Data Encoding

Overview

- What is data encoding and why do we do it?
- CSV
- XML
- JSON

Learning Objectives

- To be able to explain what data encoding is and why we do it
- To gain an understanding of what CSV, XML & JSON are

What is data encoding?

Encoding is the process of converting data into a specified format.

Decoding is the reverse process - to extract information from the converted format.

Common formats:

- JPG, MP4, AVI
- Morse code, Braille
- JSON, XML, CSV
- Analog, Digital

The process of converting data into a format required for a number of information processing needs - e.g. transmission, storage.

During session - will learn about some of most common formats that data is stored in - will use in your career!

Why do we encode data?

- Easier to store for computers (humans work with text, computers work with bytes)
- Removes redundancies from data (such as whitespace) so data size decreases
- Smaller data means it's more efficient to store and retrieve data

We encode our data as a means of organizing our data.

Not about keeping information secret, but rather about ensuring that the data is able to be properly consumed - we're not encrypting.

CSV (Comma Separated Values)

- A plain text file that uses specific structuring to arrange **tabular** data
- Only contains text data
- Each line of the file is a data record
- Is separated by a *delimiter* (comma, colon, tab etc.)

```
first_name, last_name, age
John,       Smith,     20
Sally,      Bloggs,    30
```

Each encoding format has its own rules and special characters (reserved characters used for special purposes) - like human language, where we have grammatical rules and an alphabet.

Excel will allow you to convert from csv to xls (and vice-versa).

CSV - Dealing with commas

What about data that contains a comma?

Well, in that scenario, we quote the data

```
first_name, last_name, age, test_scores
John,       Smith,     20,  "80, 76, 92"
Sally,      Bloggs,    30,  "72, 84, 90"
```

CSV - Dealing with double quotes

How about data that contains a double quote?

We 'escape' the quote by using two of them together

```
tv,        size
"Samsung", "24"" TV"
"LG",      "41"" TV"
```

CSV in Python

Luckily for us, Python has its own csv library to read and write to/from CSV files.

CSV - Reading a File

Looking back to the previous module, opening a CSV is as simple as:

```
import csv

with open(filename) as file:
  reader = csv.reader(file, delimiter=',')
  for row in reader:
    print(row)
```

`reader` is a function in the CSV library which returns an object which will iterate over the lines in the given file.

Reading to a Dictionary

We can read our CSV directly into a dictionary using `DictReader`:

```
import csv
with open("people.csv", 'r') as file:
  csv_file = csv.DictReader(file)
  for row in csv_file:
    print(row)
```

Output:

```
{'first_name': 'John', 'last_name': 'Smith', 'age:' 20}
{'first_name': 'Sally', 'last_name': 'Bloggs', 'age:' 30}
```

CSV - Writing to a File

```
import csv

with open('people.csv', mode='w') as file:
  writer = csv.writer(file, delimiter=',')

  writer.writerow(['Joe', 'Bloggs', 40])
  writer.writerow(['Jane', 'Smith', 50])
```

Writing from a Dictionary

```
with open('people.csv', mode='w') as file:
  fieldnames = ['first_name', 'last_name', 'age']
  writer = csv.DictWriter(file, fieldnames=fieldnames)
```

```
    writer.writeheader()
    writer.writerow({
      'first_name': 'Jan',
      'last_name': 'Smith',
      'age': 60
    })
```

`fieldnames` is required when writing from a dictionary.

Quiz Time! 🤓

When data is encoded in CSV format, what do we call the character (such as a comma or tab), which is used to separate different fields within a record?

1. `separator`
2. `limiter`
3. `fielder`
4. `delimiter`

Answer: `4`

You want to import the contents of a CSV file, and view the imported contents in dictionary format. Which of the following lines is likely to appear in your code?

1. `reader = csv.reader(file, delimiter=',')`
2. `csv_file = csv.DictReader(file)`
3. `writer = csv.writer(file, delimiter=',')`
4. `writer = csv.DictWriter(file, fieldnames=fieldnames)`

Answer: `2`

Exercise

Distribute exercise file.

XML

- Stands for 'eXtensible Markup Language'
- Like HTML but for storing data rather than displaying data
- Multidimensional
- A form of semi-structured data

Both HTML and XML are XML markup languages that are designed to be both human-readable and machine-readable.

XML documents can be validated against a schema (constrain which elements & attributes may be used & their allowable parent/child relationships).

Semi-structured because data does not reside in fixed fields or records, but does contain elements that can separate data into various hierarchies.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
  <person>
    <first_name>John</first_name>
    <last_name>John</last_name>
    <age>20</age>
  </person>
  <person>
    <first_name>Sally</first_name>
    <last_name>Bloggs</last_name>
    <age>30</age>
  </person>
</people>
```

First line is the XML declaration - not mandatory.

Tags - start and end for elements (content shown in between).

Attributes are designed to contain data related to a specific element.

XML Advantages

- Can store highly structured data
- Human readable
- Well understood and used

XML Disadvantages

- 'Wordy' - metadata takes up a lot of space
- Becomes progressively inefficient the more complicated the structure of the data

An element links to metadata by referencing each applicable metadata element's ID in inherited attributes.

JSON

- JavaScript Object Notation
- A file format that uses human-readable text to store and transmit data objects
- Can also store semi-structured data like XML

- Also maps to multidimensional data

Lightweight data interchange format.

Easy for humans to read and write.

Easy for machines to parse (taking a big lump and breaking into meaningful chunks) and generate.

Based on Javascript, but completely language independent.

JSON Example

```
{
  "people": [
    {
      "person": {
        "first_name": "John"
      }
    },
    {
      "person": {
        "first_name": "Sally"
      }
    }
  ]
}
```

Key/Value pairs separated by ':'.

Different pairs separated by ','.

Key is a string in double quotes.

Value can be string, boolean, another JSON object, an array or null.

JSON objects are surrounded in {}.

Arrays are surrounded in [].

JSON Advantages

- Human readable but much less wordy than XML
- Has become a data transfer and storage "standard"

JSON Disadvantages

- JSON isn't as robust a data structure as XML is.
- Can't use comments

JSON has become the standard because its simple design and flexibility makes it easy to read and understand. Also easy to manipulate in the programming language of your choice.

No ability to add comments or attribute tags, which limits ability to annotate data structure or add useful metadata.

Quiz Time! 🤓

Which of these is valid JSON?

```
{ // 1
  person: {
    first_name : "John"
  }
}

{ // 2
  "person": {
    "first_name" = "John"
  }
}

{ // 3
  "person": {
    "first_name": "John"
  }
}

{ // 4
  "person": [
    "first_name": ["John"]
  ]
}
```

Answer: 3

Learning Objectives Revisited

- To be able to explain what data encoding is and why we do it
- To gain an understanding of what CSV, XML & JSON are

Terms and Definitions Recap

CSV: A delimited text file that uses commas to separate values.

XML: a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

JSON: An open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).

Terms and Definitions Recap

Encoding: A system of rules to convert information into another form for communication through a communication channel or storage in a storage medium.

Parse: The process of analysing a string of symbols, either in natural language, computer languages or data structures, conforming to the rules of a formal grammar.

Further Reading

Reading and Writing CSV Files in Python

Reading and Writing XML Files in Python

Working with JSON Data in Python