File IO
String formatting

Files

- Files give us PERSISTENCE
  - Data in programs is cleared with every run, not the case with files
- Text files provide convenient input/output storage
  - e.g. programs can read configuration data or input files to process, and can write output to files

Files – important terms

- File: A document
- Directory: A folder containing files and other folders
- File System: Collection of all the files and folders on the computer, organized in a hierarchy

File Input/Output

- We read data from a file into our program.
- We write data from our program into a file.
- Steps for File I/O

  1. Open the file (creates a "connection" between your program and the file).

```
f = open('animals.txt')
```

  2. Read the data / write the data

  3. Close the file (close the "connection"). This should to be done once per file.

Reading Files with Methods

- Several methods for reading text from files:
  - `readline()`: reads and returns next line; returns empty string at end-of-file
  - `read()`: reads the entire file into one string
  - `readlines()`: reads the entire file into a list of strings
- All of these leave a trailing '\n' character at the end of each line.

```
f = open('animals.txt')
line = f.readline()
print(line)
line = f.readline()
f.close()
```

Reading Files in a loop

```
f = open('animals.txt')
for line in f:
     print(line.strip())
f.close()
```

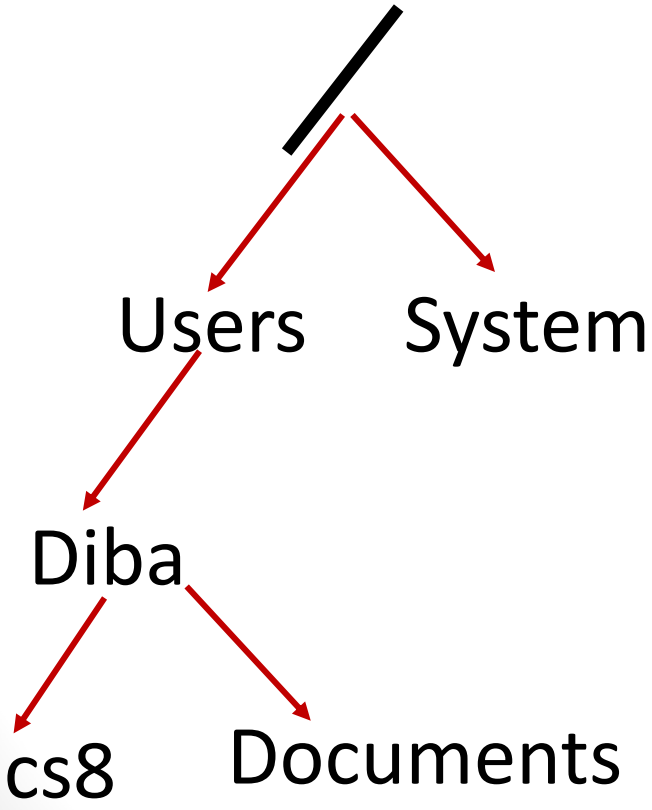
See detailed lecture notes for usage with read and readlines

Writing to file

```
outfile = open('example_2.txt', 'w')
outfile.write("Duck\nCow\nCat")
outfile.close()
```

Unix File System

- Root (/)
- Path



Concept Question

Every file on a file system can be referred to be an "absolute pathname", which consists of a sequence of … what?

A. Files
B. Directories
C. Paths

Concept Question

In contrast to an “absolute pathname”, we have the concept of a “relative pathname”. What is the technical term used for the “starting point” of a “relative pathname”?

A. Root
B. Home directory
C. Current directory
D. None of the above

Navigating the unix file system

- Some common unix commands
  - `ls`
  - `pwd`
  - `mkdir`
  - `cd`

String Methods

```
s = "CS 8: Intro to Programming”
s.find(“8”)
s.find(“Math”)
s.startswith(“CS”)
s.startswith(“Computer”)
s.endswith(“ing”)
s.endswith(“Prog”)
s.count(‘m’)
'Mississippi'.count('i’)
s.replace(":", "#")
s.upper()
‘Mississippi’.lower()
```

Concept Question

```
MS = “Mississippi”
MS.replace(“i”, “!”)
print(MS)
```

What is printed?

A. Mississippi

B. M!ss!ss!pp!

C. Error

D. None of the above

String formatting

Let's say you have an integer price:

```
price = 18.00
```

Write a statement to print:

```
The price is <price>. Wow that's cheap!
```

''' Format specification:
{ : }. Left side of colon say which argument to place into {}

To the right we specify a FIELD WIDTH (i.e., how many spaces/ columns on the screen to devote to this

```
print("-->{}<--".format(price))
print("-->{:20}<--".format(price))
# We can use '>' or '<' to justify left or right
print("-->{:<20}<--".format("18"))
print("-->{:>20}<--".format("18"))
# we can use '^' to center.
print("-->{:^20}<--".format("18"))
print("-->{:20.2f}<--".format(price))
# without 'f' , price appears in scientific notation
# width of 20, with 2 places after the decimal
```

HW

3. (10 pts) Section 4.2 discusses formatted output. What is the output of the following print statement? Please put one character per box to show the exact spacing. Try to figure it out by hand before checking your answer online. If you spoil the first grid, use the second.

```
print('{0:4},{2:6}'.format(123,456,789))
```

lab06

Your code will produce the following histogram by simulating die roll:

```
Distribution of dice rolls

 2:      7 (  2.8%)  *******
 3:     14 (  5.6%)  **************
 4:     29 ( 11.6%)  *****************************
 5:     26 ( 10.4%)  **************************
 6:     34 ( 13.6%)  **********************************
 7:     41 ( 16.4%)  *****************************************
 8:     30 ( 12.0%)  ******************************
 9:     23 (  9.2%)  ***********************
10:     23 (  9.2%)  ***********************
11:     16 (  6.4%)  ****************
12:      7 (  2.8%)  *******
----------------------------
250 rolls
```

Random numbers

```
from random import random

random()          # returns a number in the range [0,1)
randrange(x,y) # returns a random integer including x up to
                  # (but not including) y.
choice(somelist) # selects an element at random from somelist
```

Generating random numbers

Write a Python statement to generates a number between 0 and 100 (include floating point values like 55.5)

Assume you have the correct import statements

A. random()+100
B. random()*100
C. random()/100

Generating random numbers

Write a Python statement to generates a INTEGER between 50 and 100. Assume you have the correct import statements

A. random()*50
B. 50+ int(random()*50)
C. randrange(50,100)
D. Both B and C
E. None of the above

Lab06 warm up

```
def rollDice():
    '''
    returns sum of rolling two six sided die'''


def rollDistribution(n):
    '''
    rolls a pair of die n times, returns the tally'''


def printDistribution(diceTally):
    '''
    prints the diceTally as a histogram'''
```