Python Applications

workflow, data sources, requests

CS101 Lecture #12
» Midterm & feedback will be discussed next week.
» Homework #6 is due Friday, Oct. 7.
» Homework #7 is due Monday, Oct. 17.
Warmup Quiz
What is the highest possible grade in CS 101 if you get a zero on the midterm?

A 90%
B 80%
C 70%
D 60%
What is the highest possible grade in CS 101 if you get a zero on the midterm?

A 90% ★ (10% of overall grade)
B 80%
C 70%
D 60%
x = 'ABBACAB'
d = {}
for c in x:
    if c not in d:
        d[c] = 0
    d[c] += 1

What is the final value of d?
A { 'A': 3, 'C': 1, 'B': 3 }
B { 'A': 3, 'B': 3, 'C': 1 }
C { 'AAA', 'BBB', 'C'}
D [ 3, 3, 1 ]
x = 'ABBACAB'
d = {}
for c in 'ABBACAB':    #c = 'A', 'B', etc.
    if c not in d:
        d[c] = 0
        d[c] += 1
c='A' => d['A']=0
c='A' => d['A']+=1
c='B' => d['B']=0
c='B' => d['B']+=1
c='B' => d['B']+=1
...
Question #1

```python
x = 'ABBACAB'
d = {}
for c in x:
    if c not in d:
        d[c] = 0
    d[c] += 1
```

What is the final value of `d`?

A `{ 'A': 3, 'C': 1, 'B': 3 }` ✫
B `{ 'A': 3, 'B': 3, 'C': 1 }` ✫
C `{ 'AAA', 'BBB', 'C'}`
D `[ 3, 3, 1 ]`

Why both?
d = {}
for n in d:
    print( n )

What does this code print?
A The values of d.
B The keys of n.
C The key–value pairs of d.
d = { 'red':1, 'green':2, 'blue':3 }
for n in d:
    print( n )

red
blue
green
d = {}
for n in d:
    print( n )

What does this code print?
A The values of d.
B The keys of n. ⭐
C The key-value pairs of d.

❖ So how do you access value corresponding to key n?  d[ n ]
myfile = open( 'odyssey.txt' )
text = myfile.read()
d = {}
for l in text.split():
    if l not in d:
        d[ l ] = 0
    d[ l ] += 1

What does this code do?
A  Counts all of the lines in 'odyssey.txt'.
B  Counts all of the words in 'odyssey.txt'.
C  Counts all of the characters in 'odyssey.txt'.
myfile = open('odyssey.txt')
text = myfile.read()
d = {}
for l in text.split():
    if l not in d:
        d[l] = 0
    d[l] += 1

What is type of text?
What is type of text.split()?  
What is type of l?
What is contents of d?
    d['Tell'] == 30
    d['tell'] == 227
myfile = open( 'odyssey.txt' )
text = myfile.read()
d = {}
for l in text.split():
    if l not in d:
        d[ l ] = 0
    d[ l ] += 1

What does this code do?
A  Counts all of the lines in 'odyssey.txt'.
B  Counts all of the words in 'odyssey.txt'.  ⋆
C  Counts all of the characters in 'odyssey.txt'.

How can we improve this? (exclude punctuation, make all lower-case)
Workflow
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.
This structure applies at every level.
- expressions
- statements
- blocks
- programs

This is one reason why return type is so critical!
Input Sources
The user:

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

- The Internet:


- **input:**
  - accepts as argument a message
  - blocks (pauses) for the user
  - returns a string
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
CSV files look like spreadsheets with columns separated by commas.

Year, Make, Model, Price
2007, Chevrolet, Camaro, 5000.00
2010, Ford, F150, 8000.00
Given a field report on plankton populations, determine the largest plankton and the most common (at any location and during any season).
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
# assuming that we have a file autos.csv
myfile = open( 'autos.csv' )
rows = myfile.readlines()
for row in rows:
    print( row[ 0 ], row[ 1 ] )
# 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?
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).
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.
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 )
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(’,,’)
import requests

```python
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(',')`
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 )
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 \( \text{inverse} \).
Homework #6 is due Friday, Oct. 7.
Homework #7 is due Monday, Oct. 17.
Use the `read().split(',,')` approach.