Лекция 5

часть 2 "Работа с файлами"

Финансовый университет при Правительстве РФ, лектор С.В. Макрушин

v 0.6 20.10.2020

Разделы:

-

-

In [1]:

```
# загружаем стиль для оформления презентации
from IPython.display import HTML
from urllib.request import urlopen
html = urlopen("file:./lec_v1.css")
HTML(html.read().decode('utf-8'))
```

Out[1]:

Файл

-

Файл — именованная область данных на носителе информации. Работа с файлами реализуется средствами операционных систем. Операционная система предоставляет приложениям набор функций и структур (API) для работы с файлами. На уровне API уже не существенно, существует ли файл как объект файловой системы или является, например, устройством ввода-вывода.

Можно выделить два типа операций с файлом — связанные с его открытием и выполняющиеся без его

открытия.

- Операции первого типа обычно служат для **чтения и записи информации** или подготовки к чтению или записи.
- Операции второго типа выполняются с файлом как **с объектом файловой системы**, в котором файл является наименьшим элементом структурирования.

В большинстве современных операционных систем файлы организованы в виде одномерных массивов байтов.

Обычно выделяют следующие сущности, связанные с работой с файлом:

- **Дескриптор файла** (хэндлер, описатель). При открытии файла, операционная система возвращает число (или указатель на структуру), с помощью которого выполняются все остальные файловые операции. По их завершении файл закрывается, а дескриптор теряет смысл.
- **Файловый указатель**. Число, являющееся смещением относительно нулевого байта в файле. Обычно по этому адресу осуществляется чтение/запись, в случае, если вызов операции чтения или записи не предусматривает указание адреса. При выполнении операций чтения или записи файловый указатель увеличивается на число прочитанных или записанных байт. Последовательный вызов операций чтения таким образом позволяет прочитать весь файл последовательно, не заботясь о позиционировании.
- **Файловый буфер**. Операционная система (или библиотека языка программирования) осуществляет кэширование файловых операций в специальном буфере (участке памяти). При закрытии файла буфер сбрасывается.
- **Режим доступа**. В зависимости от потребностей программы, файл может быть открыт на чтение или запись. Кроме того, некоторые операционные системы и библиотеки предусматривают режим работы с текстовыми файлами. Режим обычно указывается при открытии файла.
- **Режим общего доступа**. Предназначен для регулирования совместного доступа к одному файлу несколькими программами в многозадачной операционной системе.

Базовые операции для работы с файлом:

- **Открытие файла** (обычно в качестве параметров передается имя файла, режим доступа и режим совместного доступа, а в качестве результата выступает файловый дескриптор), кроме того обычно имеется возможность в случае открытия на запись указать на то, должен ли размер файла изменяться на нулевой.
- **Закрытие файла**. В качестве аргумента выступает значение, полученное при открытии файла. При закрытии все файловые буферы сбрасываются.
- **Запись** — в файл помещаются данные.
- **Чтение** — данные из файла помещаются в область памяти.
- **Перемещение указателя** — указатель перемещается на указанное число байт вперёд или назад или перемещается по указанному смещению относительно начала или конца. Не все файлы позволяют выполнение этой операции (например, файл на ленточном накопителе может не «уметь» перематываться назад).
- Получение **текущего значения файлового указателя**.
- **Сброс буферов** — содержимое файловых буферов с не записанной в файл информацией записывается. Используется обычно для указания на завершение записи логического блока (для сохранения данных в файле на случай сбоя).

Открытие файла

-

- к оглавлению

Прежде чем работать с файлом, необходимо создать объект файла с помощью функции open(). Функция имеет следующий формат:

open(<Путь к файлу>[, mode='r'] [, buffering=-1] [, encoding=None] [,errors=None] [, newline=None] [, closefd=True])

В первом параметре указывается путь к файлу. Путь может быть абсолютным или относительным. В Windows следует учитывать, что слэш является специальным символом. По этой причине слэш необходимо удваивать или вместо обычных строк использовать неформатированные строки.

In [17]:

```
# Правильно
fn = "C:\\temp\\new\\file.txt"
fn
```

Out[17]:

```
'C:\\temp\\new\\file.txt'
```

In [15]:

```
# Правильно
fn = r"C:\temp\new\file.txt"
fn
```

Out[15]:

```
'C:\\temp\\new\\file.txt'
```

In [12]:

```
# Неправильно!
# В этом пути присутствуют сразу три специальных символа: \t, \n и \f. После преобразования
# будет выглядеть следующим образом: C:<Табуляция>emp<Перевод строки>ew<Перевод формата>ile
fn = "C:\temp\new\file.txt"
fn
```

Out[12]:

```
'C:\temp\new\x0cile.txt'
```

Если путь к файлу начинаетсяя со слеша или с имени диска, то файл находится по абсолютному пути и текущий путь не имеет значения. В обратном случае для поиска файла используется относительный путь, построение которого начинается с текущего пути. Текущий путь определяется окружением в котором производился зпуск скрипта. В Jupyter notebook текущий путь можно узнать при помощи команды `ls` .

In [20]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ЃЃв ¬ЃвЄг Data
 ‘ЃаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®¤Ґа¦Ё¬®Ґ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions

16.08.2021  12:29    <DIR>          .
16.08.2021  12:29    <DIR>          ..
16.08.2021  09:25    <DIR>          .ipynb_checkpoints
16.08.2021  09:21    <DIR>          folder1
08.09.2020  21:09             2я745 lec_v1.css
16.08.2021  09:25            40я980 lec5p1_v2020s2_v5.ipynb
20.10.2020  19:57           169я355 lec5p1_v2020s2_v5.pdf
16.08.2021  12:27           309я984 lec5p2_ex6_v2020s2_v6.ipynb
20.10.2020  20:04           214я766 lec5p2_ex6_v2020s2_v6.pdf
16.08.2021  09:26           224я852 lec5p2_ex6_v2020s2_v6_.pdf
06.04.2020  23:09           255я678 lec6_v2018s2_v4.pdf
06.04.2020  20:48                16 my.txt
20.10.2020  13:19                20 my2.txt
06.04.2020  20:47                 0 new_file3.txt
16.08.2021  11:52                16 new_file4.txt
06.04.2020  20:47                 8 new_file5.txt
06.04.2020  20:48                28 new_file7.txt
06.04.2020  20:51                94 new_file8.txt
06.04.2020  23:07           220я029 participants.csv
06.04.2020  23:07           220я029 participants_.csv
              16 д ©«®ў      1я658я600 Ў ©в
               4 Ї Ї®Є  172я206я403я584 Ў ©в бў®Ў®¤®
```

Если открываемый файл находится в текущем рабочем каталоге, то можно указать только название файла. Пример:

In [21]:

```
f1 = open('new_file.txt', mode='w')
f1.close()
```

In [23]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ЃЃв ¬ЃвЄг Data
 ‘ЃаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®¤Ґа¦Ё¬®Ґ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions

16.08.2021  12:29    <DIR>          .
16.08.2021  12:29    <DIR>          ..
16.08.2021  09:25    <DIR>          .ipynb_checkpoints
16.08.2021  12:30    <DIR>          folder1
08.09.2020  21:09             2я745 lec_v1.css
16.08.2021  09:25            40я980 lec5p1_v2020s2_v5.ipynb
20.10.2020  19:57           169я355 lec5p1_v2020s2_v5.pdf
16.08.2021  12:29           310я300 lec5p2_ex6_v2020s2_v6.ipynb
20.10.2020  20:04           214я766 lec5p2_ex6_v2020s2_v6.pdf
16.08.2021  09:26           224я852 lec5p2_ex6_v2020s2_v6_.pdf
06.04.2020  23:09           255я678 lec6_v2018s2_v4.pdf
06.04.2020  20:48                16 my.txt
20.10.2020  13:19                20 my2.txt
16.08.2021  12:29                 0 new_file.txt
06.04.2020  20:47                 0 new_file3.txt
16.08.2021  11:52                16 new_file4.txt
06.04.2020  20:47                 8 new_file5.txt
06.04.2020  20:48                28 new_file7.txt
06.04.2020  20:51                94 new_file8.txt
06.04.2020  23:07           220я029 participants.csv
06.04.2020  23:07           220я029 participants_.csv
              17 д ©«®ў      1я658я916 Ў ©в
               4 Ї Ї®Є  172я195я442я688 Ў ©в бў®Ў®¤®
```

Если открываемый файл расположен во вложенной папке, то перед названием файла перечисляются названия вложенных папок через слэш:

In [24]:

```
f2 = open('folder1\\new_file2.txt', mode='w')
f2.close()
```

In [25]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ҐҐв ¬ҐвЄг Data
 ‘ҐаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®¤Ґа¦Ё¬®Ґ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions

16.08.2021  12:29    <DIR>          .
16.08.2021  12:29    <DIR>          ..
16.08.2021  09:25    <DIR>          .ipynb_checkpoints
16.08.2021  12:30    <DIR>          folder1
08.09.2020  21:09             2я745 lec_v1.css
16.08.2021  09:25            40я980 lec5p1_v2020s2_v5.ipynb
20.10.2020  19:57           169я355 lec5p1_v2020s2_v5.pdf
16.08.2021  12:29           310я300 lec5p2_ex6_v2020s2_v6.ipynb
20.10.2020  20:04           214я766 lec5p2_ex6_v2020s2_v6.pdf
16.08.2021  09:26           224я852 lec5p2_ex6_v2020s2_v6_.pdf
06.04.2020  23:09           255я678 lec6_v2018s2_v4.pdf
06.04.2020  20:48                16 my.txt
20.10.2020  13:19                20 my2.txt
16.08.2021  12:29                 0 new_file.txt
06.04.2020  20:47                 0 new_file3.txt
16.08.2021  11:52                16 new_file4.txt
06.04.2020  20:47                 8 new_file5.txt
06.04.2020  20:48                28 new_file7.txt
06.04.2020  20:51                94 new_file8.txt
06.04.2020  23:07           220я029 participants.csv
06.04.2020  23:07           220я029 participants_.csv
              17 д ©«®ў      1я658я916 Ў ©в
               4 Ї Ї®Є  172я195я442я688 Ў ©в бў®Ў®¤®
```

Перемещение между папками в Jupyter notebook:

In [26]:

```
cd folder1
```

```
E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions\folder1
```

In [27]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ҐҐв ¬ҐвЄг Data
 ‘ҐаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®¤Ґа¦Ё¬®Ґ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions
\folder1

16.08.2021  12:30    <DIR>          .
16.08.2021  12:30    <DIR>          ..
16.08.2021  12:30                 0 new_file2.txt
               1 д ©«®ў                0 Ў ©в
               2 Ї Ї®Є  172я193я271я808 Ў ©в бў®Ў®¤®
```

При указании пути можно ссылаться на текущую папку с помощью символа . и на родительскую папку с

помощью символа `..` .

Если папка с файлом расположена выше уровнем, то перед названием файла указываются две точки и слэш `..\\` :

In [28]:

```
f3 = open('..\\new_file3.txt', mode='w')
f3.close()
```

In [29]:

```
cd ..
```

```
E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptions
```

In [30]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ЃЃв ¬ЃвЄг Data
 ‘ЃаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®¤Ґа¦Ё¬®Ґ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptio
ns

16.08.2021  12:32    <DIR>          .
16.08.2021  12:32    <DIR>          ..
16.08.2021  09:25    <DIR>          .ipynb_checkpoints
16.08.2021  12:30    <DIR>          folder1
08.09.2020  21:09             2я745 lec_v1.css
16.08.2021  09:25            40я980 lec5p1_v2020s2_v5.ipynb
20.10.2020  19:57           169я355 lec5p1_v2020s2_v5.pdf
16.08.2021  12:31           310я711 lec5p2_ex6_v2020s2_v6.ipynb
20.10.2020  20:04           214я766 lec5p2_ex6_v2020s2_v6.pdf
16.08.2021  09:26           224я852 lec5p2_ex6_v2020s2_v6_.pdf
06.04.2020  23:09           255я678 lec6_v2018s2_v4.pdf
06.04.2020  20:48                16 my.txt
20.10.2020  13:19                20 my2.txt
```

Функция `listdir(path)` возвращает все объекты (и файлы и папки), располагающиеся по указанному пути.

Для проверки, является ли найденный объект файлом можно использовать функцию `isfile` .

In [34]:

```
from os import listdir
from os.path import isfile, join

# получение имен всех файлов, находящихся по определенному пути:
mypath = '.'
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]

onlyfiles
```

Out[34]:

```
['lec5p1_v2020s2_v5.ipynb',
 'lec5p1_v2020s2_v5.pdf',
 'lec5p2_ex6_v2020s2_v6.ipynb',
 'lec5p2_ex6_v2020s2_v6.pdf',
 'lec5p2_ex6_v2020s2_v6_.pdf',
 'lec6_v2018s2_v4.pdf',
 'lec_v1.css',
 'my.txt',
 'my2.txt',
 'new_file.txt',
 'new_file3.txt',
 'new_file4.txt',
 'new_file5.txt',
 'new_file7.txt',
 'new_file8.txt',
 'participants.csv',
 'participants_.csv']
```

Дополнительные параметры функции `open`

Необязательный параметр mode в функции open() может принимать следующие значения:

- r - только чтение (значение по умолчанию). После открытия файла указатель устанавливается на начало файла. Если файл не существует, то возбуждается исключение IOError;
- r+ - чтение и запись. После открытия файла указатель устанавливается на начало файла. Если файл не существует, то возбуждается исключение IOError;
- w - запись. Если файл не существует, то он будет создан. Если файл существует, то он будет перезаписан. После открытия файла указатель устанавливается на начало файла;
- w+ - чтение и запись. Если файл не существует, то он будет создан. Если файл существует, то он будет перезаписан. После открытия файла указатель устанавливается на начало файла;
- a - запись. Если файл не существует, то он будет создан. Запись осуществляется в конец файла. Содержимое файла не удаляется;
- a+ - чтение и запись. Если файл не существует, то он будет создан. Запись осуществляется в конец файла. Содержимое файла не удаляется.

Кроме того, после режима может следовать модификатор:

- b - файл будет открыт в бинарном режиме. Файловые методы принимают и возвращают объекты типа bytes;
- t - файл будет открыт в текстовом режиме (значение по умолчанию в Windows). Файловые методы принимают и возвращают объекты типа `str` .

Разные платформы (ОС) используют разные коды в качестве символа новой строки, в частности:

- Windows: `\r\n`
- Unix: `\n`
- Старые компьютеры Макинтош: `\r`
- Существуют системы использующие: `\n\r`

При открытии файла в текстовом режиме в Python 3 все варианты окончания строки будут автоматически конвертироваться к `\n` (как при записи, так и при чтении). Например, в этом режиме в Windows при чтении символ `\r` будет удален, а при записи, наоборот, добавлен.

In [35]:

```
f4 = open('new_file4.txt', mode='w')
```

In [36]:

```
# возвращается количество записанных символов (значение всегда совпадает с длиной строки):
f4.write( "String1\nString2")
```

Out[36]:

```
15
```

In [37]:

```
f4.close()
```

In [38]:

```
ls
```

```
 ’®¬ ў гбва®©бвўЃ Е Ё¬ЃЃв ¬ЃвЄг Data
 ‘ЃаЁ©л© ®¬Ґа в®¬ : EE2C-D1DD

 ‘®ЯЃа¦Ё¬®Ѓ Ї ЇЄЁ E:\YandexDisk\Python\Ipynb\fa1_2021\lec05_files_exceptio
ns

16.08.2021  12:40    <DIR>          .
16.08.2021  12:40    <DIR>          ..
16.08.2021  09:25    <DIR>          .ipynb_checkpoints
16.08.2021  12:30    <DIR>          folder1
08.09.2020  21:09             2я745 lec_v1.css
16.08.2021  09:25            40я980 lec5p1_v2020s2_v5.ipynb
20.10.2020  19:57           169я355 lec5p1_v2020s2_v5.pdf
16.08.2021  12:37           311я106 lec5p2_ex6_v2020s2_v6.ipynb
20.10.2020  20:04           214я766 lec5p2_ex6_v2020s2_v6.pdf
16.08.2021  09:26           224я852 lec5p2_ex6_v2020s2_v6_.pdf
06.04.2020  23:09           255я678 lec6_v2018s2_v4.pdf
06.04.2020  20:48                16 my.txt
20.10.2020  13:19                20 my2.txt
```

Инструкция with ... as

-

- к оглавлению

Python поддерживает **протокол менеджеров контекста**. Этот протокол гарантирует выполнение завершающих действий (например, закрытие файла) вне зависимости от того, произошло исключение внутри блока кода или нет.

Для работы с протоколом предназначена инструкция with ... as. Инструкция имеет следующий формат:

```
with <Выражение1>[ as <Переменная>] [, ... ,
    <ВыражениеN>[ as <Переменная>]]:
    <Блок, в котором перехватываем исключения>
```

Вначале вычисляется <Выражение1>, которое должно возвращать объект, поддерживающий протокол. Этот объект должен иметь два метода: `__enter__()` и `__exit__()`. Метод `__enter__()` вызывается после создания объекта. Значение, возвращаемое этим методом, присваивается переменной, указанной после ключевого слова as. Далее выполняются инструкции внутри тела инструкции with. Если при выполнении возникло исключение, то управление передается методу `__exit__()`. Если при выполнении инструкций, расположенных внутри тела инструкции with, исключение не возникло, то управление все равно передается методу `__exit__()`. В функции `__exit__()` решаются все задачи освобождения ресурсов, если это необходимо.

Некоторые встроенные объекты по умолчанию поддерживают протокол, например, файлы.

In [39]:

```
with open("new_file4.txt", "rb") as f:
    for line in f:
        print(type(line))
        print(repr(line))
```

```
<class 'bytes'>
b'String1\r\n'
<class 'bytes'>
b'String2'
```

Для ускорения работы производится буферизация записываемых данных. Информация из буфера записывается в файл полностью только в момент закрытия файла. В необязательном параметре buffering можно указать размер буфера. Если в качестве значения указан 0, то данные будут сразу записываться в файл (значение допустимо только в бинарном режиме).

Значение 1 используется при построчной записи в файл (значение допустимо только в текстовом режиме), другое положительное число задает примерный размер буфера, а отрицательное значение (или отсутствие значения) означает установку размера, применяемого в системе по умолчанию.

При использовании текстового режима (используется по умолчанию) при чтении производится попытка преобразовать данные в кодировку Unicode, а при записи выполняется обратная операция: строка преобразуется в последовательность байтов в определенной кодировке. По умолчанию используется· кодировка, используемая в системе. Если преобразование невозможно, то возбуждается исключение. Указать кодировку, которая будет использоваться при записи и чтении файла, позволяет параметр encoding. В качестве примера запишем данные в кодировке UTF-8:

In [20]:

```
with open("new_file5.txt", "w", encoding="utf-8") as f: #указываем кодировку файла
    f.write("Строка") # Записываем строку в файл
```

In [21]:

```
with open("new_file5.txt", "r", encoding="utf-8") as f:
    for line in f:
        print(line)
```

```
Строка
```

Методы для работы с файлами

-

- к оглавлению

`close()` - закрывает файл. Так как интерпретатор автоматически удаляет объект, когда отсутствуют ссылки на него, можно явно не закрывать файл в небольших программах. Тем не менее явное закрытие файла является признаком хорошего стиля программирования. Протокол менеджеров контекста гарантирует закрытие файла вне зависимости от того, произошло исключение внутри блока кода или нет.

`write (<Данные>)` - записывает строку или последовательность байтов в файл. Если в качестве параметра указана строка, то файл должен быть открыт в текстовом режиме. Для записи последовательности байтов необходимо открыть файл в бинарном режиме.

Помните, что нельзя записывать строку в бинарном режиме и последовательность байтов в текстовом режиме. Метод возвращает количество записанных символов или байтов.

In [41]:

```
# бинарный режим:
with open("new_file6.txt", "wb") as f:
    f.write(bytes("Строка1\nСтрока2", "cp1251"))
    f.write(bytearray("\nСтрока3", "cp1251"))
```

In [42]:

```
# обязательные параметры конструктора bytes: строка и кодировка строки
bytes("Строка1\nСтрока2", "cp1251")
```

Out[42]:

```
b'\xd1\xf2\xf0\xee\xea\xe01\n\xd1\xf2\xf0\xee\xea\xe02'
```

In [43]:

```
bs = bytes(range(20))
bs
```

Out[43]:

```
b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13'
```

In [44]:

```
for b in bs:
    print(b)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [45]:

```
bytes(range(255, 265))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-45-9298608765b3> in <module>
----> 1 bytes(range(255, 265))

ValueError: bytes must be in range(0, 256)
```

In [46]:

```
bytes("Строка1\nСтрока2", "utf-8")
```

Out[46]:

```
b'\xd0\xa1\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb01\n\xd0\xa1\xd1\x82\xd1\x80
\xd0\xbe\xd0\xba\xd0\xb02'
```

`writelines(<Последовательность>)` - записывает последовательность в файл. Если все элементы последовательности являются строками, то файл должен быть открыт в текстовом режиме. Если все элементы являются последовательностями байтов, то файл должен быть открыт в бинарном режиме.

Запись в файл можно призводить при помощи функции print(), передав файл в необязательный атрибут file:

In [47]:

```
with open("new_file8.txt", "w", encoding="cp1251") as f:
    print('Строка1\nСтрока2', file=f)
    print('Строка3', end='', file=f) # значение по умолчанию end='\n'
    print(' продолжение строки3', file=f)
    v = 4
    print(f'Простой способ записать в файл f-строку v={v} .', file=f)
```

In [48]:

```
with open("new_file8.txt", "r", encoding="cp1251") as f:
    print(f.read())
```

```
Строка1
Строка2
Строка3 продолжение строки3
Простой способ записать в файл f-строку v=4 .
```

In [49]:

```
with open("new_file7.txt", "w", encoding="utf-8") as f:
    f.writelines(["Строка1\n", "Строка2"])
```

`read( [<Количество>])` - считывает данные из файла. Если файл открыт в текстовом режиме, то возвращается строка, а если в бинарном - последовательность байтов. Если параметр не указан, то возвращается содержимое файла от текущей позиции указателя до конца файла:

In [50]:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    print(f.read())
```

```
Строка1
Строка2
```

In [51]:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    print(f.read(4))
    print(f.read(4))
```

```
Стро
ка1
```

In [52]:

```
with open("new_file7.txt", "rb") as f:
    print(f.read())
```

```
b'\xd0\xa1\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb01\r\n\xd0\xa1\xd1\x82\xd1\x
80\xd0\xbe\xd0\xba\xd0\xb02'
```

In [53]:

```
with open("new_file7.txt", "rb") as f:
    print(f.read(4))
```

```
b'\xd0\xa1\xd1\x82'
```

readline ([<Количество>]) - считывает из файла одну строку при каждом вызове. Если файл открыт в текстовом режиме, то возвращается строка, а если в бинарном - последовательность байтов. Возвращаемая строка включает символ перевода строки. Исключением является последняя строка. Если она не завершается символом перевода строки, то символ перевода строки добавлен не будет. При достижении конца файла возвращается пустая строка.

Если в необязательном параметре указано число, то считывание будет выполняться до тех пор, пока не встретится символ новой строки (\n), символ конца файла или из файла не будет прочитано указанное количество символов. Иными словами, если количество символов в строке меньше значения параметра, то будет считана одна строка, а не указанное количество символов. Если количество символов в строке больше, то возвращается указанное количество символов.

In [54]:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    print(f.readline())
    print('----')
    print(f.readline())
    print('----')
    print(f.readline())
```

```
Строка1

----
Строка2
----
```

In [55]:

```
with open("new_file7.txt", "rb") as f:
    print(f.readline())
    print('----')
    print(f.readline())
    print('----')
    print(f.readline())
```

```
b'\xd0\xa1\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb01\r\n'
----
b'\xd0\xa1\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb02'
----
b''
```

`readlines ()` - считывает все содержимое файла в список. Каждый элемент списка будет содержать одну строку, включая символ перевода строки. Исключением является последняя строка. Если она не завершается символом перевода строки, то символ перевода строки добавлен не будет. Если файл открыт в текстовом режиме, то возвращается список строк, а если в бинарном - список объектов типа bytes.

In [56]:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()
lines
```

Out[56]:

```
['Строка1\n', 'Строка2']
```

`__next__()` - считывает одну строку при каждом вызове. Если файл открыт в текстовом режиме, то возвращается строка, а если в бинарном - последовательность байтов. При достижении конца файла возбуждается исключение Stopiteration. Пример:

In [57]:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    for l in f:
        print(l)
```

```
Строка1

Строка2
```

`flush()` - записывает данные из буфера на диск;

`tell()` - возвращает позицию указателя относительно начала файла в виде целого числа. Обратите внимание на то, что в Windows метод `tell()` считает символ `\r` как дополнительный байт, хотя этот символ удаляется при открытии файла в текстовом режиме, чтобы избежать этого несоответствия, следует открывать файл в бинарном режиме, а не в текстовом.

`seek (<Смещение> [, <Позиция>])` - устанавливает указатель в позицию, имеющую смещение <Смещение> относительно позиции <Позиция>. В параметре <Позиция> могут быть указаны следующие атрибуты из модуля io или соответствующие им значения:

- io.SEEK_SET или 0 - начало файла (значение по умолчанию);
- io.SEEK_CUR или 1 - текущая позиция указателя;
- io.SEEK_END или 2 - конец файла.

Помимо методов объекты файлов поддерживают несколько атрибутов:

- name - содержит название файла;
- mode - режим, в котором был открыт файл;
- closed - возвращает True, если файл был закрыт, и False в противном случае.
- encoding - название кодировки, которая будет использоваться для преобразования строк перед записью в файл или при чтении. Обратите внимание на то, что изменить значение атрибута нельзя, т. к. атрибут доступен только мя чтения. Атрибут доступен только в текстовом режиме.
- buffer - позволяет получить доступ к буферу. Атрибут доступен только в текстовом режиме. С помощью этого объекта можно записать последовательность байтов в текстовый поток.

Сохранение объектов в файл

-

- к оглавлению

Сохранить объекты в файл и в дальнейшем восстановить объекты из файла позволяют модули pickle и shelve.

Модуль pickle предоставляет следующие функции:

- `dump(<Объект>, <Файл> [, <Протокол>] [, fix _ imports=True])` - производит сериализацию объекта и записывает данные в указанный файл. В параметре <Файл> указывается файловый объект, открытый на запись в бинарном режиме.
- `load()` - читает данные из файла и преобразует их в объект. В параметре <Файл> указывается файловый объект. открытый на чтение в бинарном режиме. Формат функции: `load (<Файл> [, fix _ imports=True] [, encoding="ASCII"] [, errors="strict"])`

In [58]:

```
obj = ["Строка", (12, 3)] # объект для сохранения
```

In [59]:

```
import pickle # подключаем модуль pickle
```

In [60]:

```
with open('obj1.txt', 'wb') as f:
    pickle.dump(obj, f)
```

In [61]:

```
with open('obj1.txt', 'rb') as f:
    obj_l = pickle.load(f)
obj_l
```

Out[61]:

```
['Строка', (12, 3)]
```

В один файл можно сохранить сразу несколько объектов, последовательно вызывая функцию dump().

In [62]:

```
obj2 = (6, 7, 8, 9, 10)
```

In [63]:

```
with open('obj2.txt', 'wb') as f:
    pickle.dump(obj, f)
    pickle.dump(obj2, f)
```

In [64]:

```
with open('obj2.txt', 'rb') as f:
    obj_l2 = pickle.load(f)
    obj2_l = pickle.load(f)
obj_l2, obj2_l
```

Out[64]:

```
(['Строка', (12, 3)], (6, 7, 8, 9, 10))
```

Модуль pickle позволяет также преобразовать объект в последовательность байтов и восстановить объект из этой последовательности. Для этого предназначены две функции:

- `dumps(<Объект> [, <Протокол>] [, fix_imports=True])` - производит сериализацию объекта и возвращает последовательность байтов специального формата.
- `loads(<Последовательность байтов>[, fix_imports=True] [, errors="strict"])` - преобразует последовательность байтов обратно в объект.

In [65]:

```
bs = pickle.dumps(obj)
bs
```

Out[65]:

```
b'\x80\x03]q\x00(X\x0c\x00\x00\x00\xd0\xa1\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd
0\xb0q\x01K\x0cK\x03\x86q\x02e.'
```

In [66]:

```
pickle.loads(bs)
```

Out[66]:

```
['Строка', (12, 3)]
```

Модуль **shelve** позволяет сохранять объекты под определенным ключом (задается в виде строки) и предоставляет интерфейс доступа, сходный со словарями. Для сериализации объекта используются возможности модуля pickle. Эти действия модуль shelve производит незаметно для нас.

Чтобы открыть файл с базой объектов, используется функция `open()` . Функция имеет следующий формат:

```
open(<Путь к файлу> [, flag="c"] [, protocol=None] [, writeback=False])
```

В необязательном параметре flag можно указать один из режимов открытия файла:

- r - только чтение;
- w - чтение и запись;
- c - чтение и запись (значение по умолчанию). Если файл не существует, он будет создан;
- n - чтение и запись. Если файл не существует, он будет создан. Если файл существует, он будет перезаписан.

Функция open() возвращает объект, с помощью которого производится дальнейшая работа с базой данных. Этот объект имеет следующие методы:

- close() - закрывает файл с базой данных.
- keys() - возвращает объект с ключами;
- values() - возвращает объект to значениями;
- items() - возвращает объект, поддерживающий итерации. На каждой итерации возвращается кортеж, содержащий ключ и значение.
- get(<Ключ> [, <Значение по умолчанию>]) - если ключ присутствует; то метод возвращает значение, соответствующее этому ключу. Если ключ отсутствует, то возвращается начение None или значение, указанное во втором параметре;
- setdefault(<Ключ> [, <Значение по умолчанию>]) ---: если ключ присутствует, то метод возвращает значение, соответствующее этому ключу. Если ключ отсутствует, то вставляет новый элемент со значением, указанным во втором параметре, и возвращает это значение. Если второй параметр не указан, значением нового элемента будет None;
- pop (<Ключ> [, <Значение по умолчанию>]) - удаляет элемент с указанным ключом и возвращает его значение. Если ключ отсутствует, то возвращается значение из второго параметра. Если ключ отсутствует, и второй параметр не указан, то возбуждается исключение KeyError;
- popitem() - удаляет произвольный элемент и возвращает кортеж из ключа и значения. Если файл пустой, возбуждается исключение KeyError;
- clear () - удаляет все элементы. Метод ничего не возвращает в качестве значения;
- update () - добавляет элементы. Метод изменяет текущий объект и ничего не возвращает. Если элемент с указанным ключом уже присутствует, то его значение будет перезаписано.

Помимо этих методов можно воспользоваться функцией len() для получения количества элементов и оператором del для удаления определенного элемента, а также оператором in для проверки существования ключа.

In [67]:

```
import shelve
```

In [68]:

```
db = shelve.open("shl_1")
```

In [69]:

```
db["obj1"] = [1, 2, 3, 4, 5]
db["obj2"] = (6, 7, 8, 9, 10)
```

In [70]:

```
db["obj1"]
```

Out[70]:

```
[1, 2, 3, 4, 5]
```

In [71]:

```
db["obj2"]
```

Out[71]:

```
(6, 7, 8, 9, 10)
```

In [72]:

```
db.close()
```

In [73]:

```
db = shelve.open('shl_1')
```

In [74]:

```
db.keys()
```

Out[74]:

```
KeysView(<shelve.DbfilenameShelf object at 0x0000018374F81D88>)
```

In [75]:

```
list(db.keys())
```

Out[75]:

```
['obj1', 'obj2']
```

In [76]:

```
list(db.values())
```

Out[76]:

```
[[1, 2, 3, 4, 5], (6, 7, 8, 9, 10)]
```

In [77]:

```
for k, v in db.items():
    print('key: {}, value: {}'.format(k, v))
```

```
key: obj1, value: [1, 2, 3, 4, 5]
key: obj2, value: (6, 7, 8, 9, 10)
```

Модуль CSV

-

- к оглавлению

https://docs.python.org/2/library/csv.html (https://docs.python.org/2/library/csv.html)

https://pymotw.com/2/csv/ (https://pymotw.com/2/csv/)

In [78]:

```
import csv
```

In [79]:

```
with open('participants.csv') as f:
    f_csv = csv.reader(f, delimiter=';')
    header = next(f_csv)
    rows = [r for r in f_csv]
```

In [80]:

```
header
```

Out[80]:

```
['Last Name',
 'First Name',
 'Company Name',
 'Company Department',
 'Assigned Classifications',
 'City',
 'State',
 'Country']
```

In [121]:

```
rows
```

Out[121]:

```
[['AALTONEN',
  'Wanida',
  'University',
  'Graduate student',
  'School or university (student)',
  'London',
  '',
  'United Kingdom'],
 ['ABDELHAMID',
  'Mohamed',
  'alexandria university',
  'student',
  'School or university (student)',
  'alexandria',
  '',
  'Egypt'],
 ['ABDELMEGUID',
  'Sheref',
```

Подготовка к следующей лекции:

Саммерфильд: гл. 5 Модули

Прохоренок: гл. 12 Модули и пакеты

Встроенные функции

Материалы по ссылкам:

https://docs.python.org/3/library/functions.html (https://docs.python.org/3/library/functions.html)

http://python-reference.readthedocs.io/en/latest/docs/functions/ (http://python-reference.readthedocs.io/en/latest/docs/functions/)

In [ ]: