

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 ПРАКТИЧЕСКОЙ РАБОТЕ №2.3**  
**дисциплины «Программирование на Python»**

Выполнил:  
Степанов Леонид Викторович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение  
средств вычислительной техники  
и автоматизирование систем»,  
очная форма обучения

---

(подпись)

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

---

(подпись)

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

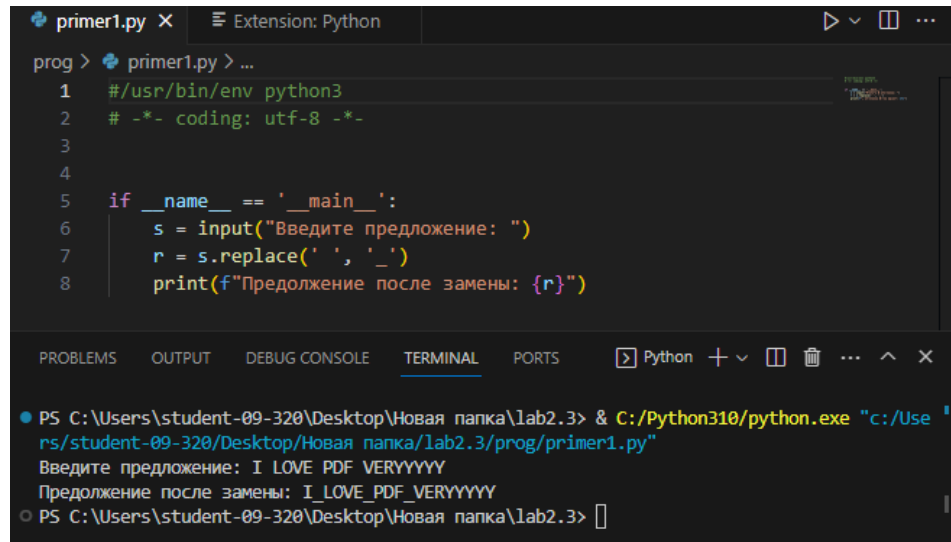
Ставрополь, 2023 г.

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. Написал программу (primer1.py), которая пробелы заменяет на СИМВОЛ «\_»

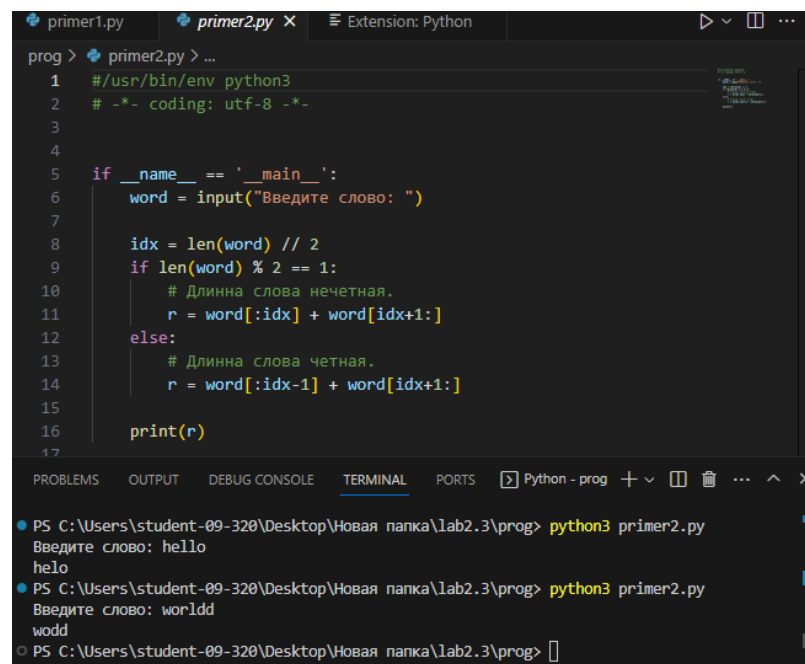


```
primer1.py x Extension: Python
prog > primer1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      s = input("Введите предложение: ")
7      r = s.replace(' ', '_')
8      print(f"Предложение после замены: {r}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - - - - -
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3> & C:/Python310/python.exe "c:/Use
rs/student-09-320/Desktop/Новая папка/lab2.3/prog/primer1.py"
Введите предложение: I LOVE PDF VERYYYYY
Предложение после замены: I_LOVE_PDF_VERYYYYY
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3> 
```

Рисунок 1 – Результат выполнения primer1.py

2. Написал программу (primer2.py), которая изменяет слово: если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.



```
primer1.py primer2.py x Extension: Python
prog > primer2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      word = input("Введите слово: ")
7
8      idx = len(word) // 2
9      if len(word) % 2 == 1:
10         # Длина слова нечетная.
11         r = word[:idx] + word[idx+1:]
12     else:
13         # Длина слова четная.
14         r = word[:idx-1] + word[idx+1:]
15
16     print(r)
17

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python - prog + - - - - -
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3\prog> python3 primer2.py
Введите слово: hello
helo
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3\prog> python3 primer2.py
Введите слово: worldl
wodd
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3\prog> 
```

Рисунок 2 – Результат выполнения primer2.py

3. Написал программу (primer3.py), в которой входные данные: строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки стала равна заданной длине. Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными словами должно отличаться не более чем на 1

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'primer1.py' selected. The main editor area displays the code for 'primer1.py', which is a Python script that takes a sentence as input, splits it into words, and prints the words. The script is as follows:

```

1 #usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     s = input("Введите предложение: ")
8     n = int(input("Введите длину: "))
9
10    # Проверить требование длины.
11    if len(s) >= n:
12        print(
13            "Заданная длина должна быть больше длины предложения",
14            file=sys.stderr
15        )
16        exit(1)
17
18    # Разделить предложение на слова
19    words = s.split(' ')
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
30    for word in words:
31        delta -= len(word)
32
33    w, r = delta // len(words) - 1, delta % (len(words) - 1)
34
35    # Сформировать список для хранения слов и пробелов.

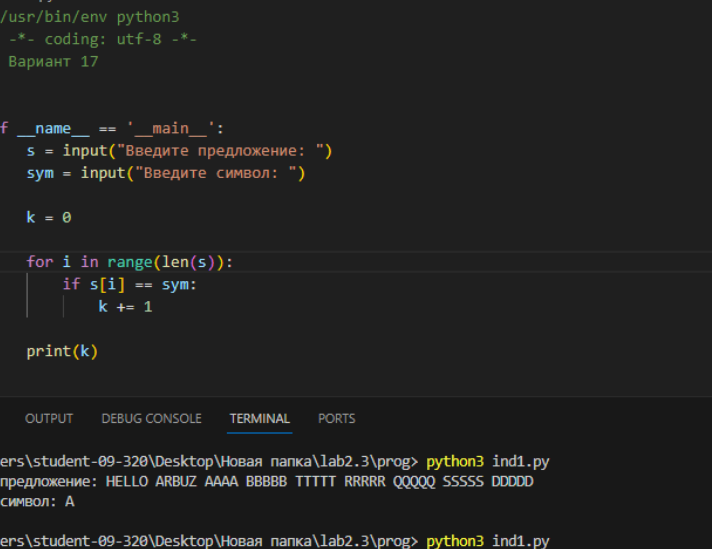
```

The 'OUTPUT' panel at the bottom shows the execution of the script. It displays the input 'hello world M hi hi alldasdasda adasdasda' and the output 'hello world M hi hi alldasdasda adasdasda'.

Рисунок 3 – Результат выполнения primer3.py

### Индивидуальные задания (Вариант 17):

Задание 1. Вывести число вхождений символа в строку:



The screenshot shows the PyCharm IDE interface. At the top, there are tabs for 'primer1.py', 'ind1.py', and 'Extension: Python'. The main editor window displays a Python script named 'ind1.py' with the following code:

```

1  #usr/bin/env python3
2  #- coding: utf-8 -*-
3  # Вариант 17
4
5
6  if __name__ == '__main__':
7      s = input("Введите предложение: ")
8      sym = input("Введите символ: ")
9
10     k = 0
11
12     for i in range(len(s)):
13         if s[i] == sym:
14             k += 1
15
16     print(k)

```

Below the editor, the 'TERMINAL' tab is active, showing the execution of the script. The terminal output is as follows:

```

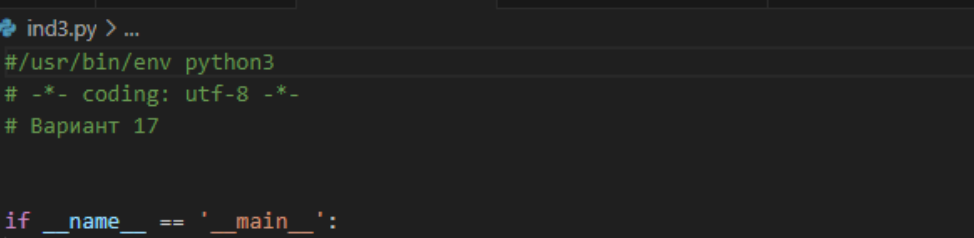
PS C:\Users\student-09-320\Desktop\Новая nanka\lab2.3\prog> python3 ind1.py
Введите предложение: HELLO ARBUZ AAAA BBBB TTTT RRRR QQQQ SSSS DDDD
Введите символ: A
5
PS C:\Users\student-09-320\Desktop\Новая nanka\lab2.3\prog> python3 ind1.py
Введите предложение: DFGJDFGKLJGDFGJGJDFJGDFJ SDIOFKFKFSFKSDO DOKSFKSFJISFJKSL;Fkaaa vjfkfjsfnd
Введите символ: G
6
PS C:\Users\student-09-320\Desktop\Новая nanka\lab2.3\prog>

```

Рисунок 4 – Результат выполнения ind1.py

[illegible]

Задание 3. Дано ошибочно написанное слово алигортм. Путём перемещения его букв получить слово алгоритм.



The screenshot shows the VS Code interface with the file explorer on the left displaying a project structure with folders 'lab2.3' and 'lab2.4'. The main editor area shows the file 'ind3.py' with the following Python code:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Вариант 17
4
5
6  if __name__ == '__main__':
7      s = "алигортм"
8
9      r = s[:2]+s[3:6]+s[2]+s[6:]
10
11     print(r)
```

Below the editor, the 'TERMINAL' tab is active, showing the command prompt output:

```
PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3\prog> python3 ind3.py
алигортм
```

The terminal output shows the string 'алигортм' printed, which is the result of the string slicing operation in the code.

Задание повышенной сложности: создаю файл (high.py) в котором в соответствии с заданием: дано предложение, вывести на экран его слова:

начинающиеся и оканчивающиеся на одну и ту же букву, которые содержат ровно три буквы е; которые содержат хотя бы одну букву о, пишу код.



```
prog > high.py > ...
1  #/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Вариант 17
4
5
6  if __name__ == '__main__':
7      s = input()
8
9      words = s.split(' ')
10
11     rez1 = []
12     rez2 = []
13     rez3 = []
14     for i, word in enumerate(words):
15         # начинающиеся и оканчивающиеся на одну и ту же букву
16         if word[0] == word[len(word)-1]:
17             rez1.append(word)
18
19         countE = 0
20         countO = 0
21         for j in range(len(word)):
22             if word[j] == "e":
23                 countE += 1
24             if word[j] == "o":
25                 countO += 1
26
27         if countE == 3:
28             rez2.append(word)
29
30         if countO >= 1:
31             rez3.append(word)
32     print(rez1)
33     print(rez2)
34     print(rez3)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\student-09-320\Desktop\Новая папка\lab2.3\prog> python3 high.py
eee ooo eesdfsdfsddfsffeee sdfsjfsdifsjds edssadsadaasdaee ooasdaaso
['eee', 'ooo', 'eesdfsdfsddfsffeee', 'sdfsjfsdifsjds', 'edssadsadaasdaee', 'ooasdaaso']
['eee', 'edssadsadaasdaee']
['ooo', 'ooasdaaso']
```

Рисунок 6 – Результат выполнения high.py

Вывод: в результате проделанной работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

## Контрольные вопросы:

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

В языке Python, строки представляют собой последовательности символов, которые используются для хранения и обработки текстовой информации. Строки могут содержать буквы, цифры, символы пунктуации и другие специальные символы.

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

В языке Python существует несколько способов задания строковых литералов:

1) Одинарные кавычки: Строковый литерал может быть заключен в одинарные кавычки. Например: 'Привет, мир!'.

2) Двойные кавычки: Строковый литерал также может быть заключен в двойные кавычки. Например: "Привет, мир!".

3) Тройные кавычки: Тройные кавычки могут быть использованы для задания многострочных строковых литералов. Например:

4) Экранирование специальных символов: В строковых литералах можно использовать экранирование для вставки специальных символов, таких как символ новой строки `\n`, символ табуляции `\t`, символ возврата каретки `\r` и другие. Например: "Привет, мир!\n"

5) Raw-строки: Raw-строки позволяют игнорировать экранирование специальных символов. Они создаются путем добавления префикса `r` перед строковым литералом. Например: `r"Привет, мир!\n"`

### 3. Операции и функции для строк

В Python существует множество операций и функций для работы со строками. Некоторые из них включают:

1) Конкатенация строк: Для объединения двух строк используется оператор `+`. Например, `"Hello" + "World"` вернет строку `"HelloWorld"`.

2) Умножение строки: Строку можно умножить на целое число, чтобы повторить ее содержимое несколько раз. Например, "Hello" \* 3 вернет строку "HelloHelloHello".

3) Индексирование строк: Строки в Python можно индексировать, чтобы получить доступ к отдельным символам. Индексация начинается с 0. Например, "Hello"[0] вернет символ "H", а "Hello"[1] вернет символ "e".

4) Срезы строк: С помощью срезов можно получить подстроку из исходной строки. Синтаксис срезов выглядит следующим образом: строка[начало:конец:шаг]. Например, "Hello"[1:4] вернет подстроку "ell". Если не указывать начало или конец среза, то будут использованы значения по умолчанию. Например, "Hello"[:3] вернет подстроку "Hel", а "Hello"[2:] вернет подстроку "llo".

5) Методы строк: В Python есть множество методов для работы со строками. Некоторые из них включают upper() для преобразования строки к верхнему регистру, lower() для преобразования строки к нижнему регистру, find() для поиска подстроки в строке, replace() для замены подстроки на другую строку, split() для разделения строки на список подстрок и многие другие.

#### 4. Индексирование строк

Индексирование строк в Python позволяет получать доступ к отдельным символам в строке. Индексация начинается с 0, то есть первый символ имеет индекс 0, второй символ - индекс 1 и так далее. Для получения символа по индексу используется квадратные скобки и индекс внутри них. Например, Если индекс выходит за пределы длины строки, будет вызвано исключение IndexError.

#### 5. Работа со срезами для строк

Срезы строк в Python позволяют получать подстроку из исходной строки. Синтаксис срезов выглядит следующим образом: строка[начало:конец:шаг]. Начало и конец указывают индексы символов, которые включаются в срез. Шаг определяет, через сколько символов брать

следующий символ в срезе. Если шаг не указан, по умолчанию он равен 1. Если не указывать начало или конец среза, то будут использованы значения по умолчанию.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки в Python относятся к неизменяемому типу данных, что означает, что после создания строки ее нельзя изменить. Это связано с тем, что строки в Python представляют собой последовательность символов, и изменение одного символа в строке потребовало бы изменения всей последовательности символов. Вместо этого, при изменении строки создается новая строка с измененным содержимым. Это обеспечивает безопасность и предотвращает неожиданные изменения в строках, которые могут быть использованы в различных частях программы.

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

Для проверки того, что каждое слово в строке начинается с заглавной буквы, можно воспользоваться методом `istitle()`. Метод `istitle()` возвращает значение `True`, если каждое слово в строке начинается с заглавной буквы, и `False` в противном случае.

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

Для проверки того, что одна строка содержится в другой строке, можно воспользоваться оператором `in`. Оператор `in` возвращает значение `True`, если первая строка содержится во второй строке, и `False` в противном случае.

9. Как найти индекс первого вхождения подстроки в строку?

Для поиска индекса первого вхождения подстроки в строку можно воспользоваться методом `find()`. Метод `find()` возвращает индекс первого вхождения подстроки в строку или `-1`, если подстрока не найдена.

10. Как подсчитать количество символов в строке?

Для подсчета количества символов в строке можно воспользоваться функцией `len()`. Функция `len()` возвращает количество символов в строке.



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

Для подсчета количества вхождений определенного символа в строку можно воспользоваться методом `count()`. Метод `count()` возвращает количество вхождений символа в строку.

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

F-строки (или форматированные строки) являются специальным видом строк в Python, которые позволяют вставлять значения переменных непосредственно внутрь строки. Для использования f-строк необходимо добавить префикс `f` перед открывающей кавычкой строки и внутри строки использовать фигурные скобки `{ }` для обозначения места вставки переменных.

F-строки позволяют использовать выражения внутри фигурных скобок для выполнения операций или вызова функций.

F-строки также поддерживают форматирование значений с использованием спецификаторов формата, таких как `:2f` для округления чисел с плавающей запятой до двух десятичных знаков.

13. Как найти подстроку в заданной части строки?

Для поиска подстроки в заданной части строки в Python можно воспользоваться методом `find()`. Этот метод возвращает индекс первого вхождения подстроки в строку или `-1`, если подстрока не найдена.

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

Для вставки содержимого переменной в строку в Python можно воспользоваться методом `format()`. Этот метод позволяет вставлять значения переменных внутрь строки, используя фигурные скобки `{ }` для обозначения места вставки.

Метод `format()` также позволяет указывать порядковый номер аргумента или использовать именованные аргументы для более точного управления вставкой значений.

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

Для проверки того, что в строке содержатся только цифры, можно воспользоваться методом `isdigit()`. Этот метод возвращает `True`, если все символы в строке являются цифрами, и `False` в противном случае.

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

Для разделения строки по заданному символу в Python можно воспользоваться методом `split()`. Этот метод разделяет строку на подстроки по указанному символу и возвращает список подстрок.

Метод `split()` также позволяет указывать другие разделители, такие как пробел или перенос строки.

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

Для проверки того, что строка состоит только из строчных букв в Python можно воспользоваться методом `islower()`. Этот метод возвращает `True`, если все символы в строке являются строчными буквами, и `False` в противном случае.

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

Для проверки того, что строка начинается со строчной буквы в Python можно воспользоваться методом `islower()` в сочетании с методом `startswith()`. Метод `islower()` проверяет, что первый символ строки является строчной буквой, а метод `startswith()` проверяет, что первый символ строки соответствует заданной последовательности символов.

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

В Python нельзя прибавить целое число к строке напрямую. Однако, можно преобразовать целое число в строку с помощью функции `str()` и затем с конкатенировать строки с использованием оператора `+`

#### 20. Как «перевернуть» строку?

Для "переворачивания" строки в Python можно использовать срезы. Срезы позволяют выбирать части строки по определенным индексам. Чтобы перевернуть строку, можно использовать отрицательный шаг среза.

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

Для объединения списка строк в одну строку с разделителем можно использовать метод `join()` в Python.

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

Для приведения строки к верхнему или нижнему регистру в Python можно использовать методы `upper()` и `lower()` соответственно.

Метод `upper()` преобразует все символы строки в верхний регистр, а метод `lower()` преобразует все символы строки в нижний регистр.

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

Для преобразования первого и последнего символов строки к верхнему регистру в Python можно использовать методы `capitalize()` и `title()`.

Метод `capitalize()` преобразует первый символ строки в верхний регистр, а метод `title()` преобразует первый символ каждого слова в верхний регистр.

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

Для проверки того, что строка состоит только из прописных букв в Python, можно использовать метод `islower()`.

Метод `islower()` возвращает `True`, если все символы строки являются прописными буквами, и `False` в противном случае.

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

Метод `splitlines()` в Python используется для разделения строки на список строк по символам новой строки. Этот метод может быть полезен, когда у вас есть многострочная строка и вы хотите разделить ее на отдельные строки.

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

Для замены всех вхождений подстроки в заданной строке в Python можно использовать метод `replace()`. Метод `replace()` заменяет все вхождения подстроки в строке и возвращает новую строку.

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

Для проверки того, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов в Python, можно использовать методы `startswith()` и `endswith()` соответственно. Метод `startswith()` возвращает `True`, если строка начинается с указанной последовательности символов, и `False` в противном случае. Метод `endswith()` возвращает `True`, если строка заканчивается указанной последовательностью символов, и `False` в противном случае.

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

Для проверки того, что строка состоит только из пробелов в Python можно использовать метод `isspace()`. Метод `isspace()` возвращает `True`, если все символы строки являются пробелами, и `False` в противном случае.

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

Если умножить строку на целое число в Python, то строка будет повторена указанное количество раз.

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

Для приведения к верхнему регистру первого символа каждого слова в строке в Python можно использовать метод `title()`. Метод `title()` преобразует первый символ каждого слова в верхний регистр, а остальные символы в нижний регистр.

31. Как пользоваться методом `partition()`?

Метод `partition()` в Python используется для деления строки на три части: часть перед первым вхождением указанной подстроки, саму подстроку и часть после нее.