ООП НА PYTHON

Урок 7. Строки. Индексы и срезы

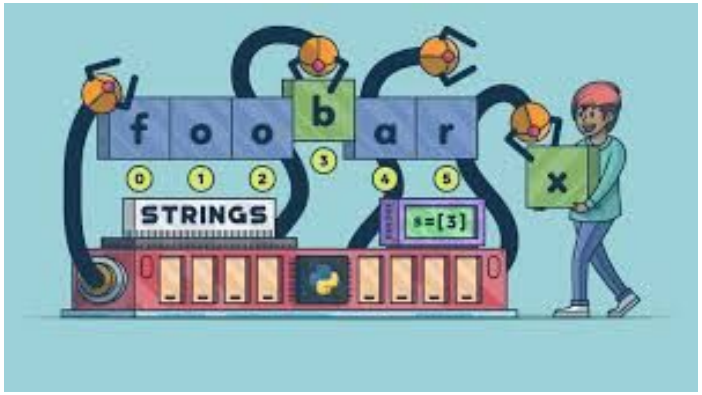
Руденков Александр Сергеевич

План занятия

1. Работа со строкой: len, in, str
2. Индексация строк
3. Срезы

Строки

Мы увидели, что Python отлично умеет работать с числами, но что, если мы хотим общаться с людьми? Люди лучше понимает слова и предложения, а не просто цифры. Чтобы написать программы, которыми смогут пользоваться люди, нам потребуется другой тип переменных, известный под названием строки. Строки - это то, как мы называем текст или клавиатурные символы на языке программирования, иными словами, это группы букв, цифр и символов. Ваше имя - это строка, как и название вашего любимого цвета, даже этот абзац (или даже вся книга) - это длинная строка букв, пробелов, цифр, символов, перемешанных друг с другом.



Строки

Строковый тип данных очень часто используется в программировании. В Python строковый тип данных имеет название **str**. Команда **input()** считывает именно строку текста. Для задания пустой строки, мы используем две кавычки одинакового типа. **Нельзя путать пустую строку и строку состоящую из одного символа пробела. Это абсолютно разные строки !!!!**

```
str1 = ''
str2 = ' '
```

Длиной строки называется количество символов из которых она состоит. Чтобы посчитать длину строки используем встроенную функцию **len()**.

```
s = 'Привет, мир'
a = len(s)
print(a)
```

→ 11

Помните, что пробелы, запятые, все знаки, заключенные в кавычки, являются символами.

Строки

В Python есть специальный оператор **in**, который позволяет проверить, что одна строка находится внутри другой.

```
s = input()
if 'a' in s:
    print('Введенная строка содержит символ a')
else:
    print('Введенная строка не содержит символ a')
```

```
спасибо
Введенная строка содержит символ a
```

Мы можем использовать оператор **in** вместе с логическим оператором **not.**

```
s = input()
if '-' not in s:
    print('Введенная строка не содержит символа тире')
```

```
Привет
Введенная строка не содержит символа тире
```

Индексация

Очень часто бывает необходимо обратиться к конкретному символу в строке. Для этого в Python используются квадратные скобки **[]**, в которых указывается индекс (номер) нужного символа в строке.

0	1	2	3	4	5
P	y	t	h	o	n
-6	-5	-4	-3	-2	-1

Обратите внимание первый символ строки равен **s[0]**, а не **s[1]**. В Python индексация начинается с 0, по аналогии с функцией **range(n)**.

Если длина строки s равна **len(s)**, то при положительной нумерации слева направо, последний элемент имеет индекс равный **len(s) - 1**, а при отрицательной индексации справа налево, первый элемент имеет индекс равный **- len(s)**.

Перебор элементов строки

Строка - это набор символов, поэтому циклический проход по строке с помощью цикла **for** языка Python разобьет строку на отдельные символы. В примере ниже переменная **i** пройдет по всем символам в строковой переменной word.

```
word = 'abcdef'
for i in range(len(word)):
    print(word[i])
```

```
a
b
c
d
e
f
```

В нашем случае длина строки **word**, равна **6**. Таким образом, вызов функции **range(len(word))** имеет вид **range(6)** и переменная цикла **i** последовательно перебирает все значения от **0** до **5**. Это означает, что выражение **word[i]** последовательно вернет все символы строки **word**.

Если нам не нужен индекс самого символа, то мы можем использовать более короткий способ итерации:

```
word = 'abcdef'
for i in word:
    print(i)
```

```
a
b
c
d
e
f
```

Практика

1 На вход программе подается одна строка. Напишите программу, которая выводит элементы строки с индексами 1, 3, 5, ... в столбик.

2 На вход программе подаются три строки: имя, фамилия и отчество. Напишите программу, которая выводит инициалы человека. На вход программе подаются три строки, каждая на отдельной строке.

```
Петров
Иван
Сергеевич
Петров И.С.
```

3 На вход программе подается одна строка состоящая из цифр. Напишите программу, которая считает сумму цифр данной строки.

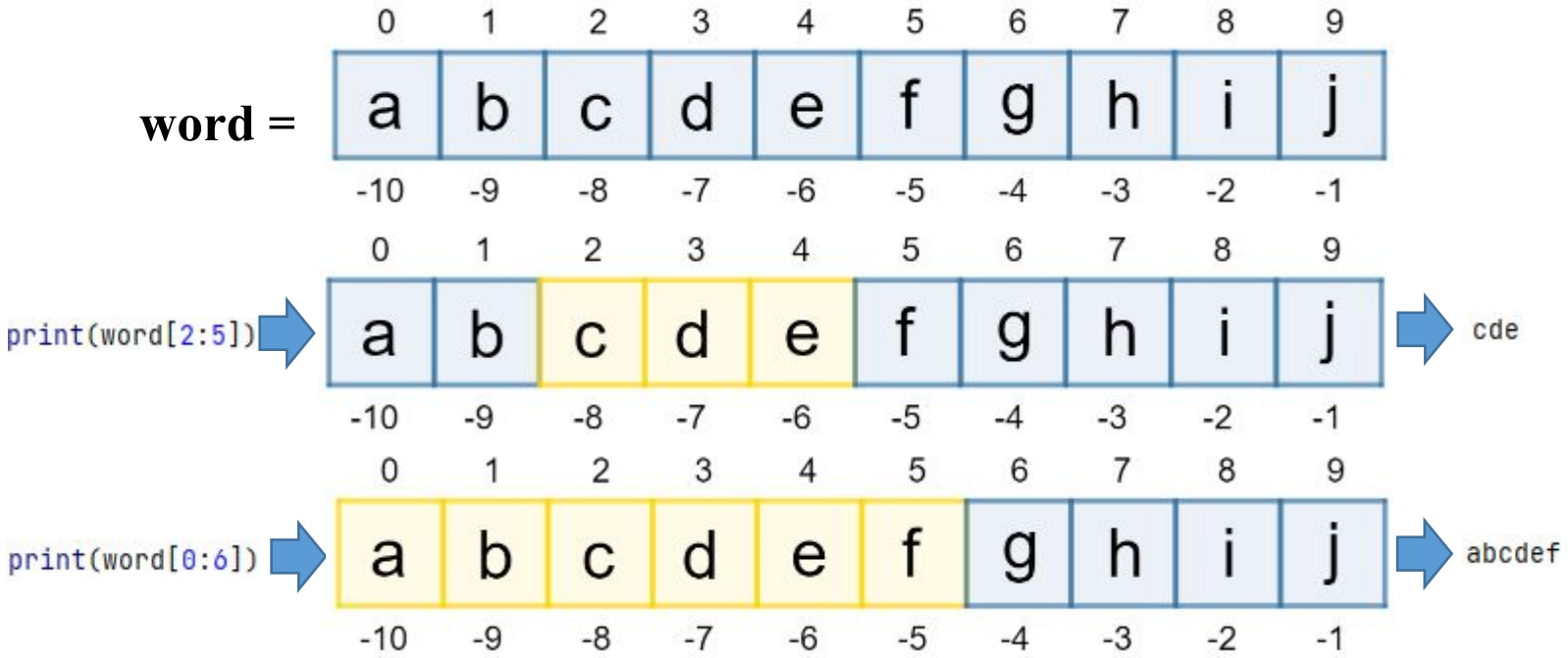
4 На вход программе подается одна строка. Напишите программу, которая определяет сколько в ней одинаковых соседних символов.

```
aabbccdd
4
```

ДЗ На вход программе подается одна строка. Напишите программу, которая определяет сколько раз в строке встречаются символы ! и -.

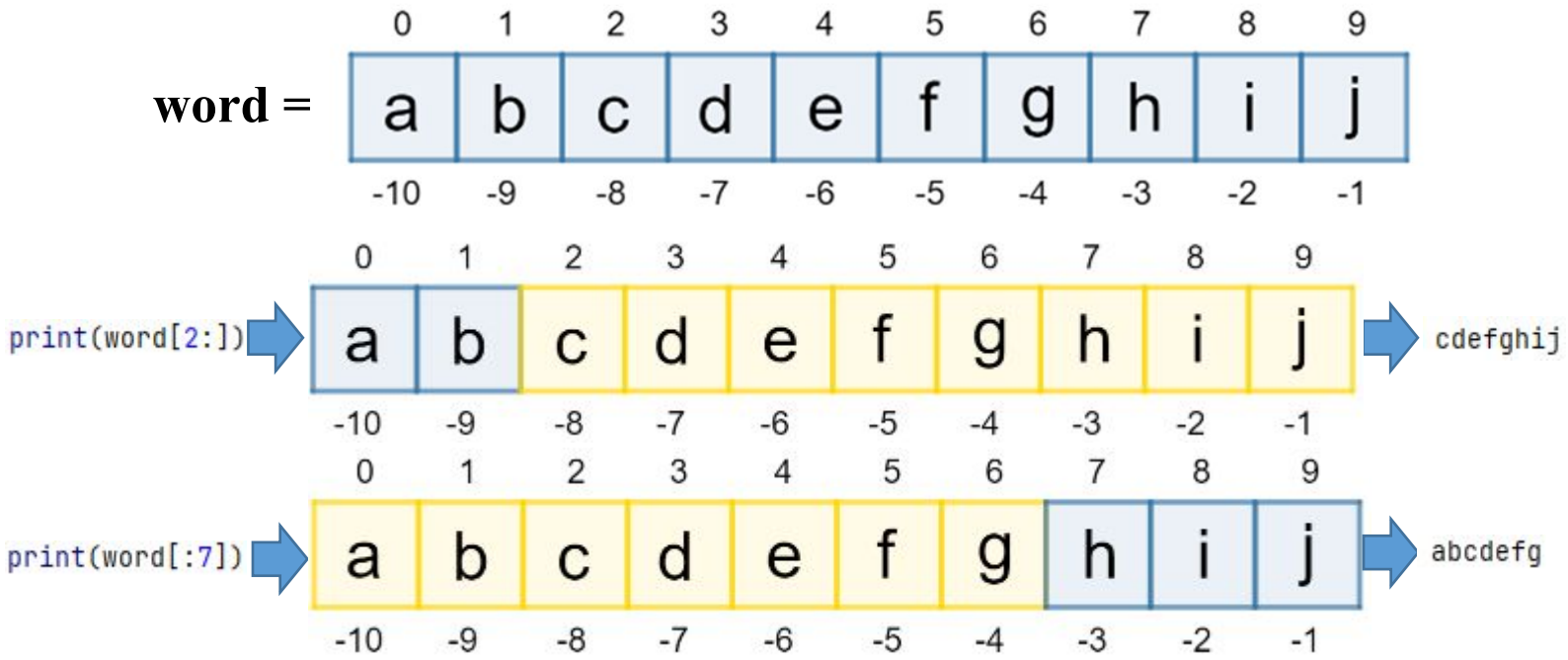
Срезы

Иногда нужно бывает работать с целыми частями строки, в таком случае мы используем срезы (**slices**). Срезы похожи на комбинацию индексации и функции **range()**. С помощью среза мы можем получить несколько символов исходной строки, создав диапазон индексов разделенных двоеточием **word[x:y]**. При построении среза **word[x:y]** первое число – это то место, где начинается срез (включительно), а второе – это место, где заканчивается срез (невключительно).



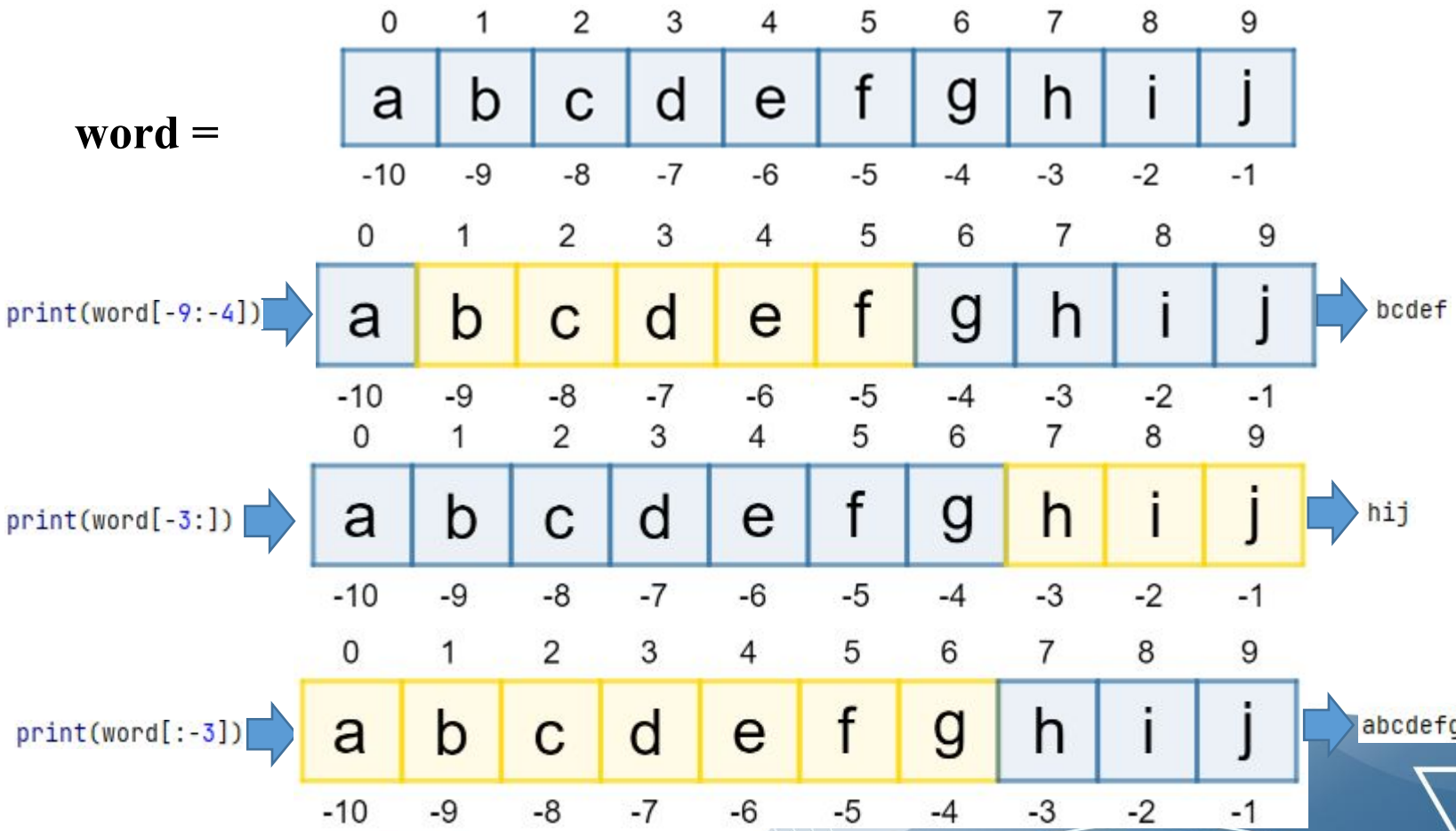
Срезы

Если опустить второй параметр в срезе word[x:] (но поставить двоеточие), то срез берется до конца строки. Аналогично если опустить первый параметр word[:y], то можно взять срез от начала строки. Срез word[:] совпадает с самой строкой word.



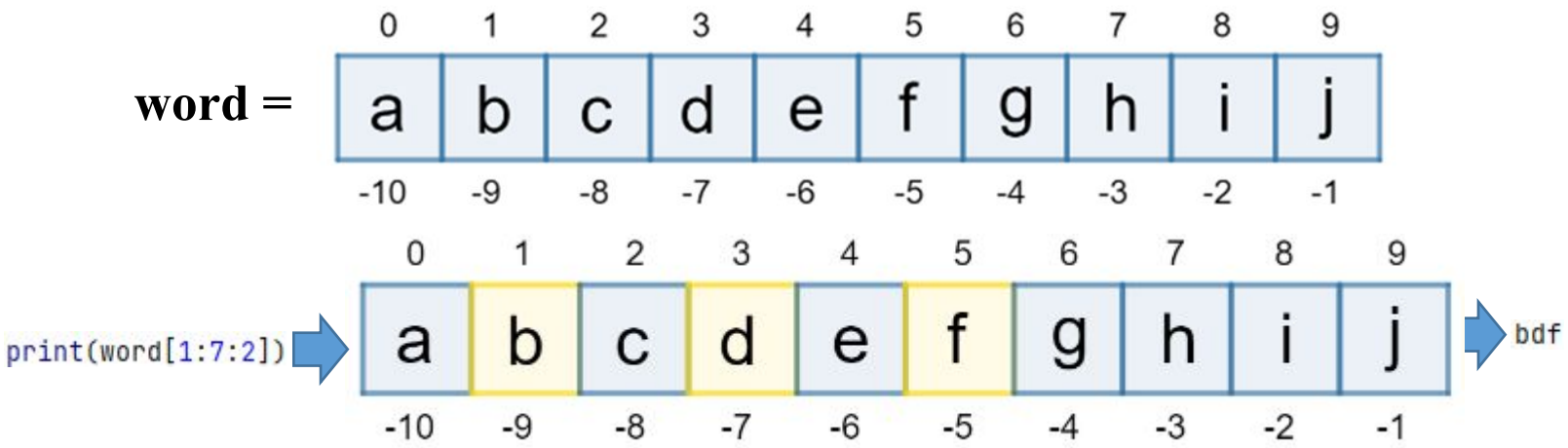
Срезы

Мы также можем использовать отрицательные индексы для создания срезов. Как уже говорилось ранее, отрицательные индексы строки начинаются с -1 и отсчитываются до достижения начала строки. При использовании отрицательных индексов первый параметр среза должен быть меньше второго, либо должен быть пропущен.



Срезы

Мы можем передать в срез третий необязательный параметр, который отвечает за шаг среза. К примеру, срез **word[1:7:2]** создаст строку **bdf** состоящую из каждого второго символа (индексы **1, 3, 5**, правая граница не включена в срез).



Срезы

Предположим, у нас есть строка **word = 'abcdefghij'** и мы хотим заменить символ с индексом 3 на 'X'.

word =

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Можно попытаться написать код:

```
word[3] = 'X'
```

```
word[3] = 'X'
TypeError: 'str' object does not support item assignment
```

Однако такой код не работает. В Python строки являются неизменяемыми, то есть мы не можем менять их содержимое с помощью индексатора. Если мы хотим поменять какой-либо символ строки **word**, мы должны создать новую строку. Следующий код использует срезы и решает поставленную задачу:

```
word = word[:3] + 'X' + word[4:]
print(word)
```

```
abcXefghij
```

Практика

1 Используя срезы, обработайте строку `s = "Компьютерная Академия ШАГ - лучший учебный центр"` **следующим образом:**

а) выведите первые 13 символов

б) выведите последние 9 символов строки

в) выведите каждый 7 символ строки s начиная от начала строки

г) выведите строку в обратном порядке

д) строку с удаленным первым и последним символом

2 На вход программе подается одно слово, записанное в нижнем регистре. Напишите программу, которая определяет является ли оно палиндромом. Программа должна вывести «YES», если слово является палиндромом и «NO» в противном случае.

3 На вход программе подается строка текста. Напишите программу, которая разрежет ее на две равные части, переставит их местами и выведет на экран.

Практика

ДЗ На вход программе подается одна строка, длина которой больше 5 символов. Напишите программу, которая выводит :

- **второй символ этой строки;**
- **предпоследний символ этой строки;**
- **первые 4 символа этой строки;**
- **всю строку, кроме последних трех символов;**
- **все символы с четными индексами;**
- **все символы с нечетными индексами;**
- **все символы в обратном порядке;**
- **все символы строки через один в обратном порядке, начиная с последнего.**

ООП НА PYTHON

Урок 8. Строки. Методы строк

Руденков Александр Сергеевич

План занятия

1. Работа со строкой: len, in, str
2. Индексация строк
3. Срезы

Основные методы строк

Метод **s.title()** возвращает строку, первый символ которой в верхнем регистре

```
s = "компьютерная Академия Шаг"
print(s.title())
```

→ Компьютерная Академия Шаг

Метод **capitalize()** возвращает копию строки **s**, в которой первый символ имеет верхний регистр, а все остальные символы имеют нижний регистр.

```
s = "Компьютерная Академия Шаг"
print(s.capitalize())
```

→ Компьютерная академия шаг

Метод **s.upper()** возвращает строку в верхнем регистре

```
s = "Компьютерная Академия Шаг"
print(s.upper())
```

→ КОМПЬЮТЕРНАЯ АКАДЕМИЯ ШАГ

Метод **s.lower()** возвращает строку в нижнем регистре

```
s = "Компьютерная Академия Шаг"
print(s.lower())
```

→ компьютерная академия шаг

Метод **s.swapcase()** возвращает строку с противоположными регистрами символов

```
s = "Компьютерная Академия Шаг"
print(s.swapcase())
```

→ кОМПЬЮТЕРНАЯ аКАДЕМИЯ шАГ

Основные методы строк

Метод **s.count()** возвращает количество подстрок в интервале либо -1

```
s = "Компьютерная Академия Шаг"
print(s.count('Шаг', 15, 35))
```

1

Метод **s.startswith()** проверяет, начинается ли строка с символа, переданного в качестве аргумента методу

```
s = "Компьютерная Академия Шаг"
print(s.startswith('К'))
```

True

Метод **s.endswith()** возвращает строку в верхнем регистре

```
s = "Компьютерная Академия Шаг"
print(s.endswith('ar'))
```

True

Метод **s.find()** находит индекс первого вхождения подстроки в исходной строке **s**. Если строка **s** не содержит подстроки, то метод возвращает значение **-1**.

```
s = "Компьютерная Академия Шаг"
print(s.find('Шаг'))
```

22

Метод **rfind()** идентичен методу **find()**, за тем исключением, что он ищет первое вхождение подстроки начиная с конца строки **s.**

Метод **index()** идентичен методу **find()**, за тем исключением, что он вызывает ошибку **ValueError: substring not found** во время выполнения программы, если подстрока не найдена.

Метод **rindex()** идентичен методу **index()**, за тем исключением, что он ищет первое вхождение подстроки начиная с конца строки s.

Основные методы строк

Метод **s.strip()** очищает от символа переноса строки (\n) и пробелов.
Метод **lstrip()** возвращает копию строки s у которой удалены все пробелы стоящие в начале строки.
Метод **rstrip()** возвращает копию строки s у которой удалены все пробелы стоящие в конце строки.
Метод **replace()** возвращает копию s со всеми вхождениями подстроки, замененными на новую подстроку.

```
s = "Компьютерная Академия Шаг"
print(s.replace('Шаг', 'STEP'))
```

```
Компьютерная Академия STEP
```

Метод **replace()** может принимать опциональный третий аргумент, который определяет количество замен.

Основные методы строк

Метод **isalnum()** определяет, состоит ли исходная строка из буквенно-цифровых символов. Метод возвращает значение True если исходная строка является непустой и состоит ***только*** из буквенно-цифровых символов и False в противном случае.

```
s = "Компьютерная Академия Шаг"
print(s.isalnum())
```

→ False

Метод **isalpha()** определяет, состоит ли исходная строка из буквенных символов. Метод возвращает значение True если исходная строка является непустой и состоит ***только*** из буквенных символов и False в противном случае.

```
s = "Компьютерная Академия Шаг"
print(s.isalpha())
```

→ False

Метод **isdigit()** определяет, состоит ли исходная строка ***только*** из цифровых символов. Метод возвращает значение True если исходная строка является непустой и состоит ***только*** из цифровых символов и False в противном случае.

```
s = "123456"
print(s.isdigit())
```

→ True

Метод **s.isupper()** проверяет, написаны ли все символы в верхнем регистре.
Метод **s.islower()** проверяет, написаны ли все символы в нижнем регистре
Метод **s.istitle()** проверяет, начинается ли строка с большой буквы

Практика

1 На вход программе подается строка текста. Напишите программу, которая определяет является ли оттенок текста хорошим или нет. Текст имеет хороший оттенок, если содержит подстроку «хорош» во всевозможных регистрах.

```
оыралоывало ХОРОШвмсва выарлво83кг834
YES
```

2 На вход программе подается строка. Напишите программу, которая подсчитывает количество буквенных символов в нижнем регистре.

```
фывыи34ВАПРО
5
```

3 На вход программе подается строка текста, состоящая из слов, разделенных ровно одним пробелом. Напишите программу, которая подсчитывает количество слов в ней.

```
Python is the best language
5
```

4 На вход программе подается строка текста. Напишите программу, которая выводит на экран символ, который появляется наиболее часто.

```
asddffccbbbbbb
b
```

5 На вход программе подается строка текста, в которой буква «z» встречается минимум два раза. Напишите программу, которая удаляет из этой строки первое и последнее вхождение буквы «z», а также все символы, находящиеся между ними.

```
abcdzxyz
abcd
```

Практика

6 На вход программе подается строка текста в которой буква «z» встречается как минимум два раза. Напишите программу, которая возвращает исходную строку и переворачивает последовательность символов, заключенную между первым и последним вхождением буквы «z».

```
adfgvzfghjjtz
adfgvztjjhgfz
```

7 На вход программе подается строка текста. Напишите программу, которая проверяет, что строка заканчивается подстрокой .com или .ru. Программа должна вывести «YES» если введенная строка заканчивается подстрокой .com или .ru и «NO» в противном случае.

```
www.google.com
YES
```

8 На вход программе подается строка текста. Напишите программу, которая выводит на экран символ, который появляется наиболее часто. На вход программе подается строка текста. Текст может содержать строчные и заглавные буквы английского и русского алфавита, а также цифры.

```
asdfgvccd122AAAAaaa
a
```

ДЗ На вход программе подается строка текста. Напишите программу, которая подсчитывает количество цифр в данной строке.

ASCII

Каждая буква, число и символ при сохранении в памяти компьютера преобразуется в числовое значение. ASCII (American Standard Code for Information Interchange - Американский стандартный код для обмена информацией) - одна из наиболее популярных систем нумерации.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII

Функция **ord** позволяет определить код некоторого символа в таблице символов Unicode. Аргументом данной функции является одиночный символ.

```
s1 = ord('S')
s2 = ord('Ы')
s3 = ord('s')
print(s1, s2, s3)
```

```
83 1067 115
```

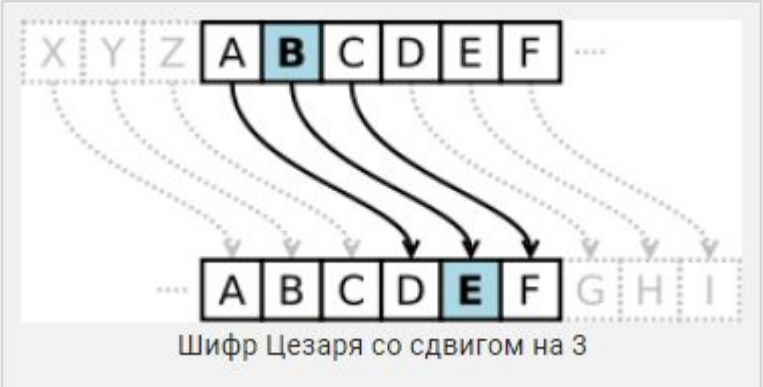
Функция **chr** позволяет определить по коду символа сам символ. Аргументом данной функции является численный код.

```
ch1 = chr(65)
ch2 = chr(75)
ch3 = chr(110)
print(ch1, ch2, ch3)
```

```
A K n
```

Кодирование/декодирование

Шифр Цезаря (шифр сдвига) - один из самых простых и наиболее широко известных методов шифрования. Шифр Цезаря - это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.



Если сопоставить каждый символ алфавита с его порядковым номером (нумеруя с 0), то шифрование и дешифрование можно выразить формулами модульной арифметики:

$y=(x+k) \% n,$

$x=(y-k) \% n,$

где x - символ открытого текста, y символ шифрованного текста, n - мощность алфавита (количество символов), а k - ключ.

Кодирование/декодирование

Используемое преобразование в шифре Цезаря обычно обозначают как ROT N, где N - сдвиг, ROT - сокращение от слова ROTATE, в данном случае «циклический сдвиг». Например, обозначение ROT 2 обозначает сдвиг на 2 позиции, то есть, «а» превращается в «в», «б» в «г», и так далее, и в конце «ю» превращается в «а», а «я» - в «б».

Число разных преобразований зависит от длины алфавита:

- для русского языка возможно 32 разных преобразования (преобразования ROT 0 и ROT 33 сохраняют исходный текст, а дальше начинаются уже повторения);
- для английского языка возможны 25 разных преобразований (преобразования ROT 0 и ROT 26 сохраняют исходный текст, а дальше начинаются уже повторения).

Неалфавитные символы - знаки препинания, пробелы, цифры - не меняются.

Шифр Цезаря

```
message = input("Enter a message : ")
key = int(input("Enter a key value from 1-26: "))
process = input("encode or decode: ")
message = message.upper()
output = ""
for letter in message:
    if letter.isupper():
        if process == 'encode':
            value = ord(letter) + key
            letter = chr(value)
            if not letter.isupper():
                value -= 26
                letter = chr(value)
        else:
            value = ord(letter) - key
            letter = chr(value)
            if not letter.isupper():
                value += 26
                letter = chr(value)
    output += letter
print ("Output message: ", output.lower())
```