Input/Output

Computer & Information Sciences

W. H. Bell

Overview

- Input/Output examples.
- Text files.
- CSV files.
- JSON files.
- Pickle files.
- Summary.

Input/Output examples

- Files.
- Text.
- Binary.
- Databases.
- SQL and NoSQL.
- Network connections.
- Text and binary components.

Input/Output examples

Python libraries exist for other common file formats.

- CSV (Comma-separated values)
- Used to store tabulated data.
- JSON (JavaScript Object Notation)
- Web service communication and NoSQL database storage.
- XML
- Java web services, some file formats (Office).
- Pickles
- Python binary file format.

File paths

- Different standards on different operating systems.
- Windows - `c:\users\someuser`
- Linux and Mac - `/home/someuser`
- Need to use Python path functionality to safely join paths.
- Corresponding Python function is `os.path.join()`

File exists

The test file was created beforehand.

```
import os.path

def fileExists(fileName):
     return os.path.isfile(fileName)

if __name__ == "__main__":
     print(fileExists("my-file.txt"))
```

Output

```
True
```

Text files: write a file

```
import os.path

def writeFile(fileName):
     outputFile = open(fileName,"w")
     outputFile.write("A text string" + "\n")
     outputFile.close()
     return os.path.isfile(fileName)

if __name__ == "__main__":
     print(writeFile("written-file.txt"))
```

Open file for writing.

Write to the file.

Test if the file exists.

Output

```
True
```

Text files: append to a file

```
import os.path

def writeFile(fileName):
     outputFile = open(fileName,"a")
     outputFile.write("Another text string" + "\n")
     outputFile.close()

if __name__ == "__main__":
     fileName = "append-file.txt"
     writeFile(fileName)
     print(fileName + " is " + str(os.path.getsize(fileName)) + " Bytes.")
```

Open file to be appended to.

Check file size

Output

```
append-file.txt is 20 Bytes.
```

```
append-file.txt is 40 Bytes.
```

Text files: read from a file

```
def readFile(fileName):
    inputFile = open(fileName,"r")
    content = inputFile.read()
    inputFile.close()
    return content.strip()

if __name__ == "__main__":
    print(readFile("written-file.txt"))
```

Open file for reading.

Read the file.

Remove the new line.

Output

```
A text string
```

CSV files: write to a file

```
import csv

def writeCSV(fileName):
     csvFile = open(fileName, "w", newline='')
     csvWriter = csv.writer(csvFile, delimiter=',', quotechar='"',
quoting=csv.QUOTE_NONNUMERIC)
     csvWriter.writerow(["Host", "IP"])
     csvWriter.writerow(["localhost", "127.0.0.1"])
     csvFile.close()

if __name__ == "__main__":
     writeCSV("my-file.csv")
```

Output text file.

Safe set of options.

my-file.csv

```
"Host","IP"
"localhost","127.0.0.1"
```

CSV files: read from a file

```
import csv

def readCSV(fileName):
     csvFile = open(fileName, "r", newline='')
     csvReader = csv.reader(csvFile, delimiter=',', quotechar='"')
     for row in csvReader:
          print(row)
     csvFile.close()

if __name__ == "__main__":
     readCSV("my-file.csv")
```

Input text file.

Each row is a list.

Output

```
['Host', 'IP']
['localhost', '127.0.0.1']
```

JSON files: write to a file

```
import json

def writeJSON(fileName):
    hosts = {}
    hosts["localhost"] = "127.0.0.1"
    outputFile = open(fileName, "w", encoding="utf-8")
    json.dump(hosts, outputFile, ensure_ascii=False, indent=4)
    outputFile.close()

if __name__ == "__main__":
    writeJSON("my-file.json")
```

Output text file.

Easy to read and flexible.

my-file.json

```
{
  "localhost": "127.0.0.1"
}
```

JSON files: read from a file

Can store lists, dictionaries, float, int, string and boolean types.

```
import json

def readJSON(fileName):
    inputFile = open(fileName, "r", encoding="utf-8")
    jsonData = json.load(inputFile)
    inputFile.close()
    return jsonData

if __name__ == "__main__":
    print(readJSON("my-file.json"))
```

Input text file.

Read into memory.

Output

```
{'localhost': '127.0.0.1'}
```

Pickle files

- Write Python objects to binary file.
- Read Python objects from binary file.
- Must trust input Pickle – malicious use is possible.
- Need to support old data – schema migration.

Pickle: write to file

```
import pickle

class Host():
    def __init__(self):
        self.name = "localhost"

def writePickle(fileName):
    pc = Host()
    outputFile = open(fileName, "wb")
    pickle.dump(pc, outputFile)
    outputFile.close()

if __name__ == "__main__":
    writePickle("my-pickle.bin")
```

Define a class that holds some data.

Output binary file.

Write the Pickle.

Pickle: read from file

```
import pickle

class Host():
    def __init__(self):
        self.name = "localhost"

def readPickle(fileName):
    inputFile = open(fileName, "rb")
    pc = pickle.load(inputFile)
    inputFile.close()
    print("Host.name = \"" + str(pc.name) + "\"")

if __name__ == "__main__":
    readPickle("my-pickle.bin")
```

Define a class that holds some data.

Input binary file.

Read the Pickle.

Output

```
Host.name = "localhost"
```

Other functionality

- Python provides functions to list files in directory.
- Can check the modification timestamp of files.
- Can check if directories exist.
- Many other file formats are supported.
- E.g. Excel files can be read using xlrd or openpyxl.

Summary

- Provided input/output examples.
- Discussed common file formats.
- Python functions needed to write and read them.