Работа с файлами

№ урока: 9 **Курс:** Python Essential

Средства обучения: PyCharm

Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление о файлах и потоках, смогут записывать и считывать данные из файлов, иметь представление о работе менеджеров контекста.

Изучив материал данного занятия, учащийся сможет:

- Иметь представление о работе с файлами
- Записывать и считывать данные в текстовом и бинарном форматах
- Иметь представление об операторе with и менеджерах контекста

Содержание урока

1. Зачем работать с файлами?
2. Файлы и файловая система
3. Открытие файлов
4. Закрытие файла
5. Запись в файл
6. Чтение из файла
7. Методы объекта файла

Резюме

Файлы используются программами для **долговременного** хранения информации, как необходимой для собственной работы (например, настройки), так и полученной во время ее исполнения (результаты вычислений и т.д.). Большинство программ сегодня в том или ином виде используют файлы, сохраняя результаты работы между сеансами запуска.

Файлы и файловая система

Файл – именованная область данных на носителе информации.

Поскольку оперативная память (ОЗУ) является энергозависимой (которая теряет свои данные при выключении компьютера), мы используем файлы для будущего использования данных, постоянно сохраняя их.

Когда мы хотим читать или записывать в файл, нам нужно сначала открыть его. Когда мы закончим работу с файлом, его нужно закрыть, чтобы освободить ресурсы, связанные с файлом.

Следовательно, в Python файловая операция выполняется в следующем порядке:

1. Открыть файл
2. Чтение или запись (выполнение операции)
3. Закрыть файл

Открытие файлов

Python имеет встроенную функцию open() для открытия файла. Эта функция возвращает файловый объект, также называемый **дескриптором**, поскольку он используется для чтения или изменения файла соответственно.

```
>>> f = open("test.txt")     # open file in current directory
>>> f = open("C:/Python38/README.txt")  # specifying full path
```

Мы можем указать режим при открытии файла. В режиме мы указываем, хотим ли мы прочитать `r`, записать `w` или добавить в файл. Мы также можем указать, хотим ли мы открыть файл в текстовом или двоичном режиме.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

По умолчанию чтение в текстовом режиме. В этом режиме мы получаем строки при чтении из файла.
С другой стороны, двоичный режим возвращает байты, и это режим, который следует использовать при работе с не текстовыми файлами, такими как изображения или исполняемые файлы.

Режимы работы с файлами

Режим	Описание
r	Открывает файл для чтения. (по умолчанию)
w	"Открывает файл для записи. Создает новый файл, если он не существует, или обрезает файл, если он существует."
x	"Открывает файл для эксклюзивного создания. Если файл уже существует, операция не выполняется."
a	"Открывает файл для добавления в конец файла без его усечения. Создает новый файл, если он не существует."
t	Открывается в текстовом режиме. (по умолчанию)
b	Открывается в бинарном режиме.
+	Открывает файл для обновления (чтения и записи)

```
f = open("test.txt")      # эквивалентно 'r' или 'rt'
f = open("test.txt",'w')  # запись в текстовом режиме
f = open("img.bmp",'r+b') # чтение и запись в двоичном режиме
```

В отличие от других языков, символ **a** не подразумевает число 97, пока он не закодирован с использованием ASCII (или других эквивалентных кодировок).

```
f = open("test.txt", mode='r', encoding='utf-8')
```

Закрытие файла
Когда мы закончили выполнение операций с файлом, нам нужно правильно закрыть файл.
Закрытие файла освободит ресурсы, которые были связаны с файлом. Это делается с помощью метода close().
В Python есть сборщик мусора для очистки объектов, на которые нет ссылок, но мы не должны полагаться на него при закрытии файла.

```
f = open("test.txt", encoding = 'utf-8')
# какие-то операции с файлом...
f.close()
```

Этот способ **не** совсем **безопасен** (читать: опасен). Если при выполнении какой-либо операции с файлом возникает исключение, код завершается без закрытия файла.
Более безопасный способ - использовать блок try ... finally.

```
try:
   f = open("test.txt", encoding = 'utf-8')
   # какие-то операции с файлом...
finally:
   f.close()
```

Таким образом, мы гарантируем, что файл будет правильно закрыт, даже если возникнет исключение, которое приводит к остановке выполнения программы.
Лучший способ закрыть файл - использовать оператор with. Это гарантирует, что файл будет закрыт при выходе из блока внутри оператора with.
Нам не нужно явно вызывать метод close(). Это делается изнутри:

```
with open("test.txt", encoding = 'utf-8') as f:
   pass
   # какие-то операции с файлом...
```

Запись в файл
Чтобы записать данные в файл, нужно открыть его в режиме записи w, добавления a или эксклюзивного создания (**exclusive creation**) x.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Нам нужно быть осторожными с режимом w, так как файл будет полностью перезаписан, если он уже существует. Из-за этого стираются все предыдущие данные.
Запись строки или последовательности байтов (для двоичных файлов) выполняется с помощью метода write(). Этот метод возвращает количество символов, записанных в файл.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
   f.write("my first file\n")
   f.write("This file\n\n")
   f.write("contains three lines\n")
```

Эта программа создаст новый файл с именем test.txt в текущем каталоге, если он не существует. Если он существует, он перезаписывается.
Мы должны сами включать символы новой строки, чтобы различать разные строки или использовать метод f.writeline().

Чтение из файла
Чтобы прочитать файл, мы должны открыть файл в режиме чтения r.
Для этого доступны различные методы. Мы можем использовать метод read(size) для чтения количества данных размера. Если параметр размера не указан, выполняется чтение до конца файла.
Мы можем прочитать файл text.txt, который мы написали в предыдущем разделе, следующим образом:

```
>>> f = open("test.txt",'r',encoding = 'utf-8')
>>> f.read(4)
'This'

>>> f.read(4)
' is '

>>> f.read()
'my first file\nThis file\ncontains three lines\n'

>>> f.read()
```

Метод read() возвращает новую строку как '\n'. По достижении конца файла при дальнейшем чтении мы получаем пустую строку.
Мы можем изменить текущий курсор в файле (**позицию**) с помощью метода seek(). Точно так же метод tell() возвращает нашу текущую позицию (в байтах).

```
>>> f.tell()
56

>>> f.seek(0)
0

>>> print(f.read())
This is my first file
This file
contains three lines
```

Мы можем читать файл построчно, используя цикл for. Это одновременно и эффективно, и быстро.

```
>>> for line in f:
...     print(line, end = '')
...
This is my first file
This file
contains three lines
```

В этой программе строки уже в самом файле содержат символ новой строки \n. Итак, мы используем параметр end="" функции print(), чтобы избежать появления двух символов новой строки при печати.
В качестве альтернативы мы можем использовать метод readline() для чтения отдельных строк файла. Этот метод читает файл до новой строки, включая символ новой строки.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
>>> f.readline()
'This is my first file\n'

>>> f.readline()
'This file\n'

>>> f.readline()
'contains three lines\n'

>>> f.readline()
''
```

Наконец, метод readlines() возвращает список оставшихся строк всего файла. Все эти методы чтения возвращают пустые значения при достижении конца файла (**EOF**).

```
>>> f.readlines()
['This is my first file\n', 'This file\n', 'contains three lines\n']
```

Методы объекта файла

Файловым объект имеет различные методы. Некоторые из них были использованы в приведенных выше примерах.

Вот полный список методов в текстовом режиме с кратким описанием:

Методы файлового дескриптора

Метод	Описание
close()	Закрывает открытый файл. Не действует, если файл уже закрыт.
detach()	Отделяет базовый двоичный буфер от TextIOBase и возвращает его.
fileno()	Возвращает целое число (дескриптор файла) файла.
flush()	Очищает буфер записи файлового потока.
isatty()	Возвращает True, если файловый поток является интерактивным.
read(n)	Читает не более n символов из файла. Читает до конца файла, если он отрицательный или None.
readable()	Возвращает True, если файловый поток можно читать.
readline(n=-1)	Читает и возвращает одну строку из файла. Считывает не более n байтов, если указано.
readlines(n=-1)	Читает и возвращает список строк из файла. Считывает не более n байтов / символов, если указано.
seek(offset,from=SEEK_SET)	Изменяет позицию файла на смещение в байтах относительно from (начало, текущее, конец).
seekable()	Возвращает True, если файловый поток поддерживает произвольный доступ.
tell()	Возвращает текущее расположение файла.
truncate(size=None)	Изменяет размер файлового потока до байтов. Если размер не указан, изменяется до текущего местоположения.
writable()	Возвращает True, если файловый поток может быть записан.
write(s)	Записывает строку s в файл и возвращает количество записанных символов.
writelines(lines)	Записывает список строк в файл.

Закрепление материала

- Что такое файл?
- Что такое файловый объект (поток)?
- При помощи какой встроенной функции можно открыть файл?
- Зачем нужно закрывать файлы?
- При помощи какой конструкции языка Python можно автоматически закрывать файлы после того, как они больше не нужны?
- Какие существуют режимы открытия файлов?
- Как прочитать данные из файла?

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Как записать данные в файл?

Дополнительное задание

Задание
Создайте список товаров в интернет-магазине. Сериализуйте его при помощи pickle и сохраните в JSON

Самостоятельная деятельность учащегося

Задание 1
Напишите скрипт, который создаёт текстовый файл и записывает в него 10000 случайных действительных чисел. Создайте ещё один скрипт, который читает числа из файла и выводит на экран их сумму.

Задание 2
Модифицируйте исходный код сервиса по сокращению ссылок из предыдущих двух уроков так, чтобы он сохранял базу ссылок на диске и не «забывал» при перезапуске. При желании можете ознакомиться с модулем shelve (https://docs.python.org/3/library/shelve.html), который в данном случае будет весьма удобен и упростит выполнение задания.

Рекомендуемые ресурсы

https://docs.python.org/3/tutorial/inputoutput.html
https://docs.python.org/3/reference/datamodel.html#context-manager
https://docs.python.org/3/library/functions.html#open

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Файл

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com