Coding Bootcamp Code in Python

FILES:
I/O AND DATA FORMATS

Reading from files

- Reading from text files, line by line
  - E.g., read file line by line, convert to uppercase, and print

```
with open(file_name, 'r') as text_file:
    for line in text_file:
        print(line.upper())
```

`with` …: context manager

- Reading from a binary file, value by value
  - E.g., read doubles (8 bytes) and print

```
from struct import unpack
with open(file_name, 'rb') as bin_file:
    double_bytes = bin_file.read(8)
    while double_bytes:
        print(unpack('d', double_bytes)[0])
        double_bytes = bin_file.read(8)
```

Not portable!!!:
  data type size?
  Encoding?
  little /big endian?

Libraries & data formats

- Standard library (Python 3.x)
  - Comma separated value files: `csv`
  - Configuration files: `ConfigParser`
  - Semi-structured data: `json`, `htmllib`, `sgmllib`, `xml`
- Non-standard libraries
  - Images: scikit-image
  - HDF5: pytables
  - pandas
  - Bioinformatics: Biopython

Use the "batteries" that are included!

Data formats: CSV

Let Sniffer figure out
CSV dialect (e.g., Excel)

```
from csv import Sniffer, DictReader
with open(file_name, 'rb') as csv_file:
    dialect = Sniffer().sniff(csv_file.read(1024))
    csv_file.seek(0)
    sum = 0.0
    csv_reader = DictReader(csv_file, fieldnames=None,
                                restkey='rest', restval=None,
                                dialect=dialect)
    for row in csv_reader:
        print('{name} --- {weight}'.format(name=row['name'],
                                           weight=row['weight']))
        sum += float(row['weight'])
    print('sum = {0}'.format(sum))
```

DictReader uses first
row to deduce field names

Access fields by name,
thanks to DictReader

Drawback: you still need to know field types

Writing to files

- Writing to text files
  - E.g., compute and write to file

Note: `'w'` *replaces* existing file, use `'x'` to avoid

```
with open(file_name, 'w') as text_file:
    for i in range(0, 10):
        text_file.write('{0}: {1}\n'.format(i, i*i))
```

- Append to text files
  - E.g., add some more squares to same file

```
with open(file_name, 'a') as text_file:
    for i in range(10, 20):
        text_file.write('{0}: {1}\n'.format(i, i*i))
```

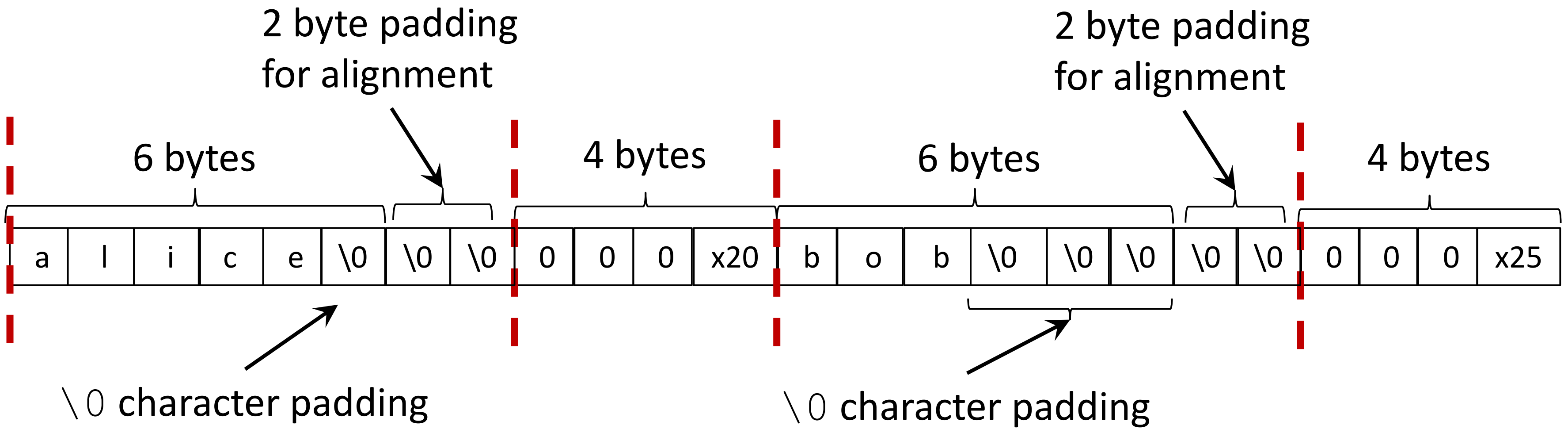
- Writing binary files: don't go there...

… unless you have to

- Writing to binary files

byte representation of name truncated to 6 characters

```
from struct import pack
with open(file_name, 'wb') as bin_file:
    for name, age in people:
        bin_file.write(pack('6si',
                            bytes(name, 'ascii'),
                            age))
```



Data formats: XML output

```
<?xml version="1.0" ?>
<blocks>
  <block name="block_01">
    <item>
      0.1
    </item>
    <item>
      1.1
    </item>
  </block>
  <block name="block_02">
    <item>
      0.2
    </item>
    <item>
      1.2
    </item>
  </block>
</blocks>
```

Data formats: creating XML

```python
from xml.dom.minidom import Document
nr_blocks = 2
nr_items = 2
doc = Document()
blocks = doc.createElement('blocks')
doc.appendChild(blocks)
for block_nr in range(1, nr_blocks + 1):
    block = doc.createElement('block')
    block_name = 'block_{0:02d}'.format(block_nr)
    block.setAttribute('name', block_name)
    blocks.appendChild(block)
    for item_nr in range(0, nr_items):
        item = doc.createElement('item')
        text = '{0}.{1}'.format(item_nr, block_nr)
        text_node = doc.createTextNode(text)
        item.appendChild(text_node)
        block.appendChild(item)
print(doc.toprettyxml(indent='  '))
```

Code Pack 09

A. Create a database with Text Files

B. Working with CSV

Coding Bootcamp Code in Python

WEB SCRAPING: GATHERING DATA FROM THE WEB

Introduction

- Caveat: web scraping code is brittle, typically not robust against
  - page layout changes (unless proper use of CSS)
  - page content changes
  - site redesign

Use site APIs (e.g., REST interface) whenever available!

- Many frameworks available, here Beautiful Soup
- However, for tables only, consider pandas

Beautiful Soup

- Open web page using `urllib`

```
import urllib
…
page = urllib.request.urlopen(page_url)
```

Note: urllib2 for Python 2.x

- Cook soup out of opened page

```
from bs4 import BeautifulSoup
…
soup = BeautifulSoup(page, "html5lib")
```

- Eat soup

```
print('looking at {0}'.format(soup.title.string))
```

Assumes page has a `title` element

Finding stuff

- First element with tag, e.g., `a`

```
print('looking at {0}'.format(soup.a))
```

- All element with tag, e.g., `a`

```
for a in soup.find_all('a'):
    print('a element: {0}'.format(a))
```

- Element content, e.g., `a`

```
for a in soup.find_all('a'):
    print('link text: {0}'.format(a.string))
```

- Element attribute, e.g., `href` in `a`

```
for a in soup.find_all('a'):
    print('link url: {0}'.format(a.get('href')))
```

Code Pack 10

A. Try the Web Scraping Code

Coding Bootcamp Code in Python

ERRORS:
DEALING WITH EXCEPTIONS

Errors

```
…
def main():
    file_name = sys.argv[1]
    with open(file_name) as in_file:
        for line in in_file:
            print('|{0}|'.format(line.rstrip('\r\n')))
    return 0
…
```

```
$ python quote.py
Traceback (most recent call last):
  File "./quote.0.py", line 13, in <module>
    status = main()
  File "./quote.py", line 6, in main
    file_name = sys.argv[1]
IndexError: list index out of range
```

exception thrown

Either check length of sys.argv, or deal with error!

Playing catch

```
…
def main():
    try:
        file_name = sys.argv[1]
    except IndexError as e:
        sys.stderr.write('### error: no input file\n')
        return 1
    with open(file_name) as in_file:
        for line in in_file:
            print('|{0}|'.format(line.rstrip('\r\n')))
    return 0
…
```

```
$ python quote.py
### error: no input file
```

More trouble

```
$ python quote.py  bla
Traceback (most recent call last):
  File "./quote.py", line 17, in <module>
    status = main()
  File "./quote.py", line 11, in main
    with open(file_name) as in_file:
IOError: [Errno 2] No such file or directory: 'bla'
```

exception
thrown

Catching more

```
…
def main():
    try:
        file_name = sys.argv[1]
        in_file = open(file_name)
        with in_file:
            for line in in_file:
                print('|{0}|'.format(line.rstrip('\r\n')))
    except IndexError as e:
        sys.stderr.write('### error: no input file\n')
        return 1
    except IOError as e:
        msg = "### I/O error on '{0}': {1}".format(e.filename,
                                                    e.strerror)
        sys.stderr.write(msg)
        return 2
    return 0
…
```

All handled!

- Now all exceptions are handled

```
$ python quote.py  bla
### I/O error on 'bla': No such file or directory
```

- Note that code size increased from 5 to 16 lines
  - Handling errors takes effort
  - Worthwhile if others are using your software!
- One can create own exceptions, derive class from `Exception`

Code Pack 11

- A. Python fundamentals:
- ~~1. Primitive Datatypes and Operators~~
- ~~2. Variables and Collections~~
- ~~3. Control Flow and Iterables~~
- ~~4. Functions~~
- ~~5. Modules~~
- ~~6. Classes~~
- 7. Advanced