Python Applications

workflow, data sources, `requests`

CS101 Lecture #12

2016-11-09

Administrivia

Administrivia

- Homework #6 is due Weds, Nov. 16.
- Homework #7 is due Friday, Nov. 25.

Workflow

Imperative programming

- Every program tells a story.
  - Beginning
  - Middle
  - End
- A good way to write a program is to make this explicit!
- Everything else we do in this class will follow this pattern.

Imperative programming

- This structure applies at every level.
  - expressions
  - statements
  - blocks
  - programs
- This is one reason why return type is so critical!

Input Sources

Input sources

- The user:
- The hard drive:
  - plain text files
  - comma-separated value files (`csv`)
- The Internet:

Review: User `input`

- `input`:
  - accepts as argument a message
  - *blocks* (pauses) for the user
  - `returns a string`

Review: Files/`open`

- `open`:
  - accepts as argument a file name
  - `returns a file` data type
- `file` has three useful methods:
  - `read returns a string`
  - `readlines returns a list`
  - `close`

Files/csv

- `csv` files look like spreadsheets with columns separated by commas.

```
Year,Make,Model,Price
2007,Chevrolet,Camaro,5000.00
2010,Ford,F150,8000.00
```

Example: `plankton.csv`

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

Files/csv

- `csv` files look like spreadsheets with columns separated by commas.

```
Year,Make,Model,Price
2007,Chevrolet,Camaro,5000.00
2010,Ford,F150,8000.00
```

- There are two ways to read them:
  - *tokenize* (`split`) the line into components
  - use the `csv.DictReader` tool to access components

Files/csv

```
# assuming that we have a file autos.csv
myfile = open( 'autos.csv' )
rows = myfile.readlines()
for row in rows:
    print( row[ 0 ], row[ 1 ] )
```

Files/csv

```
# assuming that we have a file autos.csv
from csv import DictReader
reader = DictReader( open( 'autos.csv' ) )
for row in reader:
    print( row[ 'Make' ], row[ 'Price' ] )
```

- So how would our `plankton.csv` example look?

Review: Internet data/`requests`

- `requests` is a module to access server-based resources
  - This is a complex process!
  - `get` `returns` a `Response` data type (but you don't need to know this)
  - The ONLY thing you need is the `text` attribute (NOT method).

Internet data/`requests`

- The `text` attribute is a `string`.
- But websites are HTML!
  - We will only access plain-text resources.
  - HTML requires *parsing*, which we won't cover.
  - Another possible approach is to inspect the page for structure.

Internet data/requests

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

Question

```
import requests
text = requests.get( 'mydataurl.com/data' )
data = ???
```

This code should produce a `list` containing the comma-separated numbers at the URL. What should replace the ??? ?

A `text.split(',')`

B `text.text.split(',')`

C `text().split(',')`

D `text.text().split(',')`

Question

```
import requests
text = requests.get( 'mydataurl.com/data' )
data = text.text.split(',')
```

This code should produce a `list` containing the comma-separated numbers at the URL. What should replace the ??? ?

A `text.split(',')`

B `text.text.split(',')` ⋆

C `text().split(',')`

D `text.text().split(',')`

Sorting a `dict` by value

```
def sortDictAsList( d ):
    items = list( d.items() )
    items.sort( key=lambda x:x[1] )
    return items
```

This is MAGIC. Don't worry AT ALL about understanding it in 101.

```
d = { 'a':2, 'b':1, 'c':-1, 'd':14 }
sortDictAsList( d )
```

Sorting a `dict` by value

Given a dictionary `d`, create a new dictionary that reverses the keys and values of `d`. Thus, the keys of `d` become the values of the new dictionary and the values of `d` become the keys of the new dictionary. You may assume `d` contains no duplicate values (that is, no two keys map to the same values). Associate the new dictionary with the variable `inverse`.

Reminders

Reminders

- Homework #6 is due Weds, Nov. 16.
- Homework #7 is due Friday, Nov. 25.
- Use the `read().split(',')` approach.