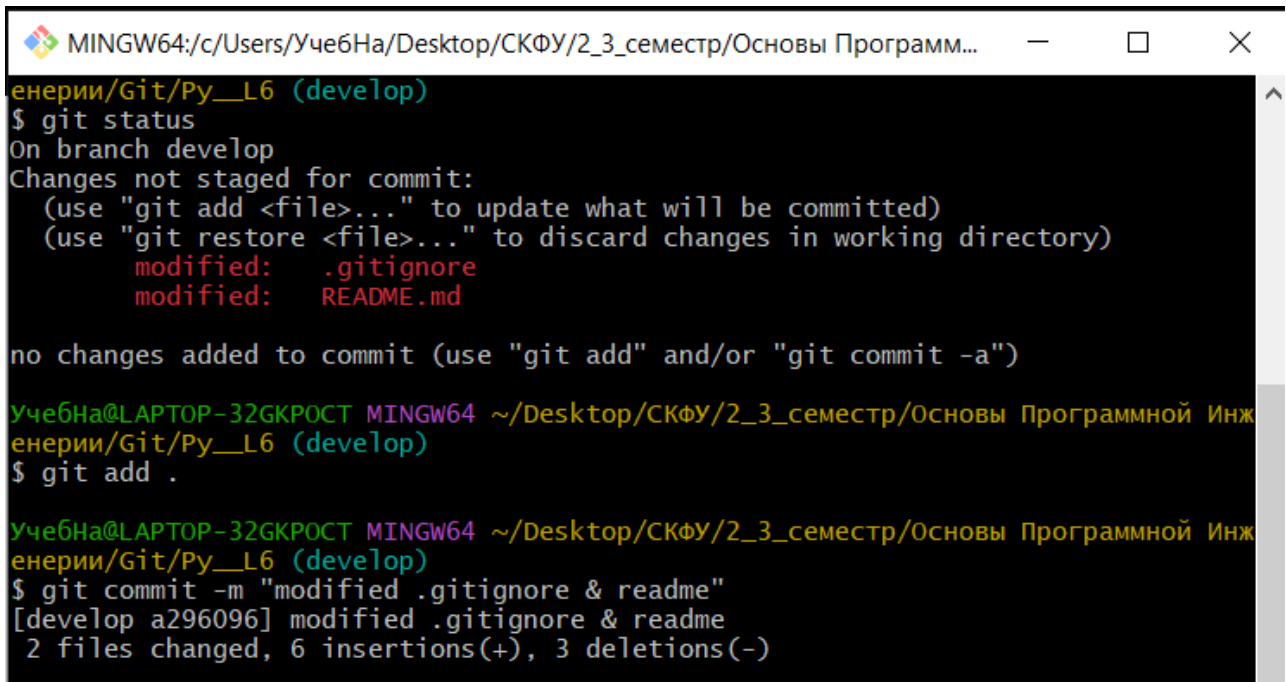


```
MINGW64:/c:/Users/Учебна/Desktop/СКФУ/2_3_семестр/Основы Программ...
Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии
$ cd "C:\Users\Учебна\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py__L6"

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L6 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L6 (develop)
$ git branch
* develop
  main
```

Рисунок 6.1 – Создание ветки develop



```
MINGW64:/c/Users/УчебНа/Desktop/СКФУ/2_3_семестр/Основы Программ...
энерии/Git/Py__L6 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
энерии/Git/Py__L6 (develop)
$ git add .

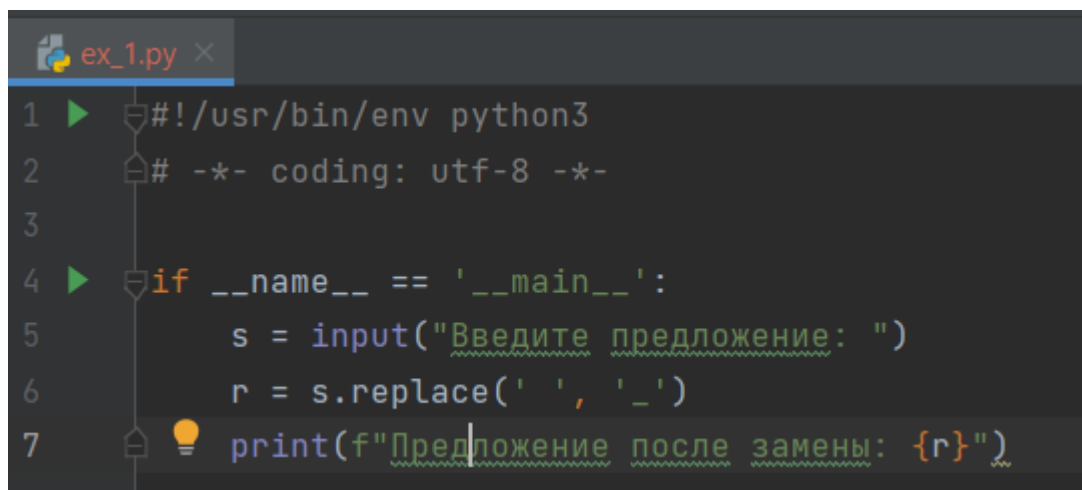
УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
энерии/Git/Py__L6 (develop)
$ git commit -m "modified .gitignore & readme"
[develop a296096] modified .gitignore & readme
2 files changed, 6 insertions(+), 3 deletions(-)
```

Рисунок 6.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.
8. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

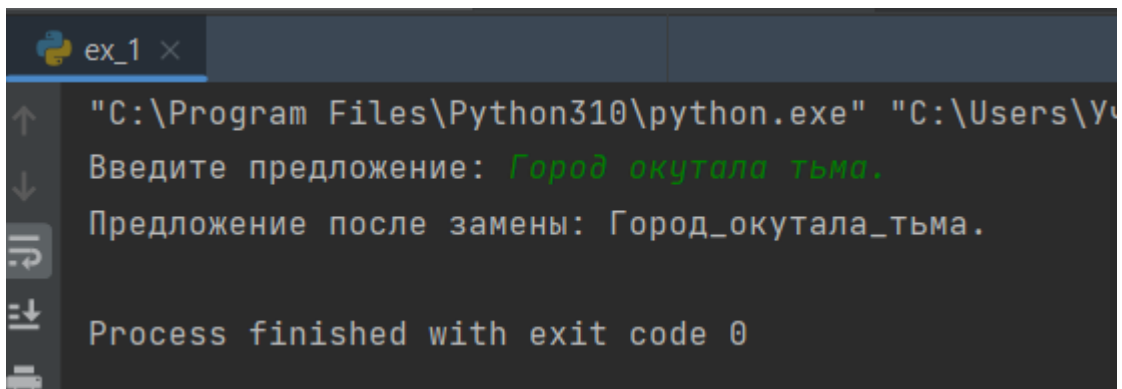
Пример 1.

Дано предложение. Все пробелы в нем заменить символом «_».



```
ex_1.py x
1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      s = input("Введите предложение: ")
6      r = s.replace(' ', '_')
7      print(f"Предложение после замены: {r}")
```

Рисунок 6.3 – Код программы-примера

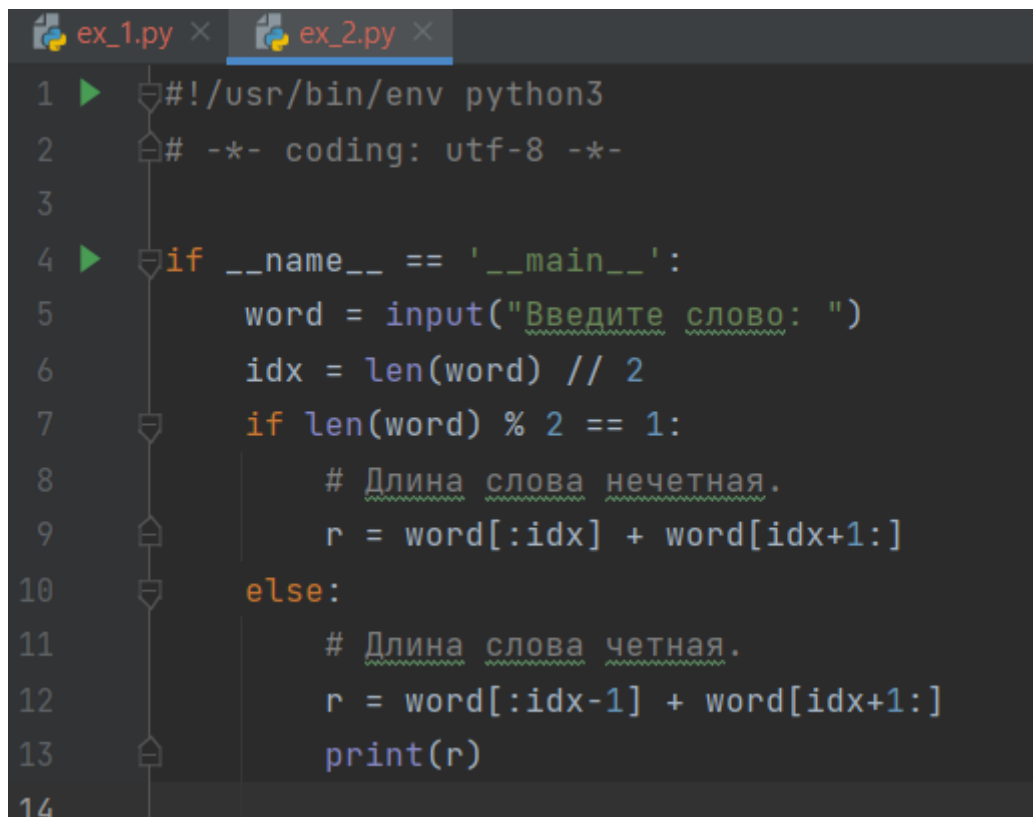


```
ex_1 x
"C:\Program Files\Python310\python.exe" "C:\Users\Y...
Введите предложение: Город окутала тьма.
Предложение после замены: Город_окутала_тьма.
Process finished with exit code 0
```

Рисунок 6.4 – Результат программы

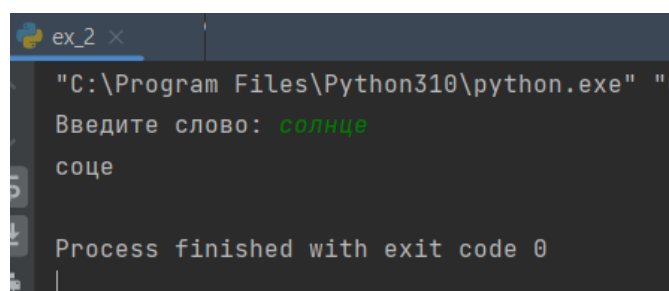
Пример 2.

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



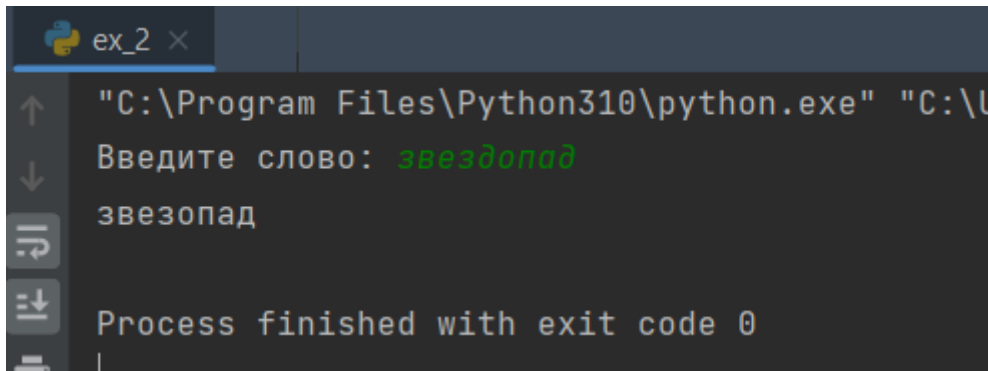
```
ex_1.py x ex_2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     word = input("Введите слово: ")
6     idx = len(word) // 2
7     if len(word) % 2 == 1:
8         # Длина слова нечетная.
9         r = word[:idx] + word[idx+1:]
10    else:
11        # Длина слова четная.
12        r = word[:idx-1] + word[idx+1:]
13        print(r)
14
```

Рисунок 6.5 – Код программы-примера



```
ex_2 x
"C:\Program Files\Python310\python.exe" "C...
Введите слово: солнце
соце
Process finished with exit code 0
```

Рисунок 6.6 – Результат программы при четной длине

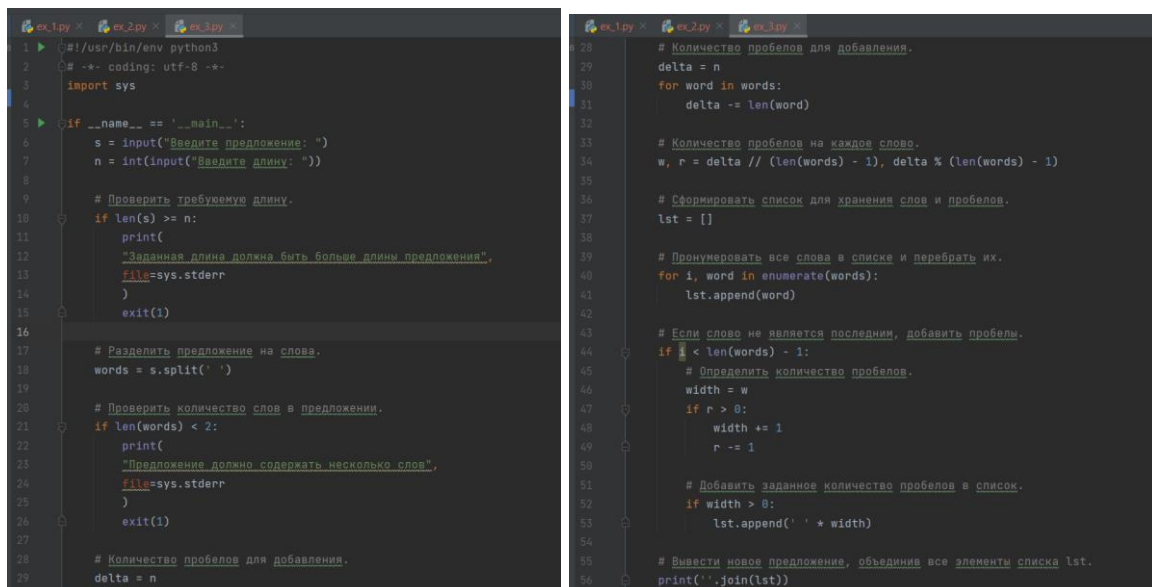


```
ex_2 x
"C:\Program Files\Python310\python.exe" "C:\U
Введите слово: звездапад
звезопад
Process finished with exit code 0
```

Рисунок 6.7 – Результат программы при нечетной длине

Пример 3.

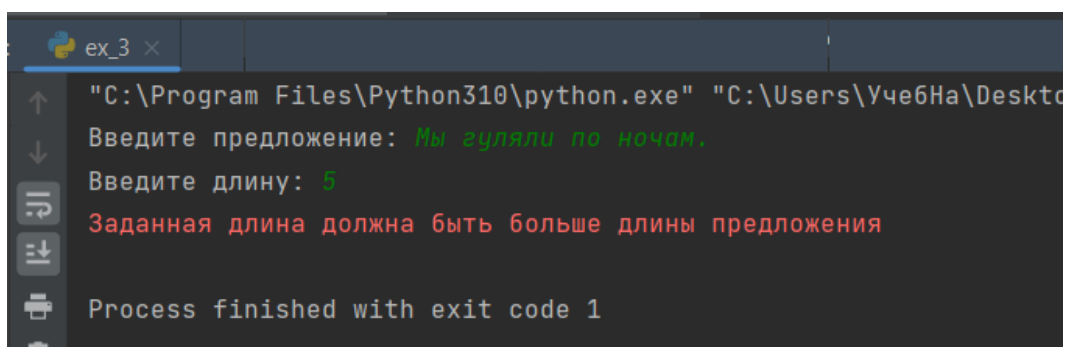
Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки стала равна заданной длине (предполагается, что требуемая длина не меньше исходной). Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными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



```
ex_1.py x ex_2.py x ex_3.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4
5 if __name__ == '__main__':
6     s = input("Введите предложение: ")
7     n = int(input("Введите длину: "))
8
9     # Проверить требуемую длину.
10    if len(s) >= n:
11        print(
12            "Заданная длина должна быть больше длины предложения",
13            file=sys.stderr
14        )
15        exit(1)
16
17    # Разделить предложение на слова.
18    words = s.split(' ')
19
20    # Проверить количество слов в предложении.
21    if len(words) < 2:
22        print(
23            "Предложение должно содержать несколько слов",
24            file=sys.stderr
25        )
26        exit(1)
27
28    # Количество пробелов для добавления.
29    delta = n
```

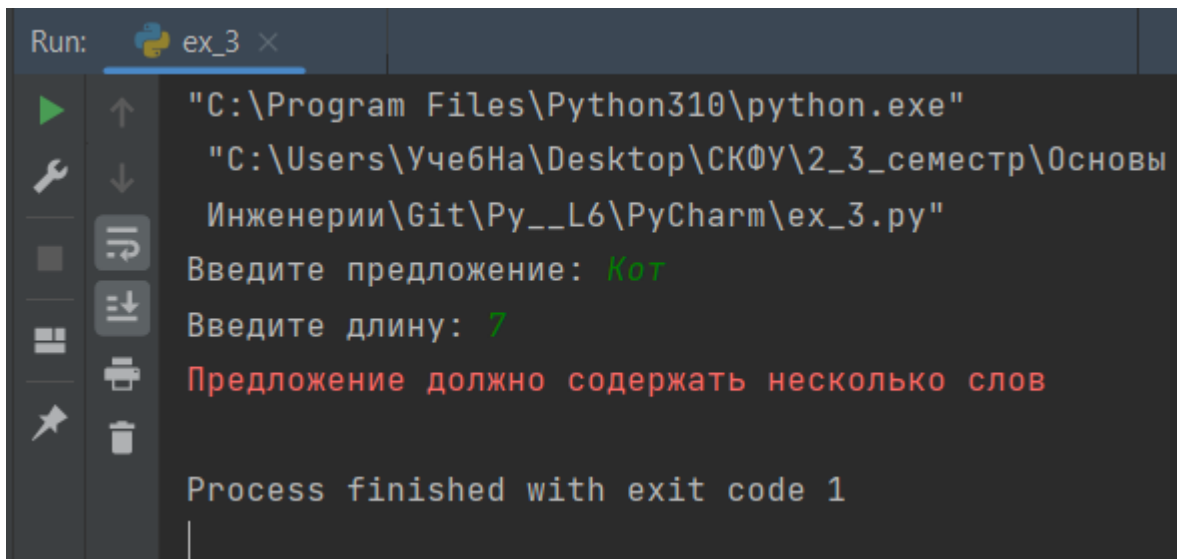
```
ex_1.py x ex_2.py x ex_3.py x
28 # Количество пробелов для добавления.
29 delta = n
30 for word in words:
31     delta -= len(word)
32
33 # Количество пробелов на каждое слово.
34 w, r = delta // (len(words) - 1), delta % (len(words) - 1)
35
36 # Сформировать список для хранения слов и пробелов.
37 lst = []
38
39 # Пронумеровать все слова в списке и перебрать их.
40 for i, word in enumerate(words):
41     lst.append(word)
42
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))
```

Рисунок 6.8 – Код программы-примера



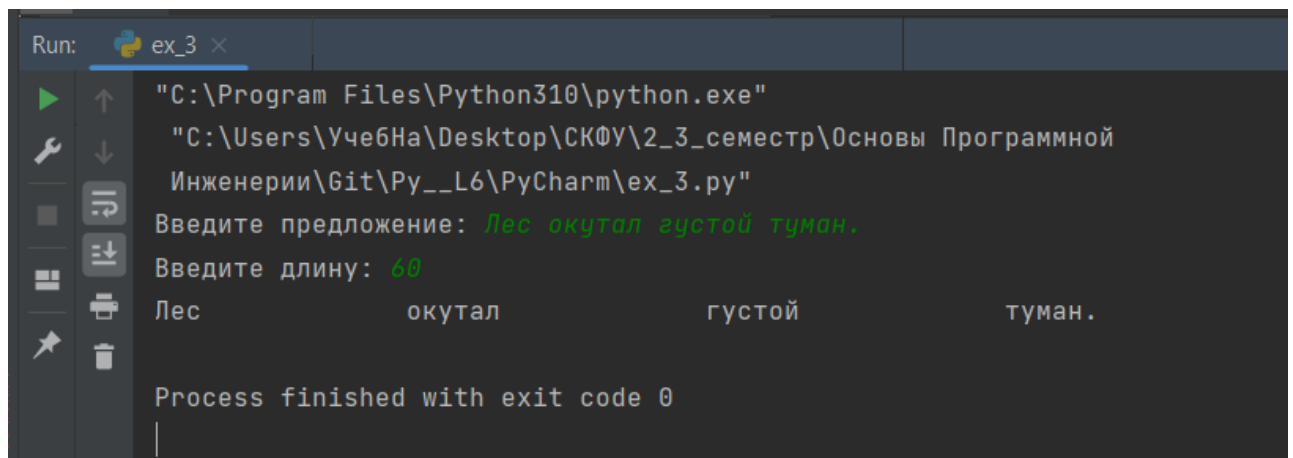
```
ex_3 x
"C:\Program Files\Python310\python.exe" "C:\Users\УчебНа\Desktop
Введите предложение: Мы гуляли по ночам.
Введите длину: 5
Заданная длина должна быть больше длины предложения
Process finished with exit code 1
```

Рисунок 6.9 – Результат программы (ошибка)



```
Run: ex_3 x
"C:\Program Files\Python310\python.exe"
"C:\Users\УчебHa\Desktop\СКФУ\2_3_семестр\Основы
Инженерии\Git\Py__L6\PyCharm\ex_3.py"
Введите предложение: Кот
Введите длину: 7
Предложение должно содержать несколько слов
Process finished with exit code 1
```

Рисунок 6.10 – Результат программы (ошибка)



```
Run: ex_3 x
"C:\Program Files\Python310\python.exe"
"C:\Users\УчебHa\Desktop\СКФУ\2_3_семестр\Основы Программной
Инженерии\Git\Py__L6\PyCharm\ex_3.py"
Введите предложение: Лес окутал густой туман.
Введите длину: 60
Лес          окутал          густой          туман.
Process finished with exit code 0
```

Рисунок 6.11 – Результат программы

9. Выполните индивидуальные задания, согласно своего варианта. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

10. Зафиксируйте сделанные изменения в репозитории.

11. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

12. Выполните слияние ветки для разработки с веткой main / master.

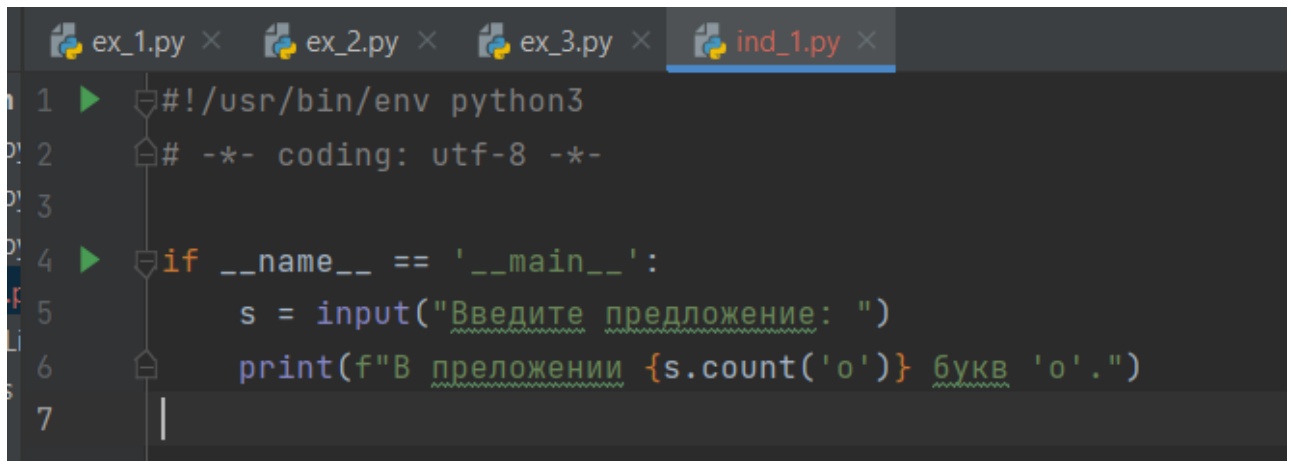
13. Отправьте сделанные изменения на сервер GitHub.

14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальные задания: (Вариант 15)

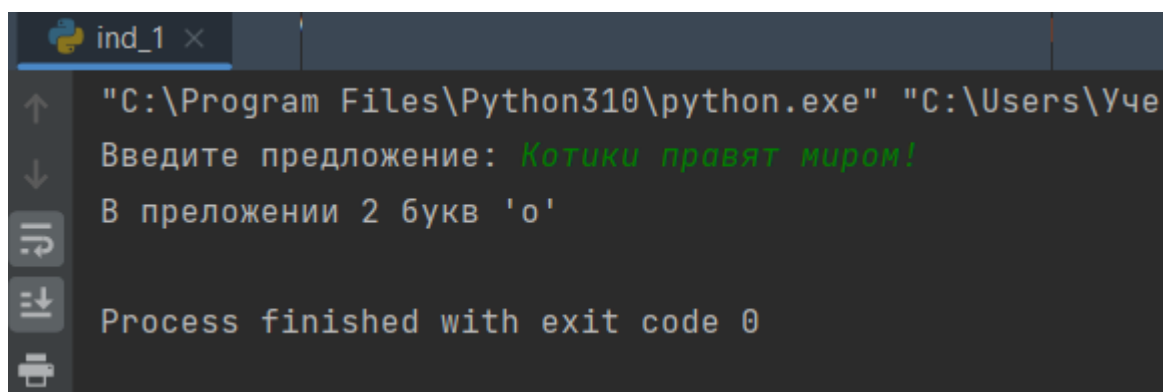
Задание 1

15. Дано предложение. Определить число букв о в нем.



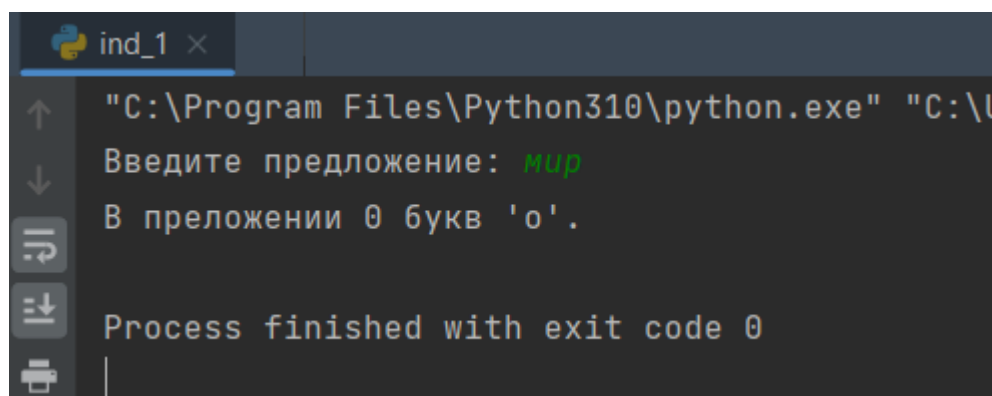
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     print(f"В предложении {s.count('o')} букв 'о'.")
7
```

Рисунок 6.12 – Код программы



```
"C:\Program Files\Python310\python.exe" "C:\Users\Уче
Введите предложение: Котики правят миром!
В предложении 2 букв 'о'
Process finished with exit code 0
```

Рисунок 6.13 – Результат программы



```
"C:\Program Files\Python310\python.exe" "C:\U
Введите предложение: мир
В предложении 0 букв 'о'.
Process finished with exit code 0
```

Рисунок 6.14 – Результат программы

Задание 2

15. Дано предложение. Все его символы, стоящие на четных местах, заменить буквой ы.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      s = input("Введите предложение: ")
6      s_list = list(s)
7
8      for i in range(len(s)):
9          if i % 2 != 0:
10             s_list[i] = 'ы'
11
12     print(f"Новая строка: {''.join(s_list)}")
13

```

Рисунок 6.15 – Код программы

```

"C:\Program Files\Python310\python.exe" "C:\Users\user\ind_2.py"
Введите предложение: кот
Новая строка: кыт

Process finished with exit code 0

```

Рисунок 6.16 – Результат программы

```

"C:\Program Files\Python310\python.exe" "C:\Users\user\ind_2.py"
Введите предложение: Мы шли по лесу.
Новая строка: Мы ылы ыылысы.

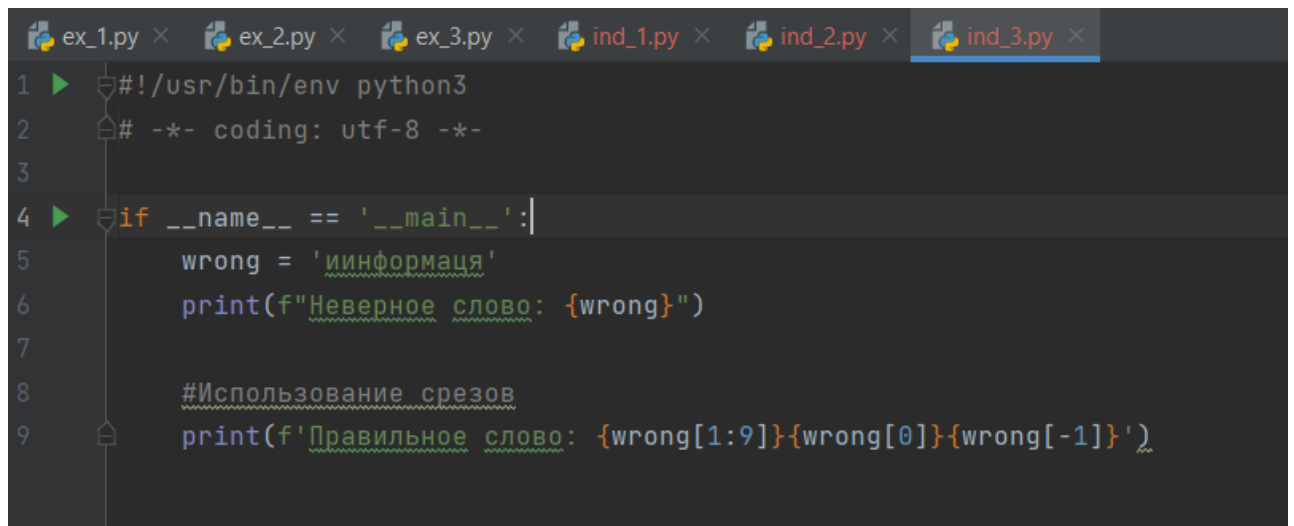
Process finished with exit code 0

```

Рисунок 6.17 – Результат программы

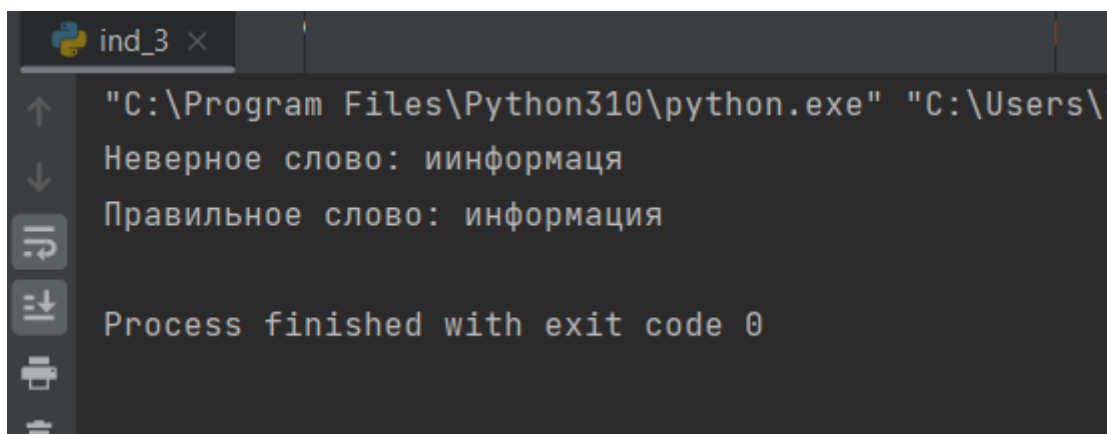
Задание 3

15. Дано ошибочно написанное слово и информация. Путем перемещения его букв получить слово информация.



```
ex_1.py x ex_2.py x ex_3.py x ind_1.py x ind_2.py x ind_3.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     wrong = 'иинформация'
6     print(f"Неверное слово: {wrong}")
7
8     #Использование срезов
9     print(f'Правильное слово: {wrong[1:9]}{wrong[0]}{wrong[-1]}')
```

Рисунок 6.18 – Код программы



```
ind_3 x
"C:\Program Files\Python310\python.exe" "C:\Users\
Неверное слово: иинформация
Правильное слово: информация
Process finished with exit code 0
```

Рисунок 6.19 – Результат программы

Задание повышенной сложности

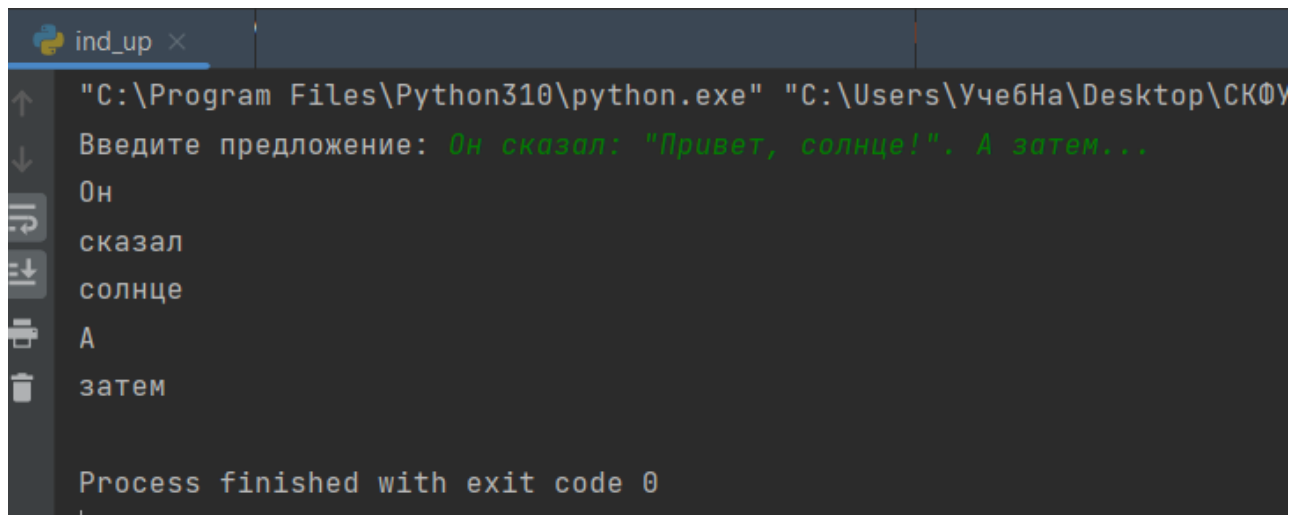
15. Дано предложение. Напечатать все его слова, отличные от слова привет

```
ex_1.py × ex_3.py × ind_1.py × ind_2.py × ind_3.py × ind_up.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     signs = ['.', ',', '!', '?', '/']
7
8     # Удаление пунктуационных знаков в предложении
9     for i in s:
10         if i in signs:
11             s = s.replace(i, "")
12
13     # Разделить предложение на слова.
14     words = s.split(' ')
15
16     # Вывод слов, отличных от слова 'Привет'.
17     for i in words:
18         if i != "Привет" and i != "привет":
19             print(i)
20
```

Рисунок 6.20 – Код программы

```
ind_up ×
↑ "C:\Program Files\Python310\python.exe"
↓ Введите предложение: Привет, мир!
⇅ мир
>> Process finished with exit code 0
```

Рисунок 6.21 – Результат программы



```
ind_up x
"C:\Program Files\Python310\python.exe" "C:\Users\УчебНа\Desktop\СКФУ
Введите предложение: Он сказал: "Привет, солнце!". А затем...
Он
сказал
солнце
А
затем

Process finished with exit code 0
```

Рисунок 6.22 – Результат программы

Вопросы для защиты работы

1. Что такое строки в языке Python?

Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации.

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

- Строки в апострофах и в кавычках
- Экранированные последовательности (- это служебные наборы символов, которые позволяют вставить нестандартные символы, которые сложно ввести с клавиатуры)
- "Сырые" строки - подавляют экранирование (Если перед открывающей кавычкой стоит символ 'r' (в любом регистре), то механизм экранирования отключается)
- Строки в тройных апострофах или кавычках (Главное достоинство строк в тройных кавычках в том, что их можно использовать для записи многострочных блоков текста)

3. Какие операции и функции существуют для строк?

- + — оператор конкатенации строк.

Он возвращает строку, состоящую из других строк

- Оператор умножения строк *

оператор создает несколько копий строки.

- Оператор принадлежности подстроки in.

Оператор in возвращает True , если подстрока входит в строку, и False , если нет

- Встроенные функции строк в python

Python предоставляет множество функций, которые встроены в интерпретатор.

Вот несколько, которые работают со строками:

Функция	Описание
chr()	Преобразует целое число в символ
ord()	Преобразует символ в целое число
len()	Возвращает длину строки
str()	Изменяет тип объекта на <code>string</code>

4. Как осуществляется индексирование строк?

Часто в языках программирования, отдельные элементы в упорядоченном наборе данных могут быть доступны с помощью числового индекса или ключа. Этот процесс называется индексация.

В Python строки являются упорядоченными последовательностями символьных данных и могут быть проиндексированы. Доступ к отдельным символам в строке можно получить, указав имя строки, за которым следует число в квадратных скобках []. Индексация строк начинается с нуля: у первого символа индекс 0, следующего 1 и так далее. Индекс последнего символа в python — “длина строки минус один”.

Отдельные символы доступны по индексу следующим образом:

```
>>> s = 'foobar'

>>> s[0]
'f'
>>> s[1]
'o'
>>> s[3]
'b'
>>> s[5]
'r'
```

5. Как осуществляется работа со срезами для строк?

Python также допускает возможность извлечения подстроки из строки, известную как “string slice”. Если s это строка, выражение формы

`s[m:n]` возвращает часть `s` , начинающуюся с позиции `m` , и до позиции `n` , но не включая позицию:

```
>>> s = 'python'
>>> s[2:5]
'tho'
```

Существует еще один вариант синтаксиса среза, о котором стоит упомянуть. Добавление дополнительного `:` и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе. Например , для строки `'python'` срез `0:6:2` начинается с первого символа и заканчивается последним символом (всей строкой), каждый второй символ пропускается.

```
>>> s = 'foobar'
>>> s[0:6:2]
'foa'
>>> s[1:6:2]
'obr'
```

6. Почему строки Python относятся к неизменяемому типу данных?

Мы не можем изменить существующую переменную, вместо этого мы должны создать новую с тем же именем.

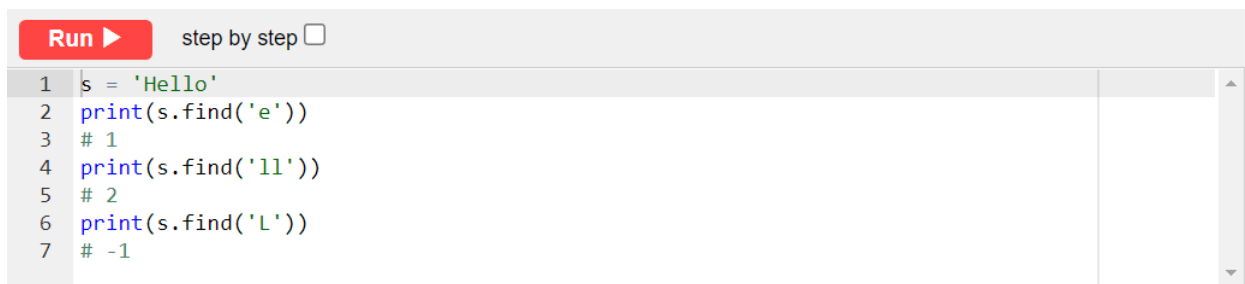
7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

`string.istitle()` определяет, начинаются ли слова строки с заглавной буквы.

```
>>> 'This Is A Title'.istitle()
True
>>> 'This is a title'.istitle()
False
>>> 'Give Me The $$$@ Ball!'.istitle()
True
```

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

Метод `find()` выполняет поиск подстроки, переданной как аргумент, внутри строки, на которую он вызывается. Функция возвращает индекс первого вхождения подстроки. Если подстрока не найдена, метод возвращает `-1`.



```
Run ▶ step by step □
1 s = 'Hello'
2 print(s.find('e'))
3 # 1
4 print(s.find('ll'))
5 # 2
6 print(s.find('L'))
7 # -1
```

Оператор `in` возвращает `True`, если подстрока входит в строку, и `False`, если нет:

```
>>> s = 'Python'
>>> s in 'I love Python.'
True
>>> s in 'I love Java.'
False
```

9. Как найти индекс первого вхождения подстроки в строку?

Метод `find()` выполняет поиск подстроки, переданной как аргумент, внутри строки, на которую он вызывается. Функция возвращает индекс первого вхождения подстроки. Если подстрока не найдена, метод возвращает `-1`.

10. Как подсчитать количество символов в строке?

Функция `len(s)` возвращает длину строки. `len(s)` возвращает количество символов в строке `s`

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

Метод `count()` строки возвращает количество вхождений подстроки в заданной строке.

```
>>> 'foo goo moo'.count('oo')
3
```

Количество вхождений изменится, если указать `<start>` и `<end>`:

```
>>> 'foo goo moo'.count('oo', 0, 8)
2
```

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

f-строки – это способ форматирования строк. Эта функция официально названа литералом отформатированной строки, но обычно упоминается как f-строки (f-string).

F-строки задаются с помощью литерала «f» перед кавычками.

```
>>> n = 20
>>> m = 25
>>> prod = n * m
>>> print(f'Произведение {n} на {m} равно {prod}')
Произведение 20 на 25 равно 500
```

13. Как найти подстроку в заданной части строки?

`string.count(<sub>[, <start> [,<end>]])` подсчитывает количество вхождений подстроки в строку.

Параметры:

sub - [str](#), строка или символ;

start - [int](#), индекс начала поиска, по умолчанию 0, необязательно;

end - int, конец, индекс конца поиска, по умолчанию [len\(str\)](#), необязательно.

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

Если для подстановки требуется только один аргумент, то значение - сам аргумент:

```
>>> 'hello, {}!'.format('Vasya')
'hello, vasya!'
```

А если несколько, то значениями будут являться все аргументы со строками подстановки (обычных или именованных):

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{} , {} , {}'.format('a', 'b', 'c')
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
```

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

string.isdigit() определяет, состоит ли строка из цифр (проверка на число).

s.isdigit() возвращает True когда строка s не пустая и все ее символы являются цифрами, а в False если нет

```
>>> '123'.isdigit()
True
>>> '123abc'.isdigit()
False
```

16. Как разделить строку по заданному символу?

Функция split() в Python используется для разделения строки на список строк на основе разделителя.


```
1. s = 'Python is Nice'
2.
3. # simple string split example
4. str_list = s.split(sep=' ')
5. print(str_list)
```

Вывод:

```
1. ['Python', 'is', 'Nice']
```

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

`string.islower()` определяет, являются ли буквенные символы строки строчными. `s.islower()` возвращает `True`, если строка `s` не пустая, и все содержащиеся в нем буквенные символы строчные, а `False` если нет. Не алфавитные символы игнорируются:

```
>>> 'abc'.islower()
True
>>> 'abc1$d'.islower()
True
>>> 'Abc1$D'.islower()
False
```

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

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

```
'aPPLE'[0].islower() #=> True
```

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

В Python при попытке выполнения подобной операции будет выдана ошибка `TypeError`.

```
'Ten' + 10 #=> 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.

```
''.join(reversed("hello world"))  
#=> 'dlrow olleh'
```

Или

Это общая парадигма для разворота (reverse) строки:

```
>>> s = 'Если так говорит товарищ Наполеон, значит, так оно и есть.'  
>>> s[::-1]  
'.'ытсе и оно кат ,тичанз ,ноелопан щиравог тировог кат илсе'
```

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

string.join() объединяет список в строку.

s.join() возвращает строку, которая является результатом конкатенации объекта с разделителем s .

```
>>> ', '.join(['foo', 'bar', 'baz', 'qux'])  
'foo, bar, baz, qux'
```

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.

```
sentence = 'The Cat in the Hat'
sentence.upper() #=> 'THE CAT IN THE HAT'
sentence.lower() #=> 'the cat in the hat'
```

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

Мы будем обращаться к символам строки по индексам. Строки в Python иммутабельны, поэтому мы будем заниматься сборкой новой строки на основе существующей.

```
animal = 'fish'
animal[0].upper() + animal[1:-1] + animal[-1].upper()
#=> 'FisH'
```

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

`string.isupper()` определяет, являются ли буквенные символы строки заглавными. `s.isupper()` возвращает `True`, если строка `s` не пустая, и все содержащиеся в ней буквенные символы являются заглавными, и `False`, если нет. Не алфавитные символы игнорируются:

```
>>> 'ABC'.isupper()
True
>>> 'ABC1$D'.isupper()
True
>>> 'Abc1$D'.isupper()
False
```

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

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

```
sentence = "It was a stormy night\nThe house creaked\nThe wind blew."  
sentence.splitlines()  
#=> ['It was a stormy night', 'The house creaked', 'The wind blew.']
```

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

Есть встроенный метод `string.replace(x, y)`:

```
>>> s = 'python'  
>>> s = s.replace('h', 't')  
>>> s  
'pytton'
```

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

`string.startswith([, [,]])` определяет, начинается ли строка с заданной подстроки.

`string.endswith([, [,]])` определяет, заканчивается ли строка заданной подстрокой.

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

`string.isspace()` определяет, состоит ли строка только из пробельных символов.

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

Строка повторится трижды.

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

`s.title()` возвращает копию, `s` в которой первая буква каждого слова преобразуется в верхний регистр, а остальные буквы — в нижний регистр:

```
>>> 'the sun also rises'.title()  
'The Sun Also Rises'
```

31. Как пользоваться методом partition()?

string.partition() делит строку на основе разделителя. s.partition() отделяет от s подстроку длиной от начала до первого вхождения . Возвращаемое значение представляет собой кортеж из трех частей:

- Часть s до <sep>
- Разделитель <sep>
- Часть s после <sep>

```
>>> 'foo.bar'.partition('.')  
('foo', '.', 'bar')  
>>> 'foo@@bar@@baz'.partition('@@')  
('foo', '@@', 'bar@@baz')
```

32. В каких ситуациях пользуются методом rfind()?

string.rfind([, [,]]) ищет в строке заданную подстроку, начиная с конца. s.rfind() возвращает индекс последнего вхождения подстроки в s , который соответствует началу <sep>:

```
>>> 'Follow Us @Python'.rfind('o')  
15
```

Как и в .find() , если подстрока не найдена, возвращается -1 :

```
>>> 'Follow Us @Python'.rfind('a')  
-1
```