

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 №6**  
**дисциплины «Программирование на Python»**  
**Вариант 29**

Выполнил:  
Саенко Андрей Максимович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», заочная  
форма обучения

---

(подпись)

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

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

Тема: Работа со строками в языке Python

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

Порядок выполнения работы:

### 1. Создан общедоступный репозиторий на GitHub

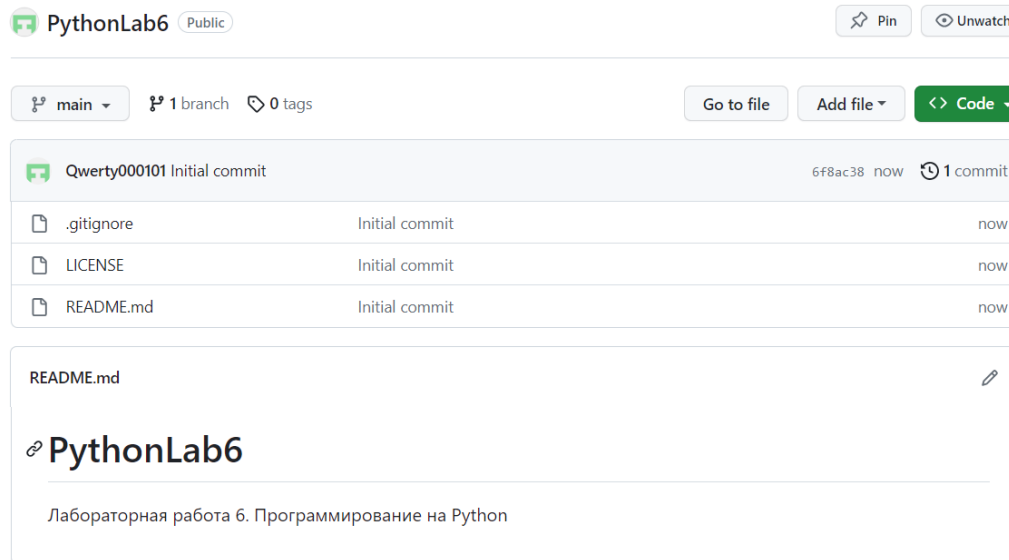


Рисунок 1 – Созданный репозиторий на GitHub

### 2. Выполнено клонирование созданного репозитория.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python>git clone https://github.com/Qwerty000101/PythonLab6.git
Cloning into 'PythonLab6'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\HAIER\Desktop\Задания\Программирование на Python>
```

Рисунок 2 – Клонирование репозитория

### 3. Организация репозитория для работы по git-flow

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab6>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]

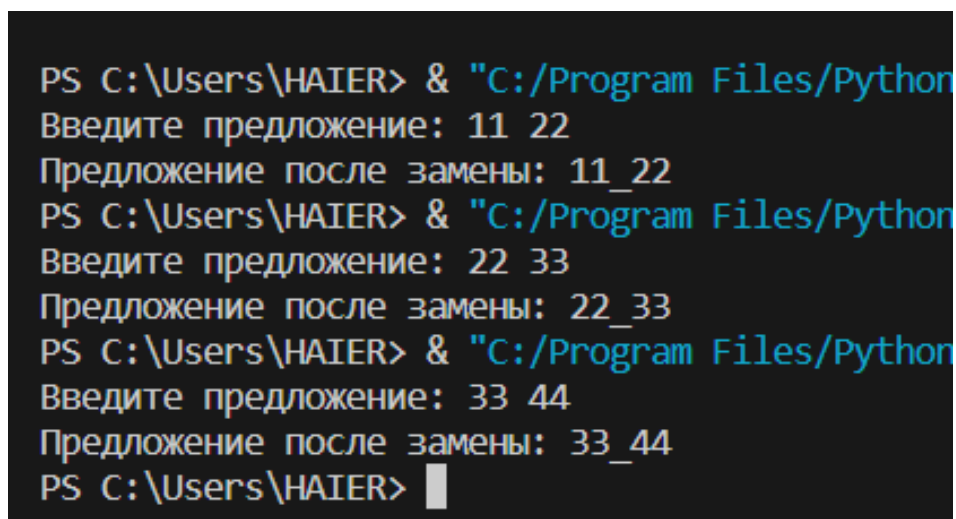
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/HAIER/Desktop/Задания/Программирование на Python/PythonLab6/.git/hooks]
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab6>
```

Рисунок 3 – Запуск git-flow

4. Проработаны примеры лабораторной работы. Для каждого примера создан отдельный модуль языка Python.

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

Результат работы программы:



```
PS C:\Users\HAIER> & "C:/Program Files/Python
Введите предложение: 11 22
Предложение после замены: 11_22
PS C:\Users\HAIER> & "C:/Program Files/Python
Введите предложение: 22 33
Предложение после замены: 22_33
PS C:\Users\HAIER> & "C:/Program Files/Python
Введите предложение: 33 44
Предложение после замены: 33_44
PS C:\Users\HAIER> █
```

Рисунок 4 – Результат работы программы

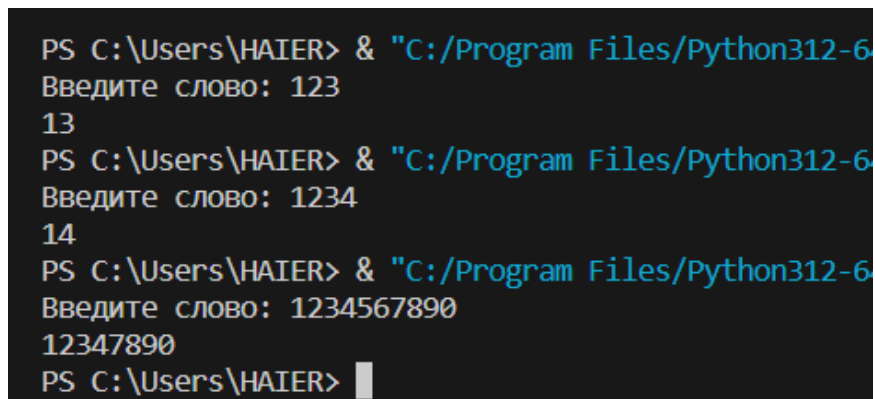
Код программы:

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

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

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

Результат работы программы:



```
PS C:\Users\HAIER> & "C:/Program Files/Python312-6
Введите слово: 123
13
PS C:\Users\HAIER> & "C:/Program Files/Python312-6
Введите слово: 1234
14
PS C:\Users\HAIER> & "C:/Program Files/Python312-6
Введите слово: 1234567890
12347890
PS C:\Users\HAIER> █
```

Рисунок 5 – Результат работы программы

### Код программы:

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

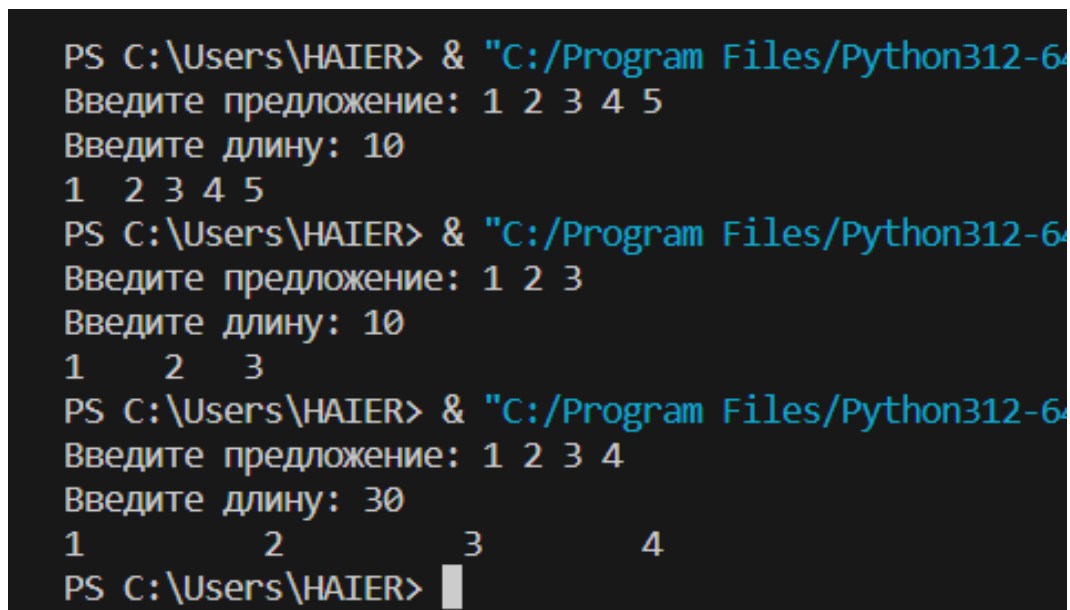
if __name__ == '__main__':
    word = input("Введите слово: ")

    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)
```

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.

### Результат работы программы:



```
PS C:\Users\HAIER> & "C:/Program Files/Python312-64
Введите предложение: 1 2 3 4 5
Введите длину: 10
1 2 3 4 5
PS C:\Users\HAIER> & "C:/Program Files/Python312-64
Введите предложение: 1 2 3
Введите длину: 10
1 2 3
PS C:\Users\HAIER> & "C:/Program Files/Python312-64
Введите предложение: 1 2 3 4
Введите длину: 30
1 2 3 4
PS C:\Users\HAIER>
```

Рисунок 6 – Результат работы программы

## Код программы:

```
#!/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))
```

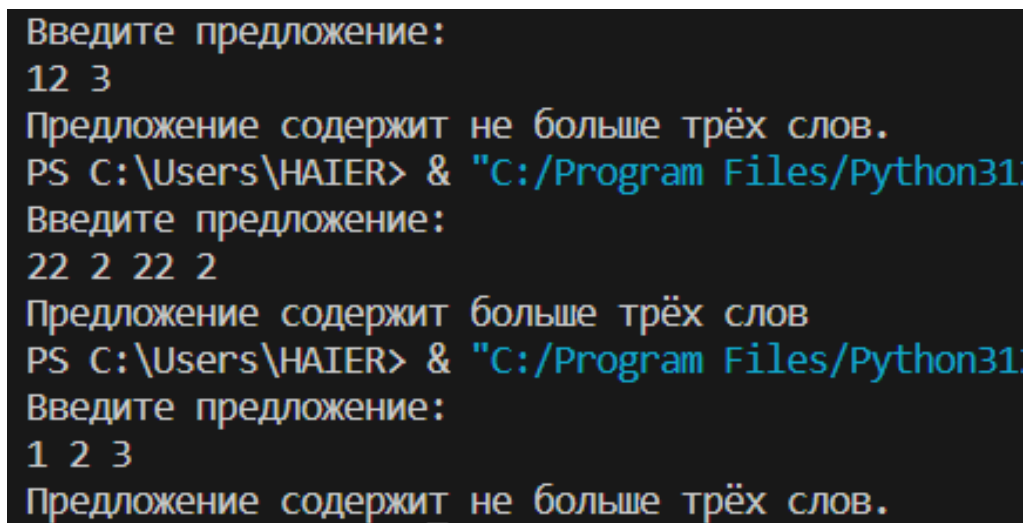
5. Выполнены индивидуальные задания и задание повышенной сложности

Вариант 29

Индивидуальное задание 1.

Дано предложение. В нем слова разделены одним пробелом (символ «-» в предложении отсутствует). Верно ли, что число слов в предложении больше трех?

Результат работы программы:



```
Введите предложение:
12 3
Предложение содержит не больше трёх слов.
PS C:\Users\HAIER> & "C:/Program Files/Python31
Введите предложение:
22 2 22 2
Предложение содержит больше трёх слов
PS C:\Users\HAIER> & "C:/Program Files/Python31
Введите предложение:
1 2 3
Предложение содержит не больше трёх слов.
```

Рисунок 7 – Результат работы программы

Код программы:

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

import sys

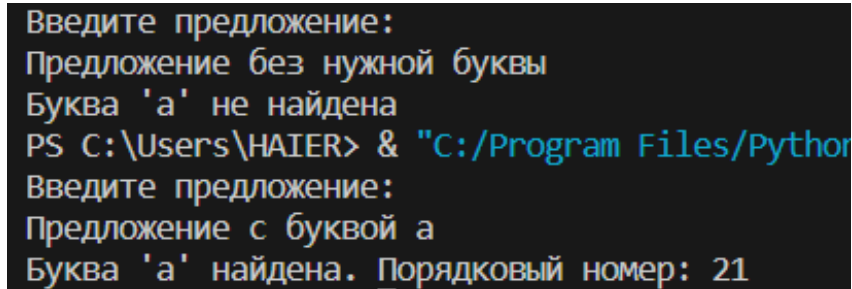
if __name__ == '__main__':
    n = input("Введите предложение:\n").split()

    if len(n) > 3:
        print("Предложение содержит больше трёх слов")
    else:
        print("Предложение содержит не больше трёх слов.")
```

## Индивидуальное задание 2.

Дано предложение. Определить, есть ли буква «а» в нем. В случае положительного ответа найти также порядковый номер первой из них.

Результат работы программы:



```
Введите предложение:
Предложение без нужной буквы
Буква 'а' не найдена
PS C:\Users\NAIER> & "C:/Program Files/Python
Введите предложение:
Предложение с буквой а
Буква 'а' найдена. Порядковый номер: 21
```

Рисунок 8 – Результат работы программы

Код программы:

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

import sys

if __name__ == '__main__':
    n = input("Введите предложение:\n")

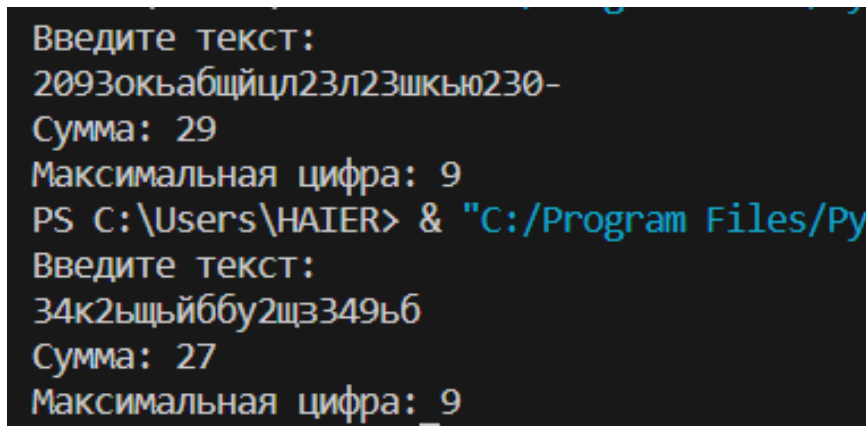
    res = n.find("a")
    if res == -1:
        print("Буква 'а' не найдена")
    else:
        print(f'Буква 'а' найдена. Порядковый номер: {res}')
```

### Индивидуальное задание 3.

Дан текст, в котором имеются цифры.

- 1) Найти их сумму.
- 2) Найти максимальную цифру.

Результат работы программы:



```
Введите текст:
2093окьябщйцл23л23шкью230-
Сумма: 29
Максимальная цифра: 9
PS C:\Users\NAIER> & "C:/Program Files/Py
Введите текст:
34к2ьщйббу2щз349ьб
Сумма: 27
Максимальная цифра: 9
```

Рисунок 8 – Результат работы программы

Код программы:

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

if __name__ == '__main__':
    n = input("Введите текст:\n")

    numbers_str = " ".join(c for c in n if c.isdigit()).split()
    numbers = [int(item) for item in numbers_str]

    print(f"Сумма: {sum(numbers)}\nМаксимальная цифра: {max(numbers)} ")
```

### Задание повышенной сложности.

Даны два предложения. Для каждого слова первого предложения определить, входит ли оно во второе предложение. Повторяющиеся слова первого предложения не рассматривать.

Код программы:

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

if __name__ == '__main__':
    s_first = input("Введите первое предложение:\n").split()
    s_second = input("Введите второе предложение:\n").split()
```



```

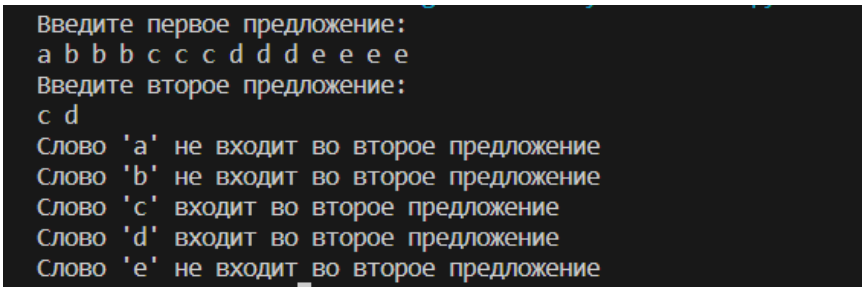
st = []

for word in s_first:
    if word in st:
        continue
    else:
        st.append(word)

for word in st:
    if s_second.count(word) != 0:
        print(f'Слово '{word}' входит во второе предложение")
    else:
        print(f'Слово '{word}' не входит во второе предложение")

```

Результат работы программы:



```

Введите первое предложение:
a b b b c c c d d d e e e
Введите второе предложение:
c d
Слово 'a' не входит во второе предложение
Слово 'b' не входит во второе предложение
Слово 'c' входит во второе предложение
Слово 'd' входит во второе предложение
Слово 'e' не входит во второе предложение

```

Рисунок 9 – Результат работы программы

Ответы на контрольные вопросы:

1. Что такое строки в языке Python?

Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

2. Какие существуют способы задания строковых литералов в языке Python

Существуют следующие способы задания строковых литералов в языке Python:

- 1) Строки в апострофах и в кавычках;
- 2) Экранированные последовательности - служебные символы;
- 3) Строки в тройных апострофах или кавычках;
- 4) «Сырые» строки.

3. Какие операции и функции существуют для строк?

Существуют следующие операции и функции для строк:

- 1) Сложение с помощью оператора «+»;
- 2) Умножение строк с помощью оператора «\*»;
- 3) Оператор принадлежности подстроки in;
- 4) chr() (преобразует целое число в символ);
- 5) ord() (преобразует символ в целое число);
- 6) len() (возвращает длину строки);
- 7) str() (изменяет тип объекта на string);
- 8) Индексация строк;
- 9) Срезы строк;
- 10) Форматирование строки (метод «format», оператор «%», f-строки);
- 11) string.replace(x, y) (изменение в строке символа x на символ y);
- 12) string.capitalize() (приводит первую букву в верхний регистр, остальные в нижний);
- 13) string.lower() (преобразует все буквенные символы в строчные);
- 14) string.swapcase() (меняет регистр буквенных символов на противоположный);
- 15) string.title() (преобразует первые буквы всех слов в заглавные);
- 16) string.upper() (преобразует все буквенные символы в заглавные);
- 17) string.count(<sub>[, <start>[, <end>]]) (подсчитывает количество вхождений подстроки в строку);
- 18) string.endswith(<sub>[,<start>[,<end>]]) (определяет, заканчивается ли строка заданной подстрокой);
- 19) string.find(<sub>[,<start>[,<end>]]) (ищет в строке заданную подстроку);
- 20) string.index(<sub>[,<start>[,<end>]]) (ищет в строке заданную подстроку)
- 21) string.rfind(<sub>[,<start>[,<end>]]) (ищет в строке заданную подстроку, начиная с конца);

22) `string.rindex(<sub>[,<start>[,<end>]])` (ищет в строке заданную подстроку, начиная с конца);

23) `string.startswith(<sub>[,<start>[,<end>]])` (определяет, начинается ли строка с заданной подстроки);

24) `string.isalnum()` (определяет, состоит ли строка из букв и цифр);

25) `string.isalpha()` (определяет, состоит ли строка только из букв);

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

27) `string.isidentifier()` (определяет, является ли строка допустимым идентификатором);

28) `iskeyword()` (проверяет, является ли строка ключевым словом Python);

29) `string.islower()` (определяет, являются ли буквенные символы строки строчным);

30) `string.isprintable()` (определяет, состоит ли строка только из печатаемых символов);

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

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

33) `string.isupper()` (определяет, являются ли буквенные символы строки заглавными);

34) `string.center(<end> [, <fill> ])` (выравнивает строку по центру);

35) `string.expandtabs(tabsize=8)` (заменяет табуляции на пробелы);

36) `string.ljust(<width>[,<fill>])` выравнивание по левому краю строки в поле.

37) `string.lstrip([chars])` (обрезает пробельные символы слева);

38) `string.replace(<old>, <new>[, <count>])` заменяет вхождения подстроки в строке.

39) `string.rjust(<width>[, <fill>])` выравнивание по правому краю строки в поле.

40) `string.rstrip([<chars>])` (обрезает пробельные символы справа);

41) `string.strip([chars])` (удаляет символы с левого и правого края строки);

42) `string.zfill(<width>)` (дополняет строку нулями слева);

43) `string.join(<iterable>)` (объединяет список в строку);

44) `string.partition(<sep>)` (делит строку на основе разделителя);

45) `s.rpartition(<sep>)` (делит строку на основе разделителя, начиная с конца);

46) `string.rsplit(sep=None, maxsplit=-1)` (делит строку на список из подстрок);

47) `string.split(sep=None, maxsplit=-1)` (делит строку на список из подстрок);

48) `string.splitlines([<keepends>])` (делит текст на список строк);

4. Как осуществляется индексирование строк?

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> - 1».

5. Как осуществляется работа со срезами для строк?

Если `s` это строка, выражение формы `s[m:n]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая эту позицию.

Если пропустить первый индекс, срез начинается с начала строки. Таким образом, `s[:m] = s[0:m]`.

Аналогично, если опустить второй индекс `s[n:]`, срез длится от первого индекса до конца строки.

Добавление дополнительного «:» и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки в Python относятся к неизменяемому типу данных, потому что нет особой необходимости изменять строки. Есть возможность создать копию исходной строки с внесёнными изменениями.

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

Для того, чтобы проверить, начинается ли каждое слово в строке с заглавной буквы, необходимо использовать `string.istitle()`.

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

Для того, чтобы проверить строку на вхождение в неё другой строки, можно воспользоваться методом `string.count()`. Если искомой подстроки нет, то результатом работы данного метода будет 0, а если есть, то любое число, отличное от нуля.

9. Как найти индекс первого вхождения подстроки в строку?

Для того, чтобы найти индекс первого вхождения подстроки в строку, можно воспользоваться методом `string.find()`.

10. Как подсчитать количество символов в строке?

Чтобы посчитать количество символов в строке, можно воспользоваться функцией `len()`.

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

Чтобы посчитать сколько раз определённый символ встречается в строке, можно воспользоваться методом `string.count()`.

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

f-строки – метод форматирования строк. Перед кавычками строки ставится буква `f` или `F`, после чего внутри кавычек можно указывать

переменные, пользуясь фигурными скобками, в этих местах будет выведено значение этих переменных.

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

Чтобы найти подстроку в заданной части строки, нужно в методе `string.find()` указать границы, в пределах которых будет осуществляться поиск.

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

Чтобы вставить содержимое переменной в строку с помощью метода `format()`, необходимо после закрывающей кавычки строки написать «`.format()`», указать в методе переменную и добавить фигурные скобки в то место строки, где требуется вывести значение этой переменной.

Для вывода нескольких переменных необходимо указать их в методе через запятую, а в каждой фигурных скобках записать порядковый номер выводимой переменной.

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

Чтобы определить, состоит ли строка только из цифр, необходимо воспользоваться методом `string.isdigit()`.

### 16. Как разделить строку по заданному символу?

Чтобы разделить строку по заданному символу, необходимо воспользоваться методом `string.split()`, где в качестве параметра должен быть указан данный символ.

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

Чтобы проверить, состоит ли строка только из строчных букв, необходимо воспользоваться методом `string.islower()`.

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

Чтобы проверить, начинается ли строка со строчной буквы, необходимо взять срез строки, оставив только первый символ и проверить его с помощью метода `islower()`.

Код программы может выглядеть вот так:

```
s = "abc"  
print(s[:1].islower())
```

Будет выведено True, так как первый символ в строке является строчной буквой.

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

Попытка прибавить целое число к строке приведёт к ошибкам, если не преобразовать его в строку.

20. Как «перевернуть» строку?

Чтобы «перевернуть» строку, можно воспользоваться срезом, написав <имя переменной, содержащей строку>[::-1].

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

Чтобы объединить список lst в одну строку, элементы которой разделены дефисом, следует написать “-“.join(lst).

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

Чтобы привести всю строку к верхнему регистру, следует воспользоваться методом string.upper().

Для того, чтобы привести всю строку к нижнему регистру, следует использовать метод string.lower().

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

Для того, чтобы преобразовать первый и последний символы строки st к верхнему регистру, нужно написать следующий код:

```
st = st[0].upper() + st[1:-1] + st[-1].upper()
```

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

Чтобы выяснить, составлена ли строка st только из прописных букв, нужно использовать методы st.isalpha() и st.islower(). Если st.isalpha() == True и st.islower() == True, то строка состоит только из прописных букв.

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

Этот метод может быть полезен, когда нужно разделить многострочный текст на отдельные строки для дальнейшей обработки. Например, при чтении текстового файла с помощью функции `readlines()`, каждая строка будет содержать символ новой строки в конце, и метод `splitlines()` может быть использован для удаления этих символов и получения списка строк без них.

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

Для этого необходимо использовать метод `string.replace()`, указав в качестве параметров заменяемую подстроку и подстроку, которая будет на месте старой.

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

Для этого нужно использовать методы `string.startswith()` и `string.endswith()`, которые возвращают `True` или `False` в зависимости от того, начинается (заканчивается) ли строка с заданной последовательности символов (заданной последовательностью символов).

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

Для этого необходимо использовать метод `string.isspace()`, который возвращает `True`, если строка состоит из символов пробела.

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

Если умножить строку на число, то будет создано и объединено в одну строку три копии этой строки.

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

Для этого необходимо использовать метод `string.title()`.



### 31. Как пользоваться методом partition() ?

Метод partition() в Python используется для разделения строки на три части: подстроку до первого вхождения заданного разделителя, сам разделитель и подстроку после него.

Синтаксис метода partition() выглядит следующим образом:

```
string.partition(<sep>)
```

Если <sep> не найден, то возвращаемый кортеж содержит s и две пустые строки

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

Метод rfind() в Python используется, когда необходимо найти последнее вхождение подстроки в строку.

#### Вывод

Приобретены навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x.