Processing CSV, JSON

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Input for Programs

- Programs solve problems - for specific instance of the problem inputs for that instance are to be given as input to the program
- To provide input data to programs, we have seen
  - Input through terminals - limiting
  - Input through files - versatile; you need to get data in a local file
- For input() and File I/O, python provided constructs, which the interpreter has to implement
  - To implement, interpreter has to work with underlying OS which controls files and terminal I/O
  - That is why different interpreters needed for different OSs

Data Formats have Proliferated

- There are many different standard data formats defined over the years - often for specific purposes
- One can generally get such data in a file. Some formats:
- text , csv, JSON, jpeg, mpeg, binary, …
- To process this data, you have to read the data from the file, and based on the format extract the data in processable structures like strings, lists, dictionaries, ..
- For many data formats, standard libraries are provided by python
- We will briefly look at two - CSV and JSON
- Then we will move to taking input from the web

Processing CSV Files

- CSV = Comma Separated Values
- A lot of data is available in csv format - all spreadsheets can be saved as csv
- CSV is a text file of the type:
- First line: header1, header2, …
- Following lines: v1, v2, v3, …
- It is useful to be able to process csv files, and create csv files
- Python provides a library for easy process
- pandas also provides csv processing
- We will briefly look at python provided library

Processing csv

- Must import the csv library

```
import csv
```

- A csv file opened as regular file (for reading, writing,...), say

```
f = open("data1.csv", "r")
```

- With csv library special functions provided to read/write
- For reading - first get a reader, then read, i.e.

```
csv_r = csv.reader(f, delimiter=',')
# delimiter can be omitted
```

- Then read row-by row using reader, just like we read files - each row will be a list of strings (first row will be headers, then data..)

```
for row in csv_r:
    <code-block>
```

Processing csv ...

- While csv library makes it easy to read - we must know the structure from before to be able to make sense of data i.e. we should know what each column means, and what a row represents
- If the structure of csv changes - our program has to change to handle the new structure.

Reading csv as a list (Each row as a list)

```
import csv

# Given the relative path
f = open("demo.csv", "r")

csv_r = csv.reader(f, delimiter=',')

for row in csv_r:
    print(row)

f.close()
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Output:

```
['Roll Number', ' Name']
['r1', ' abc']
['r2', ' pqr']
['r3', ' xyz']
```

Reading csv as a list (Each row as a list)

```
import csv

# Using with
with open("demo.csv", "r") as f:
    csv_r = csv.reader(f, delimiter=',')
    for row in csv_r:
        print(row)
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Output:

```
['Roll Number', ' Name']
['r1', ' abc']
['r2', ' pqr']
['r3', ' xyz']
```

Reading CSV as a dictionary

```
import csv

with open("demo.csv", "r") as f:
    for row in csv.DictReader(f):
        print(row)
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Output:

```
{'Roll Number': 'r1', ' Name': ' abc'}
{'Roll Number': 'r2', ' Name': ' pqr'}
{'Roll Number': 'r3', ' Name': ' xyz'}
```

Reading csv as a Dictionary...

```
import csv

# Open file in read mode
f = open('demo.csv', 'r')

# Create DictReader object
obj = csv.DictReader(f)
roll_num = []
name = []
for row in obj:
    roll_num.append(row['Roll Number'])
    name.append(row[" Name"])
print(roll_num) # ['r1', 'r2', 'r3']
print(name) # [' abc', ' pqr', ' xyz']
f.close()
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Example

- Given a csv file of marks for students, check if the total for each student is correct and then compute the class average.
- Marksheet.csv

Roll Number,Q1,Q2,Q3,Q4,Q5,Total
r1, 7, 5, 8, 9, 10, 39
r2, 6, 8, 9, 4, 10, 37
r3, 7, 8, 5, 8, 9, 37
r4, 6, 7, 8, 9, 10, 40
r5, 7,8, 4, 5, 6, 30

```
import csv
f = open("Marksheet.csv", "r")
csv_r = csv.reader(f, delimiter=',')
next(csv_r, None)  # Skip the header
avg = 0
error = [] # Store roll no.'s with incorrect total
c = 0
for row in csv_r:
    c=c+1 # Counting rows (for class average)
    marks = row[1:6]
    marks = [float(m) for m in marks]
    total = sum(marks)
    if float(row[6])==total:
        avg = avg+total
    else:
        error.append(row[0])
if len(error)==0:
    avg = avg/c
    print(f"Class Average: {avg}")
else:
    print(error)
f.close()
```

Writing CSV : Method 1 (writing list)

```
import csv

header = ["Roll Number", "Name"]
data = [["r1","abc"],["r2", "pqr"], ["r3", "xyz"]]

with open("demo.csv", "w") as f:
    csv_writer = csv.writer(f) # Create the writer object
    csv_writer.writerow(header) # Write the header
    csv_writer.writerows(data) # Write the data

# Writing header line is optional
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Writing CSV: Method 2 (writing dictionary)

- Use DictWriter; methods available are
- writeheader(DW)
- writerow(DW, rowdict)
- writerows(DW, rowdicts)

If you dont have the header data, you can just write the data - both in lists as well as dictionary

Processing CSV

- Sometimes csv uses other encodings (other than unicode) as some text may have special characters
- Thus, we have to open the file suitably

```
open('ExamMarks.csv', encoding='utf-8-sig')
```

Encodings:

- 'ascii' = English
- 'cp273' = German
- 'cp860' = Portuguese
- 'utf-8-sig' = All languages

Quiz – Single Correct

Which of the following is not a part of csv module?

A. readline()
B. writerow()
C. reader()
D. writer()

Quiz – Single Correct

Which of the following is not a part of csv module?

A. readline()
B. writerow()
C. reader()
D. writer()

Quiz (text)

In your BMI program, many students have entered their roll no, height, and weight - to compute their BMI. Your program was saving these as a list `bmi` (a list of [rollno, height, weight]). You want to save this in a csv file: stu.csv. Using csv package, give one statement for this - assume that stu.csv has been opened as f

Answer

- csv.writer(f).writerows(bmi)

Processing JSON Files

- JSON (JavaScript object notation) is a text representation of data
- It can be used to represent all standard types of data - scalar as well as structured like lists, dictionaries, etc, like XML
- It is commonly used to share structured data between programs
  - How do you share the dictionary you create in your program with your friend (e.g. so she can write some functions on that data)
- Now most websites are sharing data using JSON
- So processing JSON is a very useful capability - you will definitely need it when accessing data through APIs

JSON Format

- For understanding purposes, consider it as a nested dictionary
- The values in the dictionary can be:
  - Strings
  - Numbers (Integer, Float, etc.)
  - Boolean
  - Lists
  - Object (You will learn about this later)
  - null
  - Other dictionaries
- They typically exist as strings

JSON Example

```
{
  "first_name": "Sherlock",
  "last_name": "Holmes",
  "gender":"Male",
  "age": 36,
  "occupation": "Detective",
  "is_active": True,
  "address": {
    "home": "221B Baker Street",
    "city": "London",
    "country": "United Kingdom"
  },
  "contact_info": [
    {
      "type": "home",
      "number": "+44 123 456 7890"
    },
    {
      "type": "office",
      "number": "+44 987 654 3210"
    }
  ],
  "friends": ["John Watson", "Mrs Hudson", "Lestrade"]
}
```

JSON…

- A JSON object can only be: a dictionary or a list
- I.e. the JSON object is always like"{ …. } ", or "[…]"
- Within "{}" whatever is there should be a proper dictionary
- Within "[]" - must be a list
- If you want to represent many different objects in JSON - create a list of those objects and then convert it to JSON
- If there are multiple JSON objects - json package will complain

Using json with Strings

- Can convert a dictionary (or any other data structure) to a JSON format string - which can be passed/used or written on a file

```
json_str = json.dumps(<dictionary>, indent=4)
# indent provides how much indentation - useful when
printing
```

-
- If you have a JSON string in your program - that can be converted to a dictionary object also

  data = json.loads(str)

Reading/Writing JSON as Strings

- The json library provides functions - import it to use it
- A JSON file is opened as a regular file (say f), reading done by
- json.load(f): parses JSON data from f, populates a dictionary with the data and returns it; common use

```
data = json.load(f)
```

Reading JSON file

```
import json
f = open('demo.json', 'r')
data = json.load(f)
print(data)
# {'r1': 'abc', 'r2': 'pqr', 'r3': 'xyz'}
f.close()
```

demo.json

```
{
  "r1": "abc",
  "r2": "pqr",
  "r3": "xyz"
}
```

Writing JSON file

- Can write directly to a file:

```
json.dump(<dictionary>/<list>, f)
# f is opened for writing
```

Writing JSON file

```
import json

# Dictionary containing student roll number and name
students = {"r1":"abc", "r2": "pqr", "r3":"xyz"}

with open("demo.json", "w") as f:
     json.dump(students, f)
```

demo.json

```
{
  "r1": "abc",
  "r2": "pqr",
  "r3": "xyz"
}
```

JSON – Bigger Example

- Read the Sherlock Holmes JSON shown before.
- Increment Sherlock's age
- Edit his address to:

  28 A King Street, Manchester, United Kingdom

- Add another his office number ('office_2'):

  "+44 111 222 3333"

- Add "Molly Hooper" to his friends list
- Save the updated JSON.

JSON – Bigger Example – code

```
import json
with open('sherlock.json') as f:
    sherlock = json.load(f)


    sherlock['age'] += 1
    sherlock['address']['home'] = '28 A Kings Street'
    sherlock['address']['city'] = 'Manchester'
    sherlock['contact_info'].append({'type':'office_2', 'number':'+44
    111 222 3333'})
    sherlock['friends'].append('Molly Hooper')
```

JSON – Bigger Example – result


```
{
  "first_name": "Sherlock",
  "last_name": "Holmes",
  "gender":"Male",
  "age": 37,
  "occupation": "Detective",
  "is_active": true,
  "address": {
    "home": "28 A Kings Street",
    "city": "Manchester",
    "country": "United Kingdom"
  },
  "contact_info": [
    {
      "type": "home",
      "number": "+44 123 456 7890"
    },
    {
      "type": "office",
      "number": "+44 987 654 3210"
    },
    {
      "type": "office_2",
      "number": "+44 111 222 3333"
    }
  ],
  "friends": ["John Watson", "Mrs Hudson", "Lestrade", "Molly Hooper"]
}
```

Quiz – Single Correct

Which of the following is the correct way to read the given JSON file.

A. `data = json.loads(open("demo.json","r"))`

B. `data = json.load(open("demo.json","json"))`

C. `data = json.load_json(open("demo.json","r"))`

D. `data = json.load(open("demo.json","r"))`

demo.json

```
{
  "r1": "abc",
  "r2": "pqr",
  "r3": "xyz"
}
```

Quiz – Single Correct

Which of the following is the correct way to read the given JSON file.

A. `data = json.loads(open("demo.json","r"))`

B. `data = json.load(open("demo.json","json"))`

C. `data = json.load_json(open("demo.json","r"))`

D. `data = json.load(open("demo.json","r"))`

demo.json

```
{
   "r1": "abc",
   "r2": "pqr",
   "r3": "xyz"
}
```

Using JSON for Class Objects

- For an object of a class, how can we use JSON to transfer state
- The conversion of data from JSON object string is known as Serialization and its opposite string JSON object is known as Deserialization.
- For python types, serialization is taken care of by dumps and loads takes care of deserialization
- For user defined classes, we need to do more - need to convert objects to dictionary (using __dict__), but also requires some functions
- But can use JSON to transfer state
- Often, dictionary is used to transfer state - it is easy to convert an object to a dictionary

Converting CSV to JSON

```
import csv
import json

csv_file = open('demo.csv', 'r')
obj = csv.DictReader(csv_file)
data = [entry for entry in obj]
csv_file.close()

json_file = open('demo.json', 'w')
json.dump(data, json_file)
json_file.close()
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

demo.json

```
[
{"Roll Number":"r1", " Name":" abc"},
{"Roll Number":"r2", " Name":" pqr"},
{"Roll Number":"r3", " Name":" xyz"}
]
```

Summary

- CSV and JSON are used heavily to exchange data
- The csv and json package of python makes life simpler for programmers to use such data, and to share such data
- Better to use these packages than write your own - they are efficient, fine tuned, and reliable

Extra Slides

Reading csv as a list (Each column as a list)

```
# Method 2: Using pandas

import pandas as pd

f = pd.read_csv("demo.csv") # Read csv

# Convert the columns to list
roll_num = f["Roll Number"].to_list()
name = f[" Name"].to_list()

print(roll_num) # ['r1', 'r2', 'r3']
print(name) # [' abc', ' pqr', ' xyz']
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Writing CSV : Method 2

```
import pandas as pd

header = ["Roll Number", "Name"]
data = [["r1","abc"],["r2", "pqr"], ["r3", "xyz"]]

data = pd.DataFrame(data, columns=header)

data.to_csv('demo.csv', index=False)
```

demo.csv

```
Roll Number, Name
r1, abc
r2, pqr
r3, xyz
```

Example : Method 2

- Given a csv file of marks for students, check if the total for each student is correct and then compute the class average.
- Marksheet.csv

Roll Number,Q1,Q2,Q3,Q4,Q5,Total
r1, 7, 5, 8, 9, 10, 39
r2, 6, 8, 9, 4, 10, 37
r3, 7, 8, 5, 8, 9, 37
r4, 6, 7, 8, 9, 10, 40
r5, 7,8, 4, 5, 6, 30

```
import pandas as pd

df = pd.read_csv("Marksheet.csv") # Read csv
# Get the list of column headers
column_list = list(df.columns.values)
# Sum of columns: Q1, Q2, Q3, Q4, Q5
df["sum"] = df[column_list[1:6]].sum(axis=1)
if df['sum'].equals(df['Total']):
    print("Correct total.")
    average = df[["Total"]].mean(axis=0).to_list()[0]
    print(f"Class Average {average}")
else:
    print("Totalling mistake")
    s = df["sum"].to_list()
    t = df["Total"].to_list()
    r = df["Roll Number"].to_list()
    error = [r[i] for i in range(len(r)) if s[i]!=t[i]]
    # Get roll numbers of students with incorrect total
    print(error)
```