LECTURE 23

CSV AND JSON

MCS 275 Spring 2021
Emily Dumas

LECTURE 23: CSV AND JSON

Course bulletins:

- Worksheet solutions coming soon.
- Project 3 pitch in Monday's lecture.

NOTEBOOK

The (small) notebook of sample code from this lecture is here.

INSTALL PILLOW

Next week: Manipulating images with the Python package Pillow. To prepare, please

```
python3 -m pip install pillow
```

Or substitute the correct interpreter name for your platform.

If you have trouble, check the install instructions and let us know if you don't find a solution there.

MODULES

Python has a number of built-in modules to support reading and writing special file formats. We'll cover two of these today:

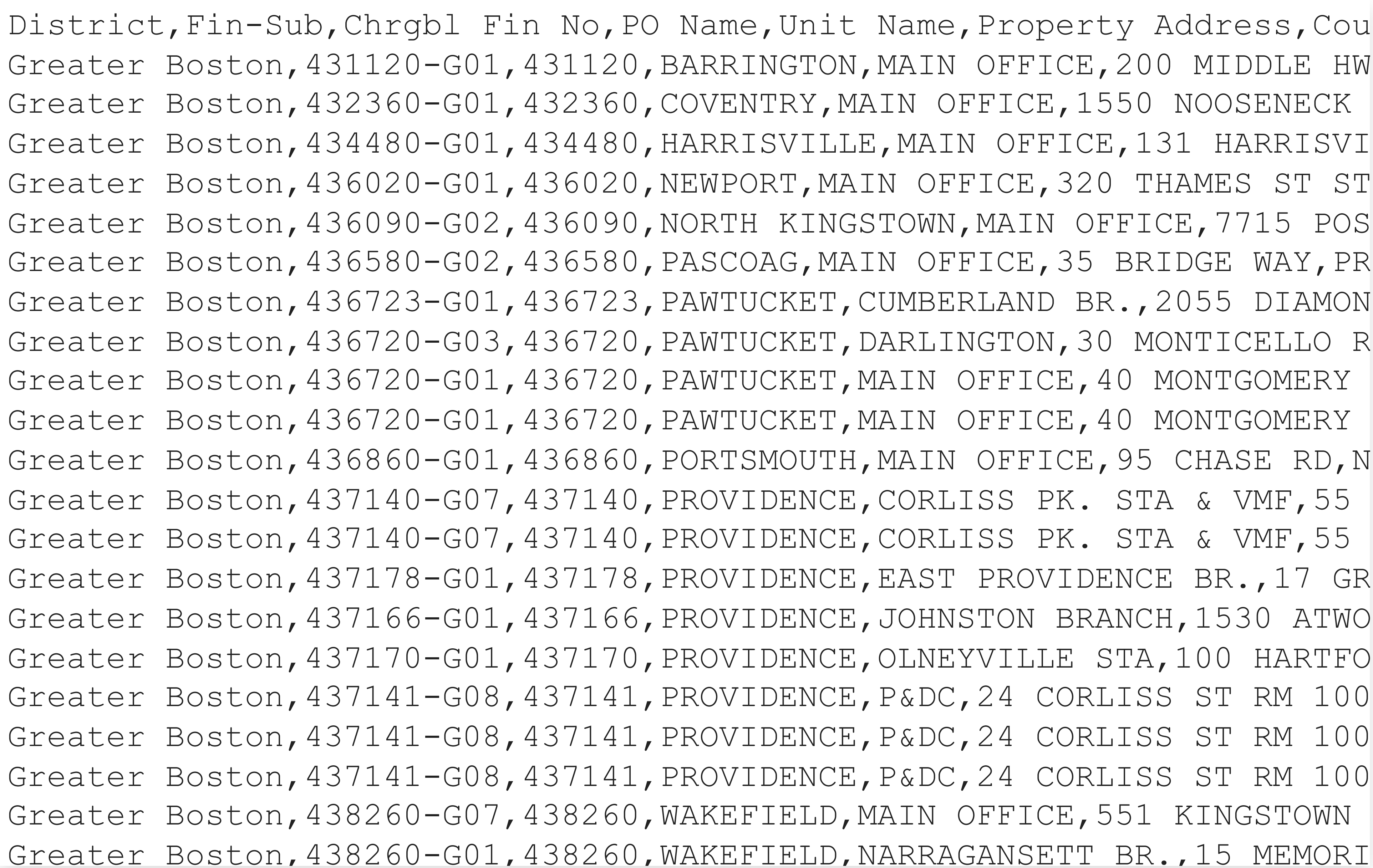
- `csv` for Comma Separated Value files
- `json` for Javascript Object Notation files

CSV

Text-based format for tabular data. Fundamentally based on rows and columns.

Used for exchanging data with spreadsheet and database programs.

Untyped. Up to reader to figure out string/float/int/etc.



```
District,Fin-Sub,Chrgbl Fin No,PO Name,Unit Name,Property Address,Cou
Greater Boston,431120-G01,431120,BARRINGTON,MAIN OFFICE,200 MIDDLE HW
Greater Boston,432360-G01,432360,COVENTRY,MAIN OFFICE,1550 NOOSENECK
Greater Boston,434480-G01,434480,HARRISVILLE,MAIN OFFICE,131 HARRISVI
Greater Boston,436020-G01,436020,NEWPORT,MAIN OFFICE,320 THAMES ST ST
Greater Boston,436090-G02,436090,NORTH KINGSTOWN,MAIN OFFICE,7715 POS
Greater Boston,436580-G02,436580,PASCOAG,MAIN OFFICE,35 BRIDGE WAY,PR
Greater Boston,436723-G01,436723,PAWTUCKET,CUMBERLAND BR.,2055 DIAMON
Greater Boston,436720-G03,436720,PAWTUCKET,DARLINGTON,30 MONTICELLO R
Greater Boston,436720-G01,436720,PAWTUCKET,MAIN OFFICE,40 MONTGOMERY
Greater Boston,436720-G01,436720,PAWTUCKET,MAIN OFFICE,40 MONTGOMERY
Greater Boston,436860-G01,436860,PORTSMOUTH,MAIN OFFICE,95 CHASE RD,N
Greater Boston,437140-G07,437140,PROVIDENCE,CORLISS PK. STA & VMF,55
Greater Boston,437140-G07,437140,PROVIDENCE,CORLISS PK. STA & VMF,55
Greater Boston,437178-G01,437178,PROVIDENCE,EAST PROVIDENCE BR.,17 GR
Greater Boston,437166-G01,437166,PROVIDENCE,JOHNSTON BRANCH,1530 ATWO
Greater Boston,437170-G01,437170,PROVIDENCE,OLNEYVILLE STA,100 HARTFO
Greater Boston,437141-G08,437141,PROVIDENCE,P&DC,24 CORLISS ST RM 100
Greater Boston,437141-G08,437141,PROVIDENCE,P&DC,24 CORLISS ST RM 100
Greater Boston,437141-G08,437141,PROVIDENCE,P&DC,24 CORLISS ST RM 100
Greater Boston,438260-G07,438260,WAKEFIELD,MAIN OFFICE,551 KINGSTOWN
Greater Boston,438260-G01,438260,WAKEFIELD,NARRAGANSETT BR.,15 MEMORI
```

Source: USPS

KEY CSV FEATURES

- May or may not have header row
- Quotes used around field values that may contain commas.

READING CSV

```
rdr = csv.reader(fobj)
for row in rdr:  # reader objects are iterable
    print("First column of this row:",row[0])
    print("Second column of this row:",row[1])
```

Note: Should always pass `newline=""` to `open()` when opening to read/write CSV.

READING CSV

```
rdr = csv.DictReader(fobj) # file MUST have header row
for row in rdr:  # rows will be dicts
    print(row["name"])
    print(row["project2_score"])
```

Note: Should always pass `newline=""` to `open()` when opening to read/write CSV.

WRITING CSV

```
w = csv.writer(fobj)
# Write a header row
w.writerow(["course","instructor"])
# Write data rows
w.writerow(["MCS 260","Dumas"])
w.writerow(["MCS 275","Dumas"])
```

Disadvantage: Easy to get the order of columns wrong, or make index mistakes.

WRITING CSV

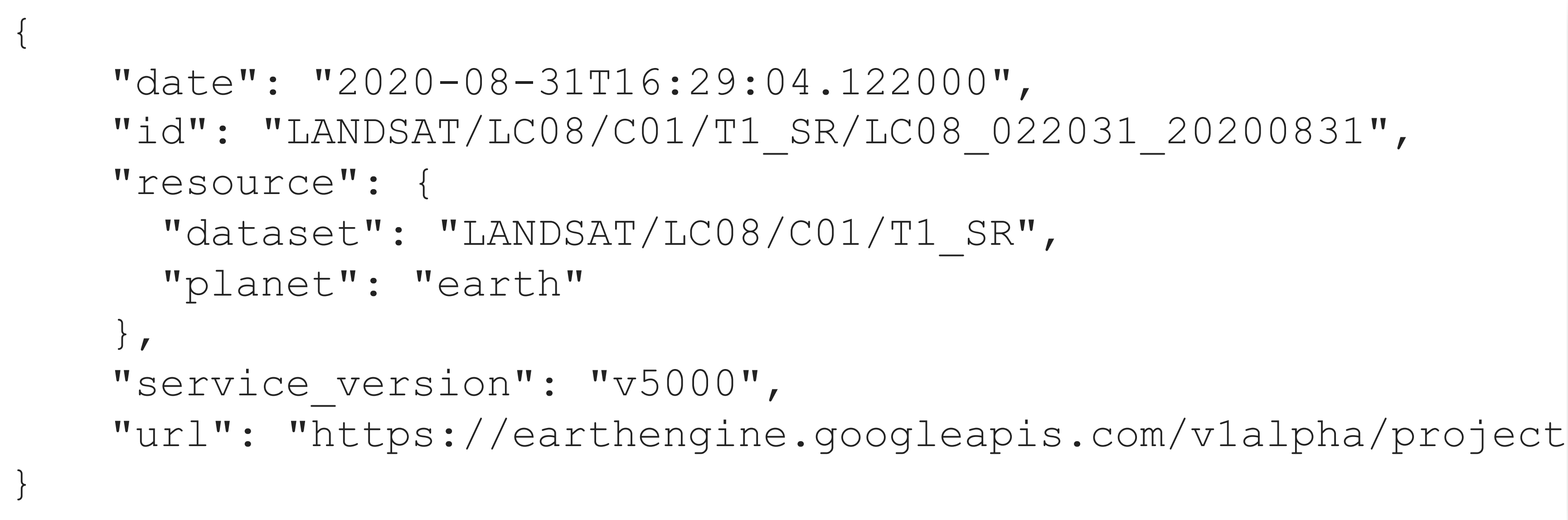
```
# Set the column order
w = csv.DictWriter(fobj, fieldnames=["course","instructor"])
# Write the header row
w.writeheader()
# Write data rows
w.writerow({"instructor":"Dumas","course":"MATH 445"})
w.writerow({"course":"MCS 481"})
```

More verbose code, but easier to read and maintain. Data order need not match column order. Missing keys handled gracefully.

JSON

JSON stands for **JavaScript object notation**. It is a text-based format for hierarchical data. Has types:

- **string** — must use double quotes.
- **number** — float, int, other? Up to reader.
- **boolean** — lower case names `true`, `false`.
- **null** — like Python `None`.
- **array** — like Python `list`. Brackets and commas.
- **object** — like Python `dict`. Curly braces, colons, and commas. Keys must be strings.



```
{
    "date": "2020-08-31T16:29:04.122000",
    "id": "LANDSAT/LC08/C01/T1_SR/LC08_022031_20200831",
    "resource": {
      "dataset": "LANDSAT/LC08/C01/T1_SR",
      "planet": "earth"
    },
    "service_version": "v5000",
    "url": "https://earthengine.googleapis.com/v1alpha/project
}
```

Source: NASA

KEY JSON FEATURES

- Does not require data to be tabular.
- Has excellent standardization and cross-language support.
- Most HTTP APIs (e.g. data portals) return JSON.
- Semi-readable for humans.

READING JSON

```
val = json.load(fobj) # read from file
val = json.loads(s)   # read from string
```

The object returned can be hard to use if you don't have documentation for the layout of the file. But since it has keys and values, it is at least explorable.

WRITING JSON

```
val = { "date": "yesterday",
        "primes": [2,3,5,7,11],
        "awesome": True
      }
json.dump(val,fobj) # save exactly one object to file
s = json.dumps(val) # make JSON string
```

Conversion table for Python → JSON

- `dict` → `object`
- `list` or `tuple` → `array`
- `int` or `float` → `number`
- `bool` → `boolean`
- `None` → `null`

REFERENCES

- MCS 260 Fall 2020:
  - Lecture 30: CSV
  - Lecture 31: JSON
- `csv` module documentation
- `json` module documentation
- Awesome JSON data sets

REVISION HISTORY

- 2021-03-11 Notebook link
- 2021-03-05 Initial publication