МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
по курсу

«ИНФОРМАТИКА»

Часть III

2021

Лабораторная работа № 3
Символы и строки. Текстовые файлы

Символы

В Python нет отдельного типа для символов. Даже если присвоить переменной значение ‘a’, она будет иметь строковый тип.

Строковый тип предоставляет программисту весь нужный функционал для работы как со строками, так и с символами.

```
ord(ch)      – выдает номер символа (нумерация с нуля):
              print(ord('0'));   // выводит  48
chr(k)       – выдает k-ый символ из таблицы символов
              print(chr(71));   // выводит символ "G"
```

Любой символ в Python является единичной строкой, что позволяет использовать для работы с ним те же функции, что и для строк.

```
s = "A"
s = "F"
print(s) # на экране F
```

Строка — это неизменяемая последовательность!

```
s = "A"
s[0] = "F"
print(s)
```

```
s[0] = "F"
  ^
TypeError: 'str' object does not support item assignment
```

Строки

Строка в Python - упорядоченная **неизменяемая** последовательность символов, используемая для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме

Размерность и состав созданной однажды неизменяемой последовательности **не может меняться**, вместо этого обычно **создаётся новая последовательность.**

Строка заключена в кавычки.

Создание строк

1. С помощью одинарных и двойных кавычек. Строки в одинарных и двойных кавычках - одно и то же. Причина наличия двух вариантов в том, чтобы позволить вставлять в строки символы кавычек, не используя экранирование

```
s1 = 'Текст в одинарных кавычках'
s2 = "Текст в двойных кавычках"
s3 = 'Слово "Сессия" обычно подразумевает проблемы'
s4 = "I'm learning Python"
```

2. С помощью тройных кавычек.

Тройные кавычки можно использовать для записи многострочных блоков текста. Внутри такой строки возможно присутствие кавычек и апострофов, главное, чтобы не было трех кавычек подряд.

```
s5 = '''Это очень длинная
       строка, ей нужно
       много места'''
```

3. С помощью ввода с клавиатуры

```
s6 = input()
```

4. С помощью ввода из файла

```
f = open('test.txt')
s7 = f.readline()
```

Над строками определены следующие функции

Метод	Описание	Пример

x in s	Если элемент присутствует в последовательности, то возвращает True, иначе - False	`# Количество букв "А" в строке s`
		`for c in s:`
		`  if c=="A":`
		`  k+=1`

s + t	Конкатенация(сложение) двух последовательностей	`s = "Скоро "`
		`t = "сессия"`
		`print( s + t) #s=«Скоро сессия"`

s * n	Эквивалентно сложению последовательности s с собой n раз	`s = "w"*10` `print(s) #s="wwwwwwwwww"`

s[i]	Возвращает i-й элемент последовательности	`s = input()`
		`for i in len(s):`
		`  if s[i]==s[i+1]...`

s[i, j]	Возвращает набор элементов последовательности с индексами из диапазона i <= k < j	`s = "математика"`
		`s = s[2:6]`
		`# s = "тема"`

min(s)	Минимальный элемент последовательности	`s = "математика"`
		`print(min(s))`
		`# выведет «а»`

max(s)	Максимальный элемент последовательности	`s = "математика"`
		`print(max(s))`
		`# выведет «т»`

len(s)	Длина последовательности	`s = "математика"`
		`print(len(s))`
		`# выведет 10`

s.index(x)	Возвращает индекс подстроки x в строке s	`s = "математика"`
		`print(s.index("и"))`
		`# выведет 7`

s.count(x)	Число вхождений подстроки x в строку s	`s = "математика"`
		`print(s.count("а"))`
		`# выведет 3`

Срезы строк

Срез (slice) — извлечение из данной строки одного символа или некоторого фрагмента подстроки или подпоследовательности.

Индекс - номер символа в строке (а также в других структурах данных: списках, кортежах).

Нумерация начинается с 0.

Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера -1.

Любые операции среза со строкой **создают новые строки** и **никогда не меняют исходную строку**.

В Питоне строки являются **неизменяемыми**.

Можно лишь в старую переменную присвоить новую строку.

```
>>> str = 'Hello'
```

1. Срез с одним параметром - взятие одного символа строки

```
>>> str[0]   'H'
```

2. Срез с двумя параметрами s[a:b] возвращает подстроку, начиная с символа с индексом **a** до символа с индексом **b**, не включая его.

Если опустить второй параметр (но поставить двоеточие), то срез берется до конца строки.

```
>>> str[0:4] 'Hell'
>>> str[0:5] 'Hello'
>>> str[1:3] 'el'
>>> str[1:]  'ello'
>>> str[0:]  'Hello'
```

3. Срез с тремя параметрами - s[a:b:d].

Третий параметр задает шаг(как в случае с функцией **range**), то есть будут взяты символы с индексами **a, a + d, a + 2 * d** и т. д.

Например, при задании значения третьего параметра, равному **2**, в срез попадет каждый второй символ

```
>>> str[0:5:1] 'Hello'
>>> str[::1]   'Hello'
>>> str[0:5:2] 'Hlo'
>>> str[::2]   'Hlo'
```

Файлы

В зависимости от способа объявления можно выделить два вида файлов Python:

- бинарные файлы;
- текстовые файлы.

Следует помнить, что физические файлы на магнитных дисках и переменные файлового типа в программе на Python – объекты различные. Переменные файлового типа в Python могут соответствовать не только физическим файлам, но и логическим устройствам, связанным с вводом/выводом информации. Например, клавиатуре и экрану соответствуют файлы со стандартными именами Input, Output.

Такие файлы в программировании называются логическими. Основное назначение логического файла - обеспечить программе средства для обмена данными с периферийными устройствами компьютера. В связи с этим вводится понятие физического файла, как совокупности данных во внешней памяти (дискета, жесткий диск, магнитная лента и др.) Кроме того, в качестве физического файла могут использоваться и сами периферийные устройства, например, принтер).

Бинарные файлы

В бинарных файлах данные отображаются в закодированной форме (с использованием только нулей (0) и единиц (1) вместо простых символов).

В большинстве случаев это просто последовательности битов.

Они хранятся в формате .bin.

Текстовые файлы

В них хранятся последовательности символов. Блокнот и другие стандартные редакторы умеют читать и редактировать этот тип файлов.

Текст может храниться в двух форматах: (.txt) — простой текст и (.rtf) — «формат обогащенного текста».

Текстовые файлы являются *последовательными* файлами.

Чтение файла производится только с начала, каждый раз считывается только одна текущая компонента (символ, строка или число), причем нельзя прочитать текущую компоненту, не прочитав предыдущей. Запись в файл осуществляется только присоединением очередной компоненты к его концу.

Текстовые файлы обеспечивают взаимодействие между вычислительными системами и пользователями.

Текстовые файлы – это файлы в которых:

а) информация представлена в текстовом виде посредством символов из набора ASCII;

б) информация может разделяться на строки произвольной длины. Признаком конца строки служат два специальных символа #10 и #13;

в) в конце файла присутствует символ #26;

г) при записи чисел, строк и логических значений они преобразуются в символьный (текстовый вид);

д) при чтении чисел они автоматически преобразуются из текстового в машинное представление.

Действия с файлом:

1. Открыть
2. Записать/Прочитать
3. Закрыть

Открытие файла

1 способ — использование метода **open**:

```
f = open('test.txt')
```

после окончания работы с файлом его необходимо принудительно закрыть:

```
f.close()
```

2 способ — использование метода with, которая упрощает обработку исключений с помощью инкапсуляции начальных операций, а также задач по закрытию и очистке.

В таком случае инструкция close не нужна, потому что with автоматически закроет файл.

```
with open('test.txt') as f:
    # работа с файлом
```

```
f = open(file_name, access_mode)
```

где file_name = имя открываемого файла

access_mode = режим открытия файла. Он может быть: для чтения, записи и т. д. По умолчанию используется режим чтения (r), если другое не указано.

Режим	**Описание**	**Режим**	**Описание**
r	Только для чтения.	w+	Для чтения и записи. Создаст новый файл для записи, если не найдет с указанным именем.
w	Только для записи. Создаст новый файл, если не найдет с указанным именем.	wb+	Для чтения и записи (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
rb	Только для чтения (бинарный).	a	Откроет для добавления нового содержимого. Создаст новый файл для записи, если не найдет с указанным именем.
wb	Только для записи (бинарный). Создаст новый файл, если не найдет с указанным именем.	a+	Откроет для добавления нового содержимого. **Создаст новый файл** для чтения записи, если не найдет с указанным именем.
r+	Для чтения и записи.	ab	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
rb+	Для чтения и записи (бинарный).	ab+	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для чтения записи, если не найдет с указанным именем.

Закрытие файла

1 способ — использование метода close()

```
>>> f.close() # закрыть файл
```

2 способ — использование конструкции try/finally, которая гарантирует, что если после открытия файла операции с ним приводят к исключениям, он закроется автоматически. Без нее программа завершается некорректно.

```
>>> f = open('example.txt','r')
>>> try:
>>>     # работа с файлом
>>> finally:
>>>     f.close()
```

Файл нужно открыть до инструкции try, потому что если инструкция open сама по себе вызовет ошибку, то файл не будет открываться для последующего закрытия.

Этот метод гарантирует, что если операции над файлом вызовут исключения, то он закроется до того как программа остановится.

Примеры работы со строками символами

```
# вывести на экран таблицу ASCII – кодов
for i in range (256):
    print(i, chr(i))

# организовать «бесконечный» цикл
s = "*"
while s!="exit":
    print("Для выхода из цикла введите exit")
    s =  input()

# сформировать строку с заглавными латинскими буквами

lat=""
for i in range(ord("A"),ord("Z")+1):
    lat=lat + chr(i)

# получить первую и последнюю позиции слова в строке
def getWord (s, fp,lp):
    while (fp <len(s))and (not(s[fp] in lat)):
        fp = fp+1
    lp = fp
    while (lp<len(s)) and (s[lp] in lat):
        lp =lp+1
    lp =lp-1
    return [fp,lp]
```

Лабораторная работа № 3

Требования к лабораторной работе:

1. В программе:
- должны использоваться функции с параметрами;
- имя исходного и результирующего файла вводятся с клавиатуры. При вводе производится проверка на существование исходного файла.
2. Исходный файл с данными создается в простейшем текстовом редакторе.
3. Количество строк в исходном и результирующем файле должны совпадать! (кроме вариантов № 12, 24). Если в исходном файле имеются пустые строки, то они также должны присутствовать и в результирующем файле на тех же местах.

Задания к лабораторной работе № 3 (по вариантам)

1. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет выводить в результирующий файл последовательность гласных строчных английских букв (“a”, “e”, … ”y”) из входной последовательности, если гласных больше в этой строке, чем согласных и наоборот: выводить последовательность согласных букв, если их больше, чем гласных. Печать должна происходить в алфавитном порядке. Например, пусть в строках исходного файла содержатся следующие символы:
bbad, sdiv
aaagh yd

В этом случае в результирующем файле должно быть:
bdsv
ay

2. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет составлять и выводить в результирующий файл из тех цифр, которые не встречаются во входных данных, максимальное число. При составлении итогового числа каждая цифра может быть использована только один раз. Если во входных данных встречаются все цифры от 0 до 9, то следует вывести "-1". Например, пусть на вход подаются следующие символы:
Например, пусть в одной из строк исходного файла содержатся следующие символы:
173439
В этом случае в результирующем файле должно быть:
86520

3. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла исходного файла** будет составлять и выводить в результирующий файл слово из тех букв английского алфавита, которые не встречаются во входных данных ни как строчные, ни как прописные, причем буквы должны идти в алфавитном порядке. Каждая буква должна быть распечатана один раз. Буквы построенного слова должны быть прописными. Если во входных данных встречаются все буквы английского алфавита, то следует вывести строчными буквами слово "no".
Например, пусть в одной из строк исходного файла содержатся следующие символы:
absCDKLMNOPvwXYabcprst.
В этом случае в результирующем файле должно быть:
EFGHIJQUZ

4. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет составлять и выводить в результирующий файл из тех цифр, которые встречаются во входных данных, максимальное число. При составлении итогового числа каждая цифра может быть использована только один раз. Если во входных данных цифры не встречаются, то следует вывести "-1".
Например, пусть в одной из строк исходного файла содержатся следующие символы:
14ф73п439
лапд
В этом случае в результирующем файле должно быть:
97431
-1

5. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу на языке Паскаль или Бейсик, которая **для каждой строки исходного файла** будет определять и выводить в результирующий файл английскую букву, встречающуюся в этой строке реже всего (но не нулевое количество), и количество там таких букв. Строчные и прописные буквы при этом считаются не различимыми. Если искомых букв несколько, то программа должна выводить на экран первую из них по алфавиту.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
It is a task for you. Yes!
В этом случае в результирующем файле должно быть:
F 1

6. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет выводить в результирующий файл последовательность цифр (`'0','1'..'9'`) из входной последовательности и, через пробел, частот их повторения. Печать должна происходить в порядке возрастания.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
54533526.
В этом случае в результирующем файле должно быть:
2 1, 3 2, 4 1, 5 3, 6 1

7. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет составлять и выводить в результирующий файл из тех цифр, которые не встречаются во входных данных, минимальное число. При составлении итогового числа каждая цифра может быть использована только один раз (первый 0 не выводить). Если во входных данных встречаются все цифры от 0 до 9, то следует вывести "-1". Например, пусть на вход подаются следующие символы:
Например, пусть в одной из строк исходного файла содержатся следующие символы:
173439.
В этом случае в результирующем файле должно быть:
2568

8. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет составлять и выводить в результирующий файл слово из тех букв английского алфавита, которые встречаются во входных данных либо как строчные, либо как прописные, причем буквы должны идти в алфавитном порядке. Каждая буква должна быть распечатана один раз. Буквы построенного слова должны быть прописными. Если во входных данных встречаются все буквы английского алфавита, то следует вывести строчными буквами слово "no".
Например, пусть в одной из строк исходного файла содержатся следующие символы:
absCDKLMNOPvwXYabcprst.

В этом случае в результирующем файле должно быть:
ABCDKLMNOPRSTVWXY

9. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет составлять и выводить в результирующий файл из тех цифр, которые встречаются во входных данных, максимальное число. При составлении итогового числа каждая цифра может быть использована только один раз. Если во входных данных цифры не встречаются, то следует вывести "-1".
Например, пусть в одной из строк исходного файла содержатся следующие символы:
14ф73п439
аеро
В этом случае в результирующем файле должно быть:
97431
-1

10. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет определять и выводить в результирующий файл английскую букву, встречающуюся в этой строке чаще всего, и количество там таких букв. Строчные и прописные буквы при этом считаются не различимыми. Если искомых букв несколько, то программа должна выводить на экран первую из них по алфавиту. Например, пусть в одной из строк исходного файла содержатся следующие символы:
It is not a simple task. Yes!
В этом случае в результирующем файле должно быть:
I 3

11. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет печатать в результирующий файл в алфавитном порядке только те буквы, которые встретились во входной последовательности ровно 3 раза. Каждая буква при этом должна быть распечатана один раз. Буквы построенного слова должны быть прописными.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
btfgbbffjrtatbama
в результирующем файле должно быть:
aft

12. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая будет проводить частотный анализ текста и последовательно выводить в результирующий файл только букву и через пробел символ * в количестве, равном количеству повторений этой буквы в тексте (в каждой строке результирующего файла информация об одной букве, другие символы не учитываются). Сведения о буквах, которые в тексте отсутствуют, на экран выводиться не должны. Сами буквы должны выводиться в алфавитном порядке.
Например, для текста:
It is science
в результирующем файле должно быть:
C **
E **
I ***
N *
S**
T *

13. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет выводить в результирующий файл последовательность цифр (`'0','1'..'9'`) из входной последовательности в порядке увеличения частоты их встречаемости. Каждая цифра при этом должна быть распечатана один раз.
Если какие-то цифры встречаются одинаковое число раз, то они выводятся по возрастанию.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
123**24#32
в результирующем файле должно быть:
1432

14. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет определять и выводить в результирующий файл буквы, встречающиеся в этой строке в порядке уменьшения частоты их встречаемости. Строчные и прописные буквы при этом считаются не различимыми. Каждая буква, которая встречается в тексте, должна быть выведена ровно один раз.
Если какие-то буквы встречаются одинаковое количество раз, то они выводятся в алфавитном порядке.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
zzzbbaattt
в результирующем файле должно быть:
tzab

15. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Необходимо определить количество букв в самом длинном слове файла, обозначив полученное число ***K*** (словом называется непрерывная последовательность английских букв, слова друг от друга отделяются любыми другими символами, длина слова не превышает 20 символов). Затем необходимо переписать строки исходного файла в результирующий файл, заменив каждую английскую букву в строке на букву, стоящую в алфавите на ***K*** букв раньше (алфавит считается циклическим, то есть перед буквой ***A*** стоит буква ***Z***), оставив другие символы неизменными. Строчные буквы при этом остаются строчными, а прописные - прописными.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
Ce Ud Fd,Gde Ud
в результирующем файле должно быть:
Zb Ra Ca,Dab Ra

16. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Необходимо определить количество букв в самом коротком слове файла, обозначив полученное число ***K*** (словом называется непрерывная последовательность английских букв, слова друг от друга отделяются любыми другими символами, длина слова не превышает 20 символов). Затем необходимо переписать строки исходного файла в результирующий файл, заменив каждую английскую букву в строке на букву, стоящую в алфавите на ***K*** букв позже (алфавит считается циклическим, то есть перед буквой ***A*** стоит буква ***Z***), оставив другие символы неизменными. Строчные буквы при этом остаются строчными, а прописные - прописными.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
Zb Ra Ca Dab Ra
в результирующем файле должно быть:
Bd Tc Ec Fcd Tc

17. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного**

файла будет определять и выводить в результирующий файл буквы, встречающиеся в этой строке в порядке уменьшения частоты их встречаемости. Строчные и прописные буквы при этом считаются не различимыми. Каждая буква, которая встречается в тексте, должна быть выведена ровно один раз. Если какие-то буквы встречаются одинаковое количество раз, то они выводятся в алфавитном порядке.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
baobaba.
в результирующем файле должно быть:
oab

18. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать, которая **для каждой строки исходного файла** будет печатать в результирующий файл в алфавитном порядке только те буквы, которые встретились во входной последовательности ровно 3 раза подряд (друг за другом). Каждая буква при этом должна быть распечатана один раз. Буквы построенного слова должны быть прописными.
Например, пусть в одной из строк исходного файла содержатся следующие символы:
bbbtffbgbbfffrtatbafffma.
в результирующем файле должно быть:
bf

19. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет печатать в результирующий файл слова, являющиеся палиндромами (читающимися в прямом и обратном порядке). Слово, состоящее из одной буквы, также считается.

20. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Требуется написать программу, которая **для каждой строки исходного файла** будет печатать в результирующий файл слова, начинающиеся и заканчивающиеся одной и той же буквой. Слово, состоящее из одной буквы, также считается.

21. В каждой строке текстового файла хранится информация следующего вида: слово и число его повторений. Требуется написать программу, которая **для каждой строки исходного файла** будет печатать в результирующий файл слова, повторив их столько раз, сколько указано

22. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Подсчитать **в каждой строке** количество слов. В новый файл в каждую строку записать это количество и, через пробел, самое короткое и самое длинное слово из каждой строки исходного файла, заключенные в скобки.

23. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. Провести частотный анализ текста (во всем файле): посчитать количество слов, начинающихся на различные буквы латинского алфавита. В новый файл в каждую строку записать букву и количество слов, начинающихся на эти буквы.

24. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. В новый файл переписать все строки исходного файла, заменив каждое слово на его палиндром.

25. В исходном текстовом файле записаны строки, содержащие произвольные алфавитно-цифровые символы. В новый файл переписать все строки исходного файла, форматируя их «по ширине», т.е. добавляя пробелы между словами таким образом, чтобы длина строки достигла величины N, заданной пользователем.

26. В исходном текстовом файле хранится информация о реках в виде: название, протяженность в километрах. Создать новый файл, в который поместить информацию о реках в порядке уменьшения их протяженности.

27. Из двух заданных файлов, содержащих строки произвольной длины, сформировать новый файл, содержащий строки по N символов. Каждая строка результирующего файла содержит N1 символов из 1-го файла, N2 пробелов и остальные символы (N-N1-N2) из 2-го файл.

28. В текстовом файле хранится информация о реках в виде: название, протяженность в километрах. Найти максимальную протяженность реки и дописать это число в конец файла. В другой текстовый файл переписать информацию о реках, в названии которых встречается фрагмент, указанный пользователем.