Лекция 5

Работа с файлами в Python

2015

* Работа с файлами и файловой системой в Python

1. Как открыть файл.
2. Базовые файловые методы.
3. Стандартный ввод/вывод.
4. Произвольный доступ.
5. Построчная работа с файлами.
6. Закрытие файла.
7. Итерация.
8. Pickling.
9. Работа с файловой системой.

Python. Как открыть файл. Функция open

f = open('my_file', 'w')

‘r’ — открытие на чтение (выступает значением по умолчанию).

‘w’ — открытие на запись, содержимое файла удаляется, если файла с таким именем нет, то он автоматически создается.

‘x’ — открытие на запись, если файл не существует, иначе исключение.

‘a’ — открытие на дозаписывание, информация добавляется в конец уже существующей информации в файле.

‘b’ — открытие в двоичном режиме.

‘t’ — открытие в текстовом режиме (выступает значением по умолчанию).

‘+’ — открытие на чтение и запись.

* Кроме того, вышеперечисленные режимы могут быть объеденённы.
* По умолчанию режим ‘rt’.
* Если вы хотите произвести чтение в двоичном режиме, то укажите ‘rb’. Еще существует аргумент **encoding** — он задает кодировку и используется только в текстовом режиме чтения файла.

Python. Как открыть файл. Функция open

Пример. Прочитать 6 символов

```
>>> f = open('file.txt', 'r')
>>> print f.read(12)
```

- В примере работает метод read, который считывает информацию с файла.
- Если в его аргументе ничего не указать, то он выведет всю информацию.
- Но можно обращаться к файлу с целью побитового вывода.
- Вспоминаем, что кириллица занимает по 2 байта на символ и выводим первые 6 букв.

Python. Как открыть файл. Функция open

Пример. Чтение по строкам

```
f = open('file.txt', 'r')
for line in f:
        print line
f.close()
```

Python. Запись в файл

Пример. Запись в файл

```
f = open('file.txt', 'w')
f.write('string1n') #n - перенос строки
f.write('sting2')
f.close() # Проверяем, записались ли значения
f = open('file.txt', 'r')
print f.read()
f.close()
```

- Открываем файл в режиме запись ‘w’, при этом все содержимое файла удалиться.
- Записываем в него слово ‘string’ с помощью метода **write()**.

Python. Запись в файл

Пример. Дозапись в файл

```
f = open('file.txt', ‘a')
f.write('Hello, ') #n - перенос строки
f.write('World!')
f.close() # Проверяем, записались ли значения
f = open('file.txt', 'r')
print f.read()
f.close()
```

Python. Чтение из файла

```
f = open('my_file', 'r')

print(f.read())

f.close()
```

Python. Произвольный доступ

По умолчанию метод read() читает данные последовательно по порядку, от начала и до конца файла.

Для произвольного доступа к файлу есть функция seek:

seek(offset[, whence])

offset – смещение в байтах относительно начала файла;

whence – по умолчанию равен нулю, указывает на то, что смещение берется относительно начала файла.

Python. Произвольный доступ. Пример

```
f = open(r'my_file', 'w')
f.write('012345678901234567890')
f.seek(5)
f.write('Hello, World!')
f.close()
f = open(r'my_file')
f.read()
```

Функция tell() возвращает текущую позицию файла

Python. Построчная работа с файлами

file.readline() - прочитать одну строку.

Функция readline() без параметра читает всю строку, наличие параметра указывает функции максимальное число символов строки, которое будет прочитано.

file.readlines() - прочитать все строки и вернуть список строк

file.writelines() - записать строки в файл

* Прочитать файл и записать его содержимое в другой файл

```
f = open(r'my_file')

lines = f.readlines()

f.close()

lines[0] = "This is a my_file2 \n" # изменяем 1-ю строку

f = open(r'my_file2','w')

f.writelines(lines)

f.close()
```

Python. Закрытие файла

* Для закрытия файла есть метод close(). Обычно файл закрывается сам после того, как вы выходите из программы, но файлы нужно закрывать вручную по нескольким причинам.

1. Питон может буферизировать запись в файл ваших данных, что может привести к неожиданным эффектам и возникновению ошибок.
2. У операционной системы есть ограничение на число одновременно открытых файлов.
3. При доступе к файлу из разных мест одновременно и на чтение, и на запись необходимо синхронизировать файловые операции. Буферизация записи может привести к тому, что запись уже произошла, а данных в файле еще нет.

Python. Закрытие файла

* Для закрытия файла есть метод close(). Обычно файл закрывается сам после того, как вы выходите из программы, но файлы нужно закрывать вручную по нескольким причинам.

1. Питон может буферизировать запись в файл ваших данных, что может привести к неожиданным эффектам и возникновению ошибок.
2. У операционной системы есть ограничение на число одновременно открытых файлов.
3. При доступе к файлу из разных мест одновременно и на чтение, и на запись необходимо синхронизировать файловые операции. Буферизация записи может привести к тому, что запись уже произошла, а данных в файле еще нет.

* Python. Гарантированное закрытие файла

Конструкция

```
myfile = open(filename, ‘w’)
try:
...обработка myfile...
finally:
myfile.close()
```

Еще одна новая конструкция

```
with open(filename, ‘w’) as myfile:
... обработка myfile, закрывается автоматически после выхода...
```

этот прием гарантирует закрытие файла в любом случае, независимо от возникновения исключения

* Python. Итерация

Пример. Побайтовое чтение из файла

```
f = open(filename)
while True:
        char = f.read(1)
        if not char: break
        process(char)
f.close()
```

* Python. Итерация

Пример. Построчное чтение

```
f = open(filename)
while True:
        line = f.readline()
        if not line: break
        process(line)
f.close()
```

* Python. Итерация

Пример. Файл в роли итератора

```
for line in open(filename):
    process(line)
```

Python. Работа с файлами. Pickling

* **Сериализация** (в программировании) — процесс перевода какой-либо структуры данных в последовательность битов.
* Обратной к операции сериализации является операция **десериализации** (структуризации) — восстановление начального состояния структуры данных из битовой последовательности.

* Сериализация используется для передачи объектов по сети и для сохранения их в файлы. Например, нужно создать распределённое приложение, разные части которого должны обмениваться данными со сложной структурой. В таком случае для типов данных, которые предполагается передавать, пишется код, который осуществляет сериализацию и десериализацию. Объект заполняется нужными данными, затем вызывается код сериализации, в результате получается, например, XML-документ. Результат сериализации передаётся принимающей стороне по, скажем, электронной почте или HTTP. Приложение-получатель создаёт объект того же типа и вызывает код десериализации, в результате получая объект с теми же данными, что были в объекте приложения-отправителя.

Python. Работа с файлами. Pickling

Практически любой тип объекта может быть сохранен на диске в любой момент его жизни, а позже прочитан с диска. Для этого есть модуль pickle:

```
import pickle
t1 = [1, 2, 3]
s = pickle.dumps(t1)
t2 = pickle.loads(s)
print (t2)

# t1 и t2 два разных объекта
```

Python. Работа с файлами. Pickling

* Модуль pickle преобразует объект Python, находящийся в оперативной памяти, в последовательность или в строку байтов, которую можно записать в любой объект, подобный файлу.

* Кроме того, модуль pickle знает, как восстановить оригинальный объект в памяти, получив последовательность байтов, то есть мы получаем обратно тот же самый объект.

* В некотором смысле модуль pickle позволяет избежать необходимости разрабатывать специальные форматы представления данных – последовательный формат, реализованный в этом модуле, достаточно универсален и эффективен для большинства применений.

Python. Работа с файлами. Pickling

* Что может сохранять модуль pickle?
* Все встроенные типы данных Python: тип boolean, Integer, числа с плавающей точкой, комплексные числа, строки, объекты bytes, массивы байт, и None.
* Списки, кортежи, словари и множества, содержащие любую комбинацию встроенных типов данных
* Списки, кортежи, словари и множества, содержащие любую комбинацию списков, кортежей, словарей и множеств содержащий любую комбинацию встроенных типов данных (и так далее, вплоть до максимального уровня вложенности, который поддерживает Python).
* Функции, классы и экземпляры классов (с caveats).

* Python. Работа с файловой системой

Примеры

```
import os
cwd = os.getcwd()  # возвращает текущий каталог
print(cwd)

# Проверка наличия файла в текущем каталоге:
os.path.exists('my_file')

# Проверка наличия файла в текущем каталоге:
print(os.listdir(os.getcwd()))
```