LECTURE 6: AN APPLICATION: SCRAPING WEB PAGES

MODULES

- A module is simply a Python file
- The import statement e.g., import foo scans a file, executes each statement in the program
  - It constructs a dictionary for all names within the module foo
- foo.bar() at execution time invokes two run-time lookups:
  - One finds the meaning for foo in the current name space
  - One finds the data field bar in foo's name dictionary

FROM FOO IMPORT BAR

- It imports a single function, not a module
- Python first constructs the module dictionary for foo, then copy the given attribute bar from this dictionary into the caller's local dictionary
- The attribute can then be used without qualifications
- bar() invokes just one look-up, in the local scope
- Be careful with "from math import *"
  - Any name in the current scope that matches a name in the imported scope will be overridden

FROM EXAMPLES

- Local is overridden
-
```
>>> e = 42
>>> from math import *
>>> print(e)
2.71828182846
```

- A remedy
-
```
>>> e = 42
>>> from math import e as e_const
>>> print(e)
42
>>> print(e_const)
2.71828182846
```

CREATING ONE'S OWN MODULES

- A Python module is just a Python program with a list of Python statements in it, saved as a file
  - The name of the module derived from the name of the file
- Normally files that are used as modules contain only class and function definitions
- See example mycollection.py and use-it.py

MODULE VS PACKAGE

- Package is a generalization of the module concept for larger projects
- One can group a set of modules into a directory or a directory tree, then import and hierarchically refer to them using *package.subpackage.module* syntax
- To create packages, some simple special files are needed besides the modules
  - https://docs.python.org/3/tutorial/modules.html#packages

USECASE: PARSING A WEBPAGE

- Subtasks:
  - Connecting to the webpage
  - Parsing information
  - Saving to disk

URLLIB AND BEAUTIFULSOUP

- pip3 install beautifulsoup
  - Test your installation: from bs4 import BeautifulSoup
- urlllib is built-in

GET THE PAGE TEXT

```
import urllib.request
from bs4 import BeautifulSoup
response = urllib.request.urlopen("http://finance.yahoo.com")
soup = BeautifulSoup(response, “html.parser”)
```

HTML

- Hyper-text markup language
  - Convenient way to structure web pages
- CSS for formatting
  - Even more styling options
- JS for advanced interactivity & computation (JS is a complete programming language, HTML is not!)

STRUCTURE OF AN HTML FILE

Go to http://aliselmanaydin.com/cse337.html

HOW ABOUT LARGE/COMPLICATED FILES?

- Parsing HTML files with pure Python is far from trivial. Example:

```
for line in lines:
    if line[:3] == "<li>":
        line_split = line.split("<li>")[1].split("</li>")[0]
    print line_split
```

- BeautifulSoup to the help
- Search for tags:
  - soup.find_all('li')
- Get text in a particular tag:
  - soup.find_all('li')[0].get_text()

SEARCH BY CLASS NAME

- soup.find_all(‘li',class_=“…”)
- soup.find_all(id=”…")

GETTING FINANCE DATA

- Data source: https://finance.yahoo.com
- We will find out certain indices
- View-> Developer -> Developer Tools (In Chrome)
  - Other browsers have similar tools

ZEROING IN TO INDEX DATA

- "Trsdu(0.3s) Fz(s) Mb(0px) D(b)" is what we are looking for
- Let's get all relevant instances
- stock_indices= soup.find_all("span", {"class":"Trsdu(0.3s) Fz(s) Mb(0px) D(b)", "data-reactid":"7"})
- How many entries?
  - len(stock_indices)
- Let's see the content
  - print(stock_indices[0].prettify())

ZEROING IN TO INDEX DATA

```
snp= stock_indices[0]
snp_text = get_text()
snp_float = float(snp_text)
```

LET'S GET HEADLINES

```
headlines = soup.find_all("a", {"class":"Fw(b) Fz(20px) Lh(23px) LineClamp(2,46px) Fz(17px)--sm1024 Lh(19px)--sm1024 LineClamp(2,38px)--sm1024 Td(n) C(#0078ff):h C(#000)"})
```

GETTING EVERYTHING AT ONCE

Encapsulate:

```
def extract_headline(tag_set):
        return tag.get_text()
```

Call inside a loop:

```
for tag in result_set:
    headline = extract_headline(tag)
```

DUMPING INTO A FILE

```
f = open(”headlines.txt","a")
def extract_headline(tag_set):
      return tag.get_text()

for tag in result_set:
    headline = extract_headline(tag)
    f.write(headline+”\n”)
f.close()
```

FIN!