Lesson 11: Pickling Objects, Working with Formatted Data

Fundamentals of Text Processing for Linguists

Na-Rae Han

Objectives

- Reading in formatted data to build Python data objects
  - CMU Pronouncing Dictionary
- Pickling and unpickling data objects

The CMU Pronouncing Dictionary

- http://www.speech.cs.cmu.edu/cgi-bin/cmudict
  - Check out the phoneme set – not IPA, but ASCII alternative!

- File format:

```
CONDUCTOR K AH0 N D AH1 K T ER0
CONDUCTORS K AH0 N D AH1 K T ER0 Z
CONDUCTS K AH0 N D AH1 K T S
CONDUIT K AA1 N D UW0 IH0 T
CONDUIT(1) K AA1 N JH UW0 IH0 T
CONDUIT(2) K AA1 N D W IH0 T
CONDUITS K AA1 N D UW0 AH0 T S
CONDUITS(1) K AA1 N D W AH0 T S
CONE K OW1 N
CONE'S K OW1 N Z
CONEFLOWER K OW1 N F L AW2 ER0
```

What kind of Python object?

```
CONDUCTS K AH0 N D AH1 K T S
CONDUIT K AA1 N D UW0 IH0 T
CONDUIT(1) K AA1 N JH UW0 IH0 T
CONDUIT(2) K AA1 N D W IH0 T
```

- A list of strings?

```
>>> cmudict[0]
'CONDUCTS K AH0 N D AH1 K T S '
```

- A dictionary: string → string?

```
>>> cmudict['CONDUIT']
'K AA1 N D UW0 IH0 T'
```

- A dictionary: string → list of strings?

```
>>> cmudict['CONDUIT']
['K', 'AA1', 'N', 'D', 'UW0', 'IH0', 'T']
```

  - Nope. There are multiple pronunciations.

What kind of Python object?

```
CONDUCTS K AH0 N D AH1 K T S
CONDUIT K AA1 N D UW0 IH0 T
CONDUIT(1) K AA1 N JH UW0 IH0 T
CONDUIT(2) K AA1 N D W IH0 T
```

- Answer: The dictionary value should be

  a list of lists of strings.

```
>>> cmudict['CONDUIT']
[['K', 'AA1', 'N', 'D', 'UW0', 'IH0', 'T'],
['K', 'AA1', 'N', 'JH', 'UW0', 'IH0', 'T'],
['K', 'AA1', 'N', 'D', 'W', 'IH0', 'T']]
>>> cmudict['CONDUIT'][0]
['K', 'AA1', 'N', 'D', 'UW0', 'IH0', 'T']
>>> cmudict['CONDUCTS']
[['K', 'AH0', 'N', 'D', 'AH1', 'K', 'T', 'S']]
```

Try it out

- Download this "snippet" file, process it to build cmudict:
  - http://www.pitt.edu/~naraehan/ling1901/cmudict.snip.txt

```
>>> cmudict['CONDUIT']
[['K', 'AA1', 'N', 'D', 'UW0', 'IH0', 'T'],
['K', 'AA1', 'N', 'JH', 'UW0', 'IH0', 'T'],
['K', 'AA1', 'N', 'D', 'W', 'IH0', 'T']]
>>> cmudict['CONDUCTS']
[['K', 'AH0', 'N', 'D', 'AH1', 'K', 'T', 'S']]
```

- When you are ready, process the real file:
  - http://www.speech.cs.cmu.edu/cgi-bin/cmudict
    - Follow "Download" link, and download "`cmudict.0.7a`"

Building cmudict

10 minutes

- Step 1: read in the file as a *list of lines*

```
>>> f = open('cmudict.snip.txt')
>>> lines = f.readlines()
>>> f.close()
>>> len(lines)
30
>>> lines[3]
'CONDUCT  K AH0 N D AH1 K T\n'
>>> lines[4]
'CONDUCT(1)  K AA1 N D AH0 K T\n'
```

Building cmudict

10 minutes

- Step 2: figure out how to extract from each line:
  - The keyword
  - The pronunciation as a list of phonemes
  - The keyword without "(1)", "(2)", etc.

```
>>> dat = lines[4].split()
>>> dat
['CONDUCT(1)', 'K', 'AA1', 'N', 'D', 'AH0', 'K', 'T']
>>> dat[0]
'CONDUCT(1)'
>>> dat[1:]
['K', 'AA1', 'N', 'D', 'AH0', 'K', 'T']
>>> dat[0][:-3]
'CONDUCT'
```

Assumes single digit!
To be extra careful, use
`dat[0][:dat[0].index('(')]`
instead

Building cmudict

10 minutes

- Step 3: initiate empty cmudict and populate it by going through each line

```
>>> cmudict = {}
>>> for l in lines:
        dat = l.split()
        wd = dat[0]
        pron = dat[1:]
        if wd.endswith(')'):
                wd = wd[:-3]
                cmudict[wd].append(pron)
        else:
                cmudict[wd] = [pron]

>>>
```

Building cmudict

10 minutes

- Step 4: verify cmudict

```
>>> cmudict
{'CONDUCTED': [['K', 'AH0', 'N', 'D', 'AH1', 'K', 'T',
'AH0', 'D']], 'CONDUCTORS': [['K', 'AH0', 'N', 'D',
'AH1', 'K', 'T', 'ER0', 'Z']], 'CONE': [['K', 'OW1',
'N']], 'CONFABULATION': [['K', 'AH0', 'N', 'F', 'AE2',
'B', 'Y', 'AH0', 'L', 'EY1', 'SH', 'AH0', 'N']],
...
'CONERY': [['K', 'OW1', 'N', 'ER0', 'IY0']]}
>>> cmudict['CONDUIT']
[['K', 'AA1', 'N', 'D', 'UW0', 'IH0', 'T'], ['K',
'AA1', 'N', 'JH', 'UW0', 'IH0', 'T'], ['K', 'AA1',
'N', 'D', 'W', 'IH0', 'T']]
>>> cmudict['CONDUCTS']
[['K', 'AH0', 'N', 'D', 'AH1', 'K', 'T', 'S']]
```

Building cmudict

10 minutes

- Step 5: process the real, huge, file.
  - http://www.speech.cs.cmu.edu/cgi-bin/cmudict
    - Follow "Download" link, and download "`cmudict.0.7a`"
  - Once you have read in the lines, get rid of the first 54 "comment" lines using slice indexing.
    - Alternatively, since these lines all start with ";;;", you can build in the following if condition so they are passed over:

```
>>> for l in lines:
        if l.startswith(';;;') : continue
        dat = l.split()
        wd = dat[0]
```

  - Don't forget to re-initialize cmudict, so you start from an empty dictionary!

Exploring the pronouncing dictionary

- How many words are there?

```
>>> len(cmudict)
123698
```

- How is the word 'anxious' pronounced?

```
>>> cmudict['ANXIOUS']
[['AE1', 'NG', 'K', 'SH', 'AH0', 'S'], ['AE1', 'NG', 'SH', 'AH0',
'S']]
>>> cmudict['ANXIOUS'][0]
['AE1', 'NG', 'K', 'SH', 'AH0', 'S']
>>> cmudict['ANXIOUS'][1]
['AE1', 'NG', 'SH', 'AH0', 'S']
>>> cmudict['ANXIOUS'][0][0]
'AE1'
>>> cmudict['ANXIOUS'][0][-4:]
['K', 'SH', 'AH0', 'S']
>>> 'NG' in cmudict['ANXIOUS'][0]
True
>>> 'N' in cmudict['ANXIOUS'][0]
False
```

Exploring the pronouncing dictionary

- How many words have multiple pronunciations?

```
>>> mul = [w for w in cmudict if len(cmudict[w]) > 1]
>>> len(mul)
8895
>>> mul[100:104]
['ALLISON', 'DOBRYNIN', 'QUADRUPLING', 'ENTERTAINING']
>>> cmudict['ENTERTAINING']
[['EH2', 'N', 'T', 'ER0', 'T', 'EY1', 'N', 'IH0', 'NG'], ['EH2',
'N', 'ER0', 'T', 'EY1', 'N', 'IH0', 'NG']]
```

- What's the word with the most possible pronunciations?

```
>>> max([len(cmudict[w]) for w in cmudict])
4
>>> four = [w for w in cmudict if len(cmudict[w]) == 4]
>>> four[:5]
['PREVENTIVE', 'REPRESENTED', 'BECAUSE', 'GRADUATES', 'WHELAN']
>>> cmudict['BECAUSE']
[['B', 'IH0', 'K', 'AO1', 'Z'], ['B', 'IH0', 'K', 'AH1', 'Z'],
['B', 'IH0', 'K', 'AA1', 'Z'], ['B', 'IH0', 'K', 'AH0', 'Z']]
```

Saving the pronouncing dictionary

- So you have built cmudict, a comprehensive pronunciation dictionary of English words:

```
>>> cmudict['ANXIOUS']
[['AE1', 'NG', 'K', 'SH', 'AH0', 'S'], ['AE1',
'NG', 'SH', 'AH0', 'S']]
>>> cmudict['ANXIETY']
[['AE0', 'NG', 'Z', 'AY1', 'AH0', 'T', 'IY0']]
>>> len(cmudict)
123698
```

- Question: How to save the *dictionary itself* so you don't have to rebuild it next time from the source text?

Pickling/unpickling

- The file writing method we learned writes out to a text file.
  - But: if you wrote out a dictionary content as a text file, then reading it back will be done as text, which then will need to be re-parsed into a dictionary object
- A Python data object can be "**pickled**" as itself, which then can be directly loaded ("**unpickled**") as such at a later point.
  - Also called: "**object serialization**"
  - Save as a dictionary object → read in as a dictionary object!

How to pickle

```
import pickle

grades = {'Bart':75, 'Lisa':98,
          'Milhouse':80, 'Nelson':65}

f = open('gradesdict.p', 'w')
pickle.dump(grades, f)
f.close()
```

`.dump()` pickle method for pickling

- The pickling functions come in its own module called `pickle`. Import it to use them.
- Pickling and unpickling involves a file operation, which means you need to open and close a file like you would other files.
- Pickling is done using the `.dump()` pickle method.
- Pickle files commonly have the extension '.p'.

Unpickling

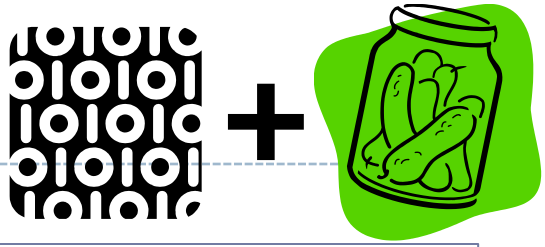
```
import pickle

f = open('gradesdict.p', 'r')
mydict = pickle.load(f)
f.close()

print mydict
# prints {'Bart':75, 'Lisa':98,
#           'Milhouse':80, 'Nelson':65}
```

`.load()` pickle method for unpickling

The same dictionary content

- Unpickling works similarly, but this time you will open the pickle file for reading, and then use the `.load()` method, assigning the pickled object to a new variable
- The unpickled object is exactly the same object you pickled, just with a (possibly) different name.

Pickling in the binary

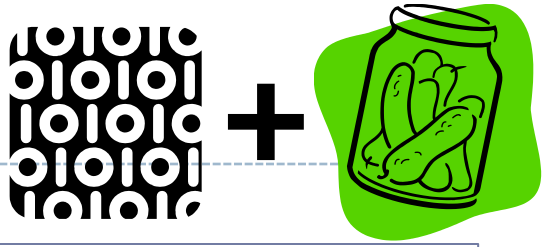


```
import pickle
grades = {'Bart':75, 'Lisa':98, 'Milhouse':80, 'Nelson':65}

f = open('gradesdict.p', 'wb')
pickle.dump(grades, f, -1)
f.close()

f = open('gradesdict.p', 'rb')
mydict = pickle.load(f)
f.close()
```

Optional "**protocol**" argument

- Optional "protocol" argument can be specified during pickle dumping.
  - -1 picks out the highest BINARY protocol available
  - Binary files are more efficient.

Pickling in the binary



```
import pickle
grades = {'Bart':75, 'Lisa':98, 'Milhouse':80, 'Nelson':65}

f = open('gradesdict.p', 'wb')
pickle.dump(grades, f, -1)
f.close()

f = open('gradesdict.p', 'rb')
mydict = pickle.load(f)
f.close()
```

File must be opened as a **binary file**, not a text file

- You are now pickling/unpickling a binary (not text) file!

➔ File object therefore has to be opened in a **BINARY mode**: `'wb'` and `'rb'` (instead of 'w' and 'r').

Try it out

3 minutes

- Pickle cmudict. Restart IDLE shell and then unpickle it.

```
>>> import pickle
>>> f = open('cmudict.p', 'wb')
>>> pickle.dump(cmudict, f, -1)
>>> f.close()

>>> ========== RESTART ==========
>>> import pickle
>>> f = open('cmudict.p', 'rb')
>>> mydict = pickle.load(f)
>>> f.close()
>>> mydict['HELLO']
[['HH', 'AH0', 'L', 'OW1'], ['HH', 'EH0', 'L', 'OW1']]
>>>
```

Wrap-up

- **Next class**
  - Working with a corpus
  - Unicode handling

- **Exercise #8**
  - **Review time!**
  - Practice Python for 1+ hour. Review what we leaned in class, try writing your own code, etc.