Lecture: Reading CSV and JSON into Python
Learning objectives

In this lecture we will...
- Demonstrate the main methods to read CSV/TSV and JSON files in Python
- **Understand** some of the edge cases that make reading these formats difficult
In this lecture we'll look through a few functions to read CSV/TSV and JSON data in Python:

- `string.split()`
- `csv.reader` (library)
- `eval()` and `ast.eval()`
- `json.loads` (library)
Code: String.split()

- Converts a string to a list, given a separator
- By default, any whitespace separator is used (tab, space, newline)
- But different separators can be provided via an optional argument

```python
In [1]: x = "marketplace customer_id review_id product_id product_parent"

In [2]: x.split()
Out[2]: ['marketplace', 'customer_id', 'review_id', 'product_id', 'product_parent']

In [3]: x = "marketplace; customer_id; review_id; product_id; product_parent"

In [4]: x.split(';;')
Out[4]: ['marketplace', 'customer_id', 'review_id', 'product_id', 'product_parent']
```

**Note:** preserves whitespace!
What happens when the delimiter appears in the column?

In [1]: x = '4.0, "good product, would buy again"

In [2]: x.split(',

Out[2]: ['4.0', ' "good product", ' would buy again"']

Note: splits into three columns rather than two!

- This could be addressed by using a different delimiter (e.g. ';'), though this doesn't generalize for fields containing arbitrary text
- Normally, the field will be escaped by quotes
**Code: CSV.reader**

```python
In [1]: import csv

In [2]: path = "datasets/amazon/amazon_reviews_us_Gift_Card_v1_00.tsv"

In [3]: f = open(path)

In [4]: reader = csv.reader(f, delimiter = '\t')

In [5]: next(reader)
```

**Note:** specify what delimiter to use (tab)

**Out[5]:**
```
['marketplace',
 'customer_id',
 'review_id',
 'product_id',
 'product_parent',
 'product_title',
 'product_category',
 'star_rating',
 'helpful_votes',
 'total_votes',
 'vine',
 'verified_purchase',
 'review_headline',
 'review_body',
 'review_date']
```

**first line is the header**
next line is the first review in the dataset
Reading json files is even easier as they're very similar to Python's built-in dictionaries:

Code: `eval()`

```
In [1]: path = "datasets/yelp_data/review.json"

In [2]: f = open(path)

In [3]: line = f.readline()

In [4]: line
```

```
Out[4]: '{"review_id":"y6i_UHJM0_hPBq9bxWvW4w","user_id":"by2nCi50v5vroFiqKGopiW","business_id":"0W41kclzZThpx3V65bVgig","stars":5,"date":"2016-05-28","text":"Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours. \n\nThey ask you how you want you meat, lean or something maybe, I can't remember. Just say you don't want it too fatty. \n\nGet a half sour pickle and a hot pepper. Hand cut french fries too."","useful":0,"funny":0,"cool":0}
```

**Note**: first line of Yelp's review data
Code: `eval()`

Reading json files is even easier as they're very similar to Python's built-in dictionaries:

```python
In [5]: d = eval(line)

In [6]: d

Out[6]: {'business_id': '0W4lkclzzZThpx3V65bVgig',
         'cool': 0,
         'date': '2016-05-28',
         'funny': 0,
         'review_id': 'vOi_UHJMo_hPBq9bxWvW4w',
         'stars': 5,
         'text': "Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours. \n\nThey ask you how you want you meat, lean or something maybe, I can't remember. Just say you don't want it too fatty. \n\nGet a half sour pickle and a hot pepper. Hand cut french fries too.",
         'useful': 0,
         'user_id': 'bv2nC15Qv5vroF1qKGopiw'}

In [7]: d['user_id']

Out[7]: 'bv2nC15Qv5vroF1qKGopiw'
```
Code: `eval()`

Note that the "eval" function just treats an arbitrary string as if it were python code:

```
In [1]: eval("4 + 2")
Out[1]: 6
```

- While convenient, this could be **dangerous** to run on untrusted datasets since it could execute arbitrary code
- We can use some library functions to make sure that only valid json data gets executed
- We'll look at the **ast** (abstract syntax tree) and **json** libraries
Note that the outputs are identical, the code is merely "safer" to execute.
<table>
<thead>
<tr>
<th>In [6]:</th>
<th><code>import json</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>In [7]:</td>
<td><code>json.loads(line)</code></td>
</tr>
</tbody>
</table>
| Out[7]: | ```json
{'business_id': '0W4lkclz2Thpx3V65bVgig',
 'cool': 0,
 'date': '2016-05-28',
 'funny': 0,
 'review_id': 'v0i_UHJNO_hPBq9bxWvW4w',
 'stars': 5,
 'text': "Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours. \n\n
They ask you how you want your meat, lean or something maybe, I can't remember. Just say you don't want it too fatty. \n\nGet a half sour pickle and a hot pepper. Hand cut french fries too.\n
'user_id': 'bv2nCI5Qv5vRoF1qKGopiw'}
``` |

- Note that the outputs are identical, the code is merely "safer" to execute.
Summary of concepts

- Understand the **methods** `.split()` and `eval()`
- Understand the **libraries** `ast` and `json`
- Be able to read JSON and CSV data in Python

On your own...

- Try reading the Amazon dataset (or the first few lines) using `csv.reader`
- Try reading the Yelp dataset using `json.loads()`