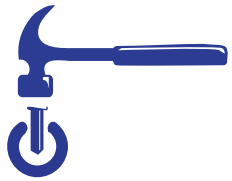
Software skills for librarians:

Library carpentry

Module 4: Additional topics

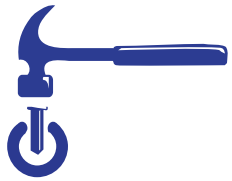


Location of OpenRefine projects

- On Linux: `~/.local/share/openrefine/`
- On MacOS: `~/Library/Application Support/OpenRefine/`
- On Windows:
  `C:\Documents and Settings\user\Local Settings\Application Data\OpenRefine`

or `C:\Users\user\AppData\Roaming\OpenRefine`

or `C:\Users\user\OpenRefine`



Transposing data in OpenRefine

- Open refine has a transpose option

  But it doesn't work as you would expect

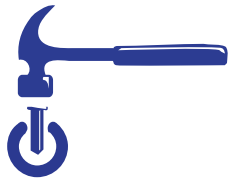
- Converts records in rows

  To a pair of key, value columns

- Select 'Transpose' from column heading

  and select start and finish columns

- The 'Columnize' option performs the reverse

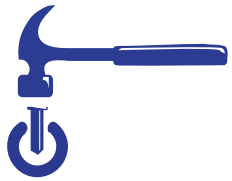


Looking up online data

- Requires a suitable online API
- And a key in your own data
- Select ‘Add column by fetching URLs’

  Enter GREL expression to format URL
- Adds raw data into new column
- Select ‘Add column based on this column’

  Extract fields using `value.parseJson().fieldname`

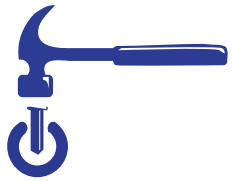


Example: Read data from Voyager

- Interface documentation: http://www.lib.cam.ac.uk/api
- Import data with Voyager bib ids
- Select 'Add column by fetching URLs'

  Enter:
  ```
  "http://www.lib.cam.ac.uk/api/voyager/bibData.cgi?bib_id="
  +value+"&database=depfacaedb&format=json"
  ```
- Extract titles:
- Select 'Add column based on this column'

  Enter: `value.parseJson().bib_record.title`



Reading CSV in Python

- Use Python's CSV module in the standard library

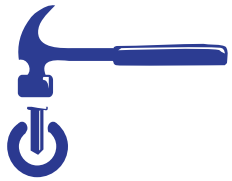
```
import csv
```

- Open a binary file and create a CSV reader object:

```
with open("file.csv", "rb") as infile:
    mycsv=csv.reader(infile)
```

- Use an iterator to read rows as a list:

```
for row in mycsv:
    print ", ".join(row)
```



Writing CSV in Python

- Open a binary file and create a CSV writer object:

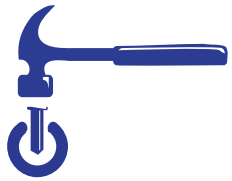
```
with open("file.csv", "wb") as outfile:
    newcsv=csv.writer(outfile)
```

- Methods for writing a single row:

```
newcsv.writerow(list)
```

- Or a list of rows:

```
newcsv.writerows(list_rows)
```



Using dictionaries

- Can use dictionaries instead of lists:

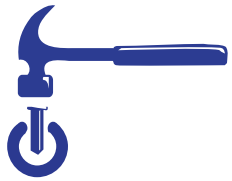
```
csv.DictReader(file, fieldnames=list)
```

- The fieldnames are required when writing:

```
csv.DictWriter(file, fieldnames=list)
```

- Optional parameters control format:

```
delimiter=":"
quotechar="/"
quoting=csv.QUOTE_NONNUMERIC
```



Using MARC in Python

- Install the Pymarc library

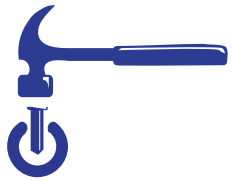
```
pip install pymarc
```

- Create a MARC reader object:

```
with open("cat.marc", "rb") as fh:
    marcfile=MARCReader(fh)
```

- Use an interator to read records:

```
for record in marcfile:
    print(record.title())
```



Operations on records

- Use subscripts to access a single field:

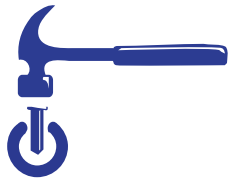
```
field=record['245']
```

- Add or remove fields:

```
record.add_field(field)
record.remove_field(tag)
```

- Writing a record:

```
outfile.write(record.as_marc())
```



Operations on fields

- Use subscripts to access a subfield:

```
title=record['245']['a']
```

- Add or remove a subfield

```
field.add_subfield(code, string)
field.delete_subfield(code)
```

- Return entire field as a string:

```
field.value()
```