Python Applications

Workflow, data sources, requests

CS101 Lecture #12

2016-10-31

Administrivia

Administrivia

- HW5 was due on Monday 6pm
- Office hour today from 4pm-5pm
  - Faculty room opposite side of 414

Warmup questions

dict

```
d = {'one':1, 'two':2, 'three':3}

t = {}
for x in d:
    t[d[x]] = x
```

What is this piece of code doing?

Mutable

```
def fun(l, a):
    l.append(a)

x = []
y = x[:]
z = y

for i in range(10):
    fun(z,i)
```

What is the value of x in the end?

Workflow

Imperative programming

- A programming paradigm that uses statements to change a program's state
- Focuses on specifying *how* a program operates

- In contrast to *declarative programming*
  - Focuses on describing *what* the program should accomplish without specifying how
  - (SQL)

Imperative programming

- Every program (function, block, etc) tells a story
  - Beginning
  - Middle
  - End

- A good way to write a program is to make this structure explicit
- Programmer can tell what a function does by looking at name, arguments and return type
- One reason why return type is critical

Sort a List/dict/etc.

Sort a list

```
l = [‘a’, ‘c’, ‘b’, ‘d’]
l.sort()
l.sort(reverse=True)

sorted(l)
sorted(l, reverse=True)
```

Sort a dict (by key)

```
d = [‘one’:1, ‘two’:2, ‘three’:3]
sorted(d)
>>>[‘one’, ‘three’, ‘two’]

sorted(d, reverse=True)
>>>[‘two’, ‘three’, ‘one’]
```

Sort a dict by value

```
d = ['one':1, 'two':2, 'three':3]
sorted(d, key=lambda x:d[x])

>>> ['one', 'two', 'three']
```

For more information about sort(), and the lambda function:
https://wiki.python.org/moin/HowTo/Sorting

Sort a list of tuples

```
d = [‘one’:1, ‘two’:2, ‘three’:3]
t = list(d.items())
sorted(t, key=lambda x:x[1])
>>> [(‘one’,1), (‘two’,2), (‘three’,3)]

sorted(t, key=lambda x:x[0])
>>> [(‘one’,1), (‘three’,3), (‘two’,2)]
```

For more information about sort(), and the lambda function:
https://wiki.python.org/moin/HowTo/Sorting

Input Sources

Input sources

- User type in
- Hard drive
  - Plain text files
  - Comma-Separated Value files (.csv)
- The Internet!

Review: User `input`

- `input()`
  - Accepts as argument a hint message
  - Pauses for user to type in
  - Finishes when user hits ‘Return’ key
  - Returns as a `string`

Review: Files/`open`

- `Open`
  - Accept as argument a file name
  - Access mode: `'r', 'w', 'a'`
  - Returns a `file` data type

- `file` type

Review: Files/`open`

- `file`
  - A iterable for the opened file
  - `file.read()` returns a string of the entire file
  - `file.readlines()` returns a list
  - `file.close()`

Files/csv

- csv looks like spreadsheets, with columns separated by commas

```
Year, Make, Model, Price
2007, Chevrolet, Camaro, 5000.00
2010, Ford, F150, 8000.00
2011,Dodge,Grand Caravan,7500.00
```

Files/csv

- csv looks like spreadsheets, with columns separated by commas

```
Year, Make, Model, Price
2007, Chevrolet, Camaro, 5000.00
2010, Ford, F150, 8000.00
2011,Dodge,Grand Caravan,7500.00
```

- Two ways to read
  - Tokenize (split) the line into a list of items
  - Use the csv.DictReader to access components by field name

Files/csv

Year, Make, Model, Price
2007, Chevrolet, Camaro, 5000.00
2010, Ford, F150, 8000.00
2011,Dodge,Grand Caravan,7500.00

```
# Assuming filename is autos.csv
myfile = open('auto.csv')
rows = myfile.readlines()

for row in rows:
    col = row.strip().split(',')
    print(col[2], col[3])
```

Files/csv

Year, Make, Model, Price
2007, Chevrolet, Camaro, 5000.00
2010, Ford, F150, 8000.00
2011,Dodge,Grand Caravan,7500.00

```
# Assuming filename is autos.csv
from csv import DictReader
reader = DictReader(open('autos.csv'))

for row in reader:
    print(row['Model'], row['Price'])
```

Example: `plankton.csv`

- Given a field report on plankton populations, determine the largest plankton and the most common one (at any location during any season)

Internet/`request`

- `requests` is a module to access server-based resources
  - Network protocols: complex process
  - Returns a `response` data type
  - Data is in the `text` attribute

Internet/`request`

- The `text` attribute is a string of the website data

- Website data are in HTML
  - Access plain-text within the HTML data
  - Inspect the page for keywords

Internet/request

```
import requests
url = 'http://www.nws.noaa.gov/mdl/gfslamp/lavlamp.shtml'
website = requests.get(url)
offset = website.text.find('KCMI')+169
temperature_str = website.text[ offset:offset+3 ]
temperature = float(temperature_str)
```