Ввод и вывод

№ урока:	8	**Курс:**	Python Essential

Средства обучения:	Python 3; интегрированная среда разработки (PyCharm или Microsoft Visual Studio + Python Tools for Visual Studio)

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, сохранять и загружать данные в формате JSON, сериализовать данные при помощи pickle.

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
- Использовать модули json, pickle, сериализовать и десериализовать данные

Содержание урока

1. Файлы
2. Текстовые и бинарные файлы
3. Открытие, закрытие файлов
4. Режимы открытия файлов
5. Считывание из файлов и запись в файлы
6. Использование конструкции with; менеджеры контекста
7. Файловые объекты StringIO и BytesIO
8. Сохранение и загрузка данных в формате JSON
9. Сериализация с помощью pickle

Резюме

- *Файл* – именованная область данных на носителе информации.
- *Файловый объект* – объект, предоставляющий файл-ориентированный API (методы read(), write() и т.д.) для доступа к ресурсу. В зависимости от способа создания, файловый объект может предоставлять доступ к реальному файлу на диске или другому виду устройства хранения или передачи данных (стандартные потоки ввода/вывода, буферы в памяти, сокеты и т.д.). Файловые объекты также называют *потоками*.
- К стандартным потокам относятся sys.stdin (стандартный поток ввода), sys.stdout (стандартный поток вывода) и sys.stderr (стандартный поток вывода ошибок), которые по умолчанию привязаны к терминалу, но могут быть перенаправлены в файл на диске или стандартный поток другого приложения.
- Как уже говорилось ранее, Python 2 и Python 3 обрабатывают строки по-разному.
- В Python 2 класс *str* представляет собой строку как последовательность байтов, он же и используется в тех случаях, когда требуется представить в памяти последовательность байтов, которая не является строкой. Для строк в кодировке Unicode используется класс *unicode*.
- В Python 3 класс str представляет Unicode-строки, и именно Unicode является стандартной кодировкой для интерпретатора. Для байтовых строк и последовательностей байтов был введён новый тип *bytes*.
- В обоих версиях языка существует также тип данных, являющийся изменяемым аналогом типов str и bytes в Python 2 и Python 3 соответственно: *bytearray*.
- В Python 2 файловые объекты представляются типом *file*.

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- На уровне типов данных в Python 2 нет отличия между текстовыми и бинарными файлами. При открытии можно указать текстовый либо бинарный режим, но это влияет только на преобразования концов строк при выполнении под ОС Windows, а под Unix-системами, где преобразования концов строк не требуются, не влияет ни на что.
- В Python 3 существует три вида файловых объектов: текстовые файлы (text files), «обычные» (небуферизированные) бинарные файлы (raw binary files) и буферизированные бинарные файлы (buffered binary files). Разные виды потоков представляются соответствующими классами модуля io.
- Модуль io был обратно портирован в последние версии Python 2, поэтому в Python 2 также при желании можно использовать систему ввода-вывода, аналогичную Python 3.
- Текстовые файлы записывают и считывают данные типа str и автоматически выполняют преобразования кодировок и концов строк. Бинарные файлы записывают и считывают данные типов bytes и bytearray и не производят никаких манипуляций с данными: всё записывается и считывается в таком же виде, как и сохраняется.
- Обычно для создания файлового объекта используется встроенная функция open.
- Сигнатура функции в Python 2: open(file, mode='r', buffering=-1).
- Сигнатура функции в Python 3 (и в Python 2 при использовании функции io.open): open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None).
- Основные параметры:
    - file – имя файла или файловый дескриптор;
    - mode – режим открытия файла;
    - encoding – кодировка файла;
    - buffering – использовать ли буферизацию: отрицательное число (по умолчанию, указывать явно не нужно) – стандартное значение для данного вида файлового объекта, 0 – отключить буферизацию, 1 – построчная буферизация (для текстовых файлов), другое значение – включить буферизацию и задать соответствующий размер буфера.
- Обязательным параметром является только первый. Чаще всего функция open() используется с двумя параметрами.
- mode может начинаться с символов «r» (чтение), «w» (запись, очищает файл, если уже существует), «x» (исключительное создание, неуспешно, если файл уже существует), «a» (добавление, запись в конец файла).
- Также параметр mode может иметь вторую букву для определения типа файла: «t» для текстового (по умолчанию) и «b» для бинарного.
- Также можно добавить символ «+» для открытия в режиме чтения и записи одновременно.
- Порядок последних двух символов не имеет значения: «rb+» и «r+b» задают один и тот же режим.
- После окончания работы с файлом следует обязательно его закрыть при помощи метода close(), особенно если он был открыт для записи. При использовании буферизированного вывода данные, которые записываются в файл, не попадают в него сразу, а записываются в буфер. Содержимое буфера записывается в файл при его заполнении или вызове методов flush() или close(). Кроме того, если файл открыт для записи, он будет заблокирован для открытия для записи другими процессами до момента закрытия. Все открытые файлы автоматически закрываются при удалении соответствующих файловых объектов из памяти сборщиком мусора интерпретатора Python и при завершении работы самого интерпретатора, однако следует держать файлы открытыми минимально требуемое время.
- Для построчного считывания текстового содержимого из файлов можно воспользоваться методом readline() для считывания строки, readlines() для получения списка всех строк или использовать файловый объект как итератор.
- Метод writelines(lines) записывает список строк в файл. Символы концов строк автоматически не добавляются.
- Методы read(size=-1) и write(data) считывают и записывают данные (байты или символы). Если при вызове метода read() не указать желаемое количество байтов или символов, считываются все данные до конца файла.
- Метод readinto(arr) бинарных файлов считывает данные в массив (bytearray или array.array) arr. Количество считываемых данных определяется размером массива.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- Метод tell() возвращает текущую позицию считывания/записи в файле. Метод seek(offset, whence) устанавливает её. Параметр offset задаёт отступ, а whence – точку, от которой данный отступ считается: io.SEEK_SET (0) – начало файла, io.SEEK_CUR (1) – текущая позиция, io.SEEK_END (2) – конец файла.
- В Python 3 у функции print есть именованный параметр file (по умолчанию – sys.stdout), который указывает, куда следует направить вывод.
- В Python 2 print является оператором. После данного ключевого слова через запятую указываются выражения, значения которых необходимо вывести через символ пробела. В конце выводится символ новой строки, но только если не указать запятую в конце. Для направления вывода в файл в качестве первого аргумента задаётся файловый объект, перед которым указываются два символа «>>».
- Полезными являются классы io.StringIO и io.BytesIO, которые представляют собой потоки для считывания и записи в строки или байтовые строки в памяти. Они могут быть использованы для того, чтобы использовать строки и байтовые строки в качестве текстовых и бинарных файлов.
- При возникновении ошибки при работе с файлами выбрасывается исключение OSError (IOError в Python 2).
- Файловые объекты являются менеджерами контекста.
- *Менеджер контекста (context manager)* – это объект, который описывает определённый контекст, который устанавливается при выполнении оператора with. Менеджеры контекста используются для сохранения и восстановления глобального состояния, блокирования и разблокирования ресурсов, автоматического закрытия файлов и т.д.
- Для того, чтобы объект был менеджером контекста, он должен описывать два специальных метода:
    - __enter__(self) – вход в контекст. Выполняется оператором with. Значение, которое вернул метод, привязывается к имени, указанному после ключевого слова as оператора with.
    - __exit__(self, exc_type, exc_value, traceback) – вызывается при выходе из контекста. Параметры описывают исключение, которое прервало выполнение кода в контексте. Если никакое исключение не возникло, то последние три параметра равны None. Если данный метод обработал исключение, и оно не должно распространяться дальше, он должен вернуть истинное значение. Если же исключение должно быть выброшено, данный метод не должен его выбрасывать сам, это является обязанностью вызывающей стороны (чаще всего, оператора with).
- *Сериализация* – это процесс сохранения объектов в двоичном или строковом виде для хранения, передачи и восстановления. Обратный процесс называется *десериализацией*.
- Для сохранения разнообразных данных в текстовом виде можно использовать формат *JSON* (JavaScript Object Notation), который является стандартом де-факто в веб-приложениях и довольно популярен в других областях. Он поддерживается (на уровне языка, стандартной библиотеки языка или, по крайней мере, существующих сторонних библиотек) всеми современными языками программирования и знаком многим разработчикам, что делает его отличным выбором для сохранения данных для передачи между приложениями, написанными на разных языках.
- Функция dumps модуля json сохраняет JSON-представление объекта в строку. Функция dump – в текстовый файл.
- Функция loads модуля json загружает объект из строки. Функция load – из текстового файла.
- Модуль json позволяет сериализовать словари, списки, кортежи, строки, целые и вещественные числа, перечисления, булевские значения и None. Для поддержки других типов данных следует расширить классы JSONEncoder и JSONDecoder.
- Модуль pickle реализует бинарные протоколы для сериализации и десериализации объектов Python.
- Pickle поддерживает сериализацию огромного количества типов данных, включая созданные пользователем (многие из которых поддерживаются автоматически, но в более сложных случаях объект должен реализовать определённые специальные методы), корректно обрабатывает ссылки на уже сериализованные объекты и сериализацию рекурсивных типов данных.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- В отличие от JSON, который может быть прочитан где угодно, pickle специфичен для Python и не может быть использован для обмена данными с приложениями, написанными на других языках.
- Кроме того, недостатком pickle является его небезопасность: десериализовать можно только те данные в формате pickle, которые получены из надёжного источника, иначе есть риск выполнения произвольного кода.
- Функции dump, dumps, load и loads модуля pickle аналогичны по своему предназначению соответствующим функциям модуля JSON, но работают с байтовыми строками и бинарными файлами.
- Опциональный параметр protocol данных функций задаёт версию протокола. Последнюю версию протокола можно получить как константу pickle.HIGHEST_PROTOCOL, текущую версию по умолчанию – pickle.DEFAULT_PROTOCOL.
- На момент написания данного урока существует пять версий протокола: 0 и 1 – это устаревшие версии, которые использовались в Python 2.2 и ниже; 2 – это основная версия протокола для Python 2; 3 – версия протокола, которая появилась в Python 3, стандартный протокол в Python 3 на текущий момент, не может быть десериализован в Python 2; 4 – версия протокола, появившаяся в Python 3.4, поддерживает очень большие по объёму памяти объекты, поддерживает большее количество типов объектов, добавлены некоторые оптимизации.

Закрепление материала

- Что такое файл?
- Что такое файловый объект (поток)?
- Чем отличаются строки и байтовые строки в Python 3? Какие типы данных соответствуют им в Python 2?
- Чем отличаются текстовые и бинарные файлы в Python 2? В Python 3?
- 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?
- Как записать данные в файл?
- Что такое JSON?
- Что такое pickle?

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 решение предыдущего задания так, чтобы оно работало не с текстовыми, а бинарными файлами.

Задание 3
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.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

Рекомендуемые ресурсы

Документация Python 3
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
https://docs.python.org/3/library/io.html
https://docs.python.org/3/library/json.html
https://docs.python.org/3/library/pickle.html

Документация Python 2
https://docs.python.org/2/tutorial/inputoutput.html
https://docs.python.org/2/reference/datamodel.html#context-manager
https://docs.python.org/2/library/functions.html#open
https://docs.python.org/2/library/io.html
https://docs.python.org/2/library/json.html
https://docs.python.org/2/library/pickle.html

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

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com