Лабораторная работа №4:
«Работа со строками».

Теоретическое введение

Символьная строка – это последовательность символов, расположенных в памяти рядом (в соседних ячейках). Для работы с символами во многих языках программирования есть переменные специального типа: символы, символьные массивы и символьные строки (которые, в отличие от массивов, рассматриваются как цельные объекты). Основной тип данных для работы с символьными величинами в языке Python – это символьные строки (тип string). Для того, чтобы записать в строку значение, используют оператор присваивания

```
s='Вася пошел гулять'
```

Строка заключается в кавычки или в одиночные апострофы. Если строка ограничена кавычками, внутри неё могут быть апострофы, и наоборот. Для ввода строки с клавиатуры применяют функцию input ():

```
s = input( "Введите имя: " )
```

Для того, чтобы выделить отдельный символ строки, к нему нужно обращаться так же, как к элементам массива (списка): в квадратных скобках записывают номер символа. Например, так можно вывести на экран символ строки s с индексом 5 (длина строки в этом случае должна быть не менее 6 символов):

```
print ( s[5] ).
```

Отрицательный индекс говорит о том, что отсчёт ведётся с конца строки. Например, s[-1] означает то же самое, что s[len(s)-1], то есть последний символ строки. В отличие от большинства современных языков программирования, в Python нельзя изменить символьную строку, поскольку строка – это неизменяемый объект. Поэтому оператор присваивания s[5] = "a" не сработает – будет выдано сообщение об ошибке.

Для того, чтобы выделить часть строки (подстроку) применяется операция получения среза (англ. slicing), например s[3:8] означает символы строки s с 3-го по 7-й (то есть до 8-го, не включая его).

Следующий фрагмент копирует в строку s1 символы строки s с 3-го по 7- й (всего 5 символов):

```
s = "0123456789"
s1 = s[3:8]
```

В строку s1 будет записано значение "34567".

Для удаления части строки нужно составить новую строку, объединив части исходной строки до и после удаляемого участка:

```
s = "0123456789"
s1 = s[:3] + s[9:]
```

Срез s[:3] означает «от начала строки до символа s[3], не включая его», а запись s[9:] – «все символы, начиная с s[9] до конца строки». Таким образом, в переменной s1 остаётся значение "0129".

С помощью срезов можно вставить новый фрагмент внутрь строки:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
```

Переменная s получит значение "012ABC3456789".

Если заданы отрицательные индексы, к ним добавляется длина строки N. Таким образом, индекс «–1» означает то же самое, что N-1. При выполнении команд

```
s = "0123456789"
s1 = s[:-1] # "012345678"
s2 = s[-6:-2] # "4567"
```

мы получим s1 = "012345678" (строка s кроме последнего символа) и s2 = "4567". Срезы позволяют выполнить реверс строки (развернуть её наоборот):

```
s1 = s[::-1]
```

Так как начальный и конечный индексы элементов строки не указаны, задействована вся строка. Число «–1» означает шаг изменения индекса и говорит о том, что все символы перебираются в обратном порядке.

Пример обработки строк

Вводится строка, содержащая имя, отчество и фамилию человека, например:

Василий Алибабаевич Хрюндиков.

Каждые два слова разделены одним пробелом, в начале строки пробелов нет. В результате обработки должна получиться новая строка, содержащая фамилию и инициалы:

Хрюндиков В.А.

Описание алгоритма

1. Ввести строку s
2. Найти в строке s первый пробел
3. Имя = всё, что слева от первого пробела
4. Удалить из строки s имя с пробелом
5. Найти в строке s первый пробел
6. Отчество = всё, что слева от первого пробела
7. Удалить из строки s отчество с пробелом # осталась фамилия
8. s = s + " " + имя[0] + "." + отчество[0] + "."

Листинг программы

Вариант 1

```
print ( "Введите имя, отчество и фамилию:" )
```

```
    s = input()
    n = s.find ( " " )
    name = s[:n] # вырезать имя
    s = s[n+1:]
    n = s.find ( " " )
    name2 = s[:n] # вырезать отчество
    s = s[n+1:] # осталась фамилия
    s = s + " " + name[0] + "." + name2[0] + "."
    print ( s )
```

Вариант2

```
print ( "Введите имя, отчество и фамилию:" )
s = input()
fio = s.split()
s = fio[2] + " " + fio[0][0] + "." + fio[1][0] + "."
print ( s )
```

Задания к лабораторной работе №4 «Работа со строками».

Выделить в строке-предложении *s* все слова, разделенные символами-разделителями «_.,;:\n\t!?». Обработать выделенные слова в соответствии с вариантом задания.

Регулярное слово – слово, состоящее только из больших латинских букв.
Палиндром – это слово, которое одинаково читается слева направо и справа налево.

1. Подсчитать количество слов, начинающихся на большую букву и содержащих хотя бы один арифметический знак. Напечатать все слова, содержащие две рядом стоящие одинаковые буквы.
2. Подсчитать количество регулярных слов. Напечатать в перевернутом виде все слова, которые содержат два экземпляра заданного символа.
3. Напечатать все слова, начинающиеся с большой буквы. Напечатать самое длинное регулярное слово, которое состоит из одинаковых символов.
4. Напечатать слово, содержащее наибольшее количество цифр. Напечатать количество слов, содержащих хотя бы два арифметических знака.
5. Напечатать все регулярные слова. Напечатать в перевернутом виде самое длинное слово, состоящее только из цифр и латинских букв.

6. Найти количество слов, содержащих более одной цифры, и, исключив все арифметические знаки из этих слов, напечатать их. Напечатать в порядке возрастания все числа, встретившиеся в словах.
7. Определить количество слов, которые содержат заданное подслово и хотя бы одну цифру, и напечатать их. Напечатать в порядке убывания все числа, встретившиеся в словах.
8. Определить количество слов, содержащих и буквы, и цифры, и арифметические знаки. Напечатать их. Напечатать все симметричные слова, содержащие наибольшее количество цифр.
9. Подсчитать количество регулярных слов, содержащих хотя бы две одинаковые буквы. Напечатать все слова, имеющие одну цифру, удалив из таких слов все арифметические знаки.
10. Найти самое длинное регулярное слово и удалить из него все гласные буквы. Найти все слова, в которых имеются либо только цифры, либо только латинские буквы.
11. Подсчитать количество слов, состоящих из одинаковых букв или одинаковых цифр. Напечатать в перевернутом виде слова, имеющие хотя бы один арифметический знак.
12. Напечатать все слова, которые начинаются с большой буквы и заканчиваются заданным двухбуквенным подсловом. Определить количество слов, содержащих согласные латинские буквы, и напечатать порядковые номера этих слов.
13. Напечатать все слова, имеющие в своем составе согласные латинские буквы. Определить количество слов, которые не имеют в своем составе ни одной цифры, и напечатать эти слова.
14. Напечатать все симметричные слова, предварительно удалив из них цифры. Напечатать все слова, состоящие только из согласных латинских букв.
15. Найти все слова, содержащие числа в диапазоне от 10 до 100, и подсчитать их сумму. Напечатать слова, не имеющие цифр, предварительно удалив арифметические знаки.
16. Подсчитать количество слов, начинающихся с большой буквы и оканчивающихся цифрой. Напечатать слова, содержащие заданное подслово и хотя бы один арифметический знак.
17. Подсчитать количество слов, содержащих хотя бы одну согласную латинскую букву и хотя бы одну цифру. Напечатать все слова, состоящие только из четных цифр, и подсчитать сумму этих цифр.

18. Напечатать все слова, которые содержат хотя бы один арифметический знак и заканчиваются на цифру. Определить количество слов, содержащих все маленькие латинские гласные буквы.
19. Найти количество симметричных регулярных слов и напечатать их. Напечатать в перевернутом виде все слова, содержащие согласные латинские буквы.
20. Найти и напечатать все слова, содержащие наибольшее количество букв, если только буквы расположены в алфавитном порядке. Подсчитать количество симметричных слов, имеющих более двух арифметических знаков.
21. Подсчитать количество слов, начинающихся на большую букву и содержащих хотя бы один арифметический знак. Напечатать все слова, содержащие две рядом стоящие одинаковые буквы.
22. Подсчитать количество регулярных слов. Напечатать в перевернутом виде все слова, которые содержат два экземпляра заданного символа.
23. Напечатать все слова, начинающиеся с большой буквы. Напечатать самое длинное регулярное слово, которое состоит из одинаковых символов.
24. Напечатать слово, содержащее наибольшее количество цифр. Напечатать количество слов, содержащих хотя бы два арифметических знака.
25. Напечатать все регулярные слова. Напечатать в перевернутом виде самое длинное слово, состоящее только из цифр и латинских букв.
26. Найти количество слов, содержащих более одной цифры, и, исключив все арифметические знаки из этих слов, напечатать их. Напечатать в порядке возрастания все числа, встретившиеся в словах.
27. Определить количество слов, которые содержат заданное подслово и хотя бы одну цифру, и напечатать их. Напечатать в порядке убывания все числа, встретившиеся в словах.
28. Определить количество слов, содержащих и буквы, и цифры, и арифметические знаки. Напечатать их. Напечатать все симметричные слова, содержащие наибольшее количество цифр.
29. Подсчитать количество регулярных слов, содержащих хотя бы две одинаковые буквы. Напечатать все слова, имеющие одну цифру, удалив из таких слов все арифметические знаки.
30. Найти самое длинное регулярное слово и удалить из него все гласные буквы. Найти все слова, в которых имеются либо только цифры, либо только латинские буквы.

ПРИЛОЖЕНИЕ. Функции и методы для работы со строками

Встроенные функции	**Описание и пример использования**
chr(<код_символа>)	возвращает символ по коду символа
ord(<символ>)	возвращает код символа `>>ord( '3' ) # вернет 51`
len(<s>)	возвращает длину строки s `>>len('Python') # Вернет 6`
str(<obj>)	создает строку из объекта obj
Методы класса str	
dir(str)	Вывод всех методов
str.count(<sub>)	возвращает сколько раз строка sub встречается в строке
str.endswith(<sub >)	Возвращает True если строка кончается на sub, иначе возвращает False.
str.find(<sub> [,<start> [,<end>]])	возвращает первую слева позицию, на которой находится строка sub, если не находит, то возвращает -1, поиск начинается с позиции start и до поз. end, если это указано, иначе поиск начинается с начала.
str.rfind(<sub> [,<start> [,<end>]])	возвращает первую справа позицию, на которой находится строка sub, если не находит, то возвращает -1, поиск начинается с позиции start и до позиции end, если это указано, иначе поиск начинается с начала
str.index(<sub> [,<start> [,<end>]])	работает так же как find, но если подстрока не найдена, вызывает исключение и останавливает программу

str.join(<iter>)	объединяет word с iter
	`>>> '*'.join( word ) #Здесь основная строка '*'`
	`'P*y*t*h*o*n'`

str.startswith(<sub>)	возвращает True, если строка начинается с sub и False, если нет

str.upper()	создает строку из исходной в верхнем регистре
	`>>> word.upper()`
	`'PYTHON'`

str.lower()	создает строку из исходной в нижнем регистре
	`>>> word.lower()`
	`'python'`

str.swapcase()	создает строку, где верхний регистр символов заменен на нижний и наоборот.
	`>>> word.swapcase()`
	`'pYTHON'`

str.strip()	создает строку без пробелов слева и справа.
str.lstrip()	создает строку без пробелов слева.
str.rstrip()	создает строку без пробелов справа.
str.partition(<sep>)	создает три строки, первая часть строки до

	разделителя sep, потом строка разделитель sep и третья часть после разделителя.
	`>>> s1,s2,s3 = word.partition('t')`
	`>>> s1,s2,s3`
	`('Py', 't', 'hon')`

str.rpatition(<sep>)	делает тоже самое, но разделитель ищется справа.
str.split([<sep>])	возвращает несколько строк, разделенных sep. Если sep не указан, то строки разделяются по пробелам, а пустые строки (строки не содержащие символы или содержащие только пробелы) не возвращаются.
str.rsplit([<sep>])	делает тоже самое, но обрабатывает строку справа налево.

str.replace(<old> ,<new> [,<count>])	- заменяет все old в строке word на new и создает новую строку. Если указан параметр count, то замена производится count раз. count должно содержать число.
	`>>> word.replace( 'P','J')`
	`'Jython'`
	`>>> word #Обратите внимание, что оригинальная строка не изменилась!!!`
	`'Python'`