Introduction to Computer Programming
Section 8
Lecture 24
Files
Reading and Writing Files

- Data in RAM is fleeting…
- Files preserve data for the longer term
- Files store data that can’t fit in RAM
- Files are used to pass data between apps
  - Take a picture and save it
  - Email it, post it on Twitter, Facebook
Plain Text Files

• Like a very, very long string
  – By convention, text files are broken up into lines
  – By convention, lines are separated by “\n”

• ASCII or Unicode characters

• Can be viewed with simple text editor like Notepad

• Examples:
  
  name.txt  name.py  name.html
Binary Files

• The data is encoded as numbers, or some format that only a special app understands.
• Cannot be read by simple text editors
  – If you try, you will get garbage !!

• Examples:
  name.jpg  name.doc  name.pdf  name.ppt
Random Access vs. Sequential Access

• Data in memory is accessed randomly: you can go directly into memory and get a data value

  \texttt{mylist[407]}

• Data in files is typically accessed sequentially: \textit{read, read, read, (407 times, sigh) .... Found it!}
Accessing Files in Python

• To access a file, you first must open it with the built-in open function:

  ```python
  myfile = open(filename, mode)
  ```

A file object stores a connection between your program and the operating system. The mode indicates what you want to do with the file:

  read, write, or append
Processing an Input File

1. Open the file in *Read Mode*
   ```python
   infile = open("filename", "r")
   ```

2. Read and process each line in the file in a loop
   ```python
   line = infile.readline()
   ```

3. Close the file
   ```python
   infile.close()
   ```
Creating an Output File

1. Open the file in **Write Mode**
   Note: If the file does not exist it will be created.
   Beware: If the file does exist it will be overwritten.
   
   ```python
   outfile = open(“filename”, “w”)
   ```

2. Write lines into the file in a loop
   ```python
   outfile.write(line + “\n”)
   ```

3. Close the file
   ```python
   outfile.close()
   ```
Appending to an Output File

1. Open the file in *Append Mode*
   If the file does not exist it will be created. If the file exists, simply “append” any new data to the end of the file.

   ```python
   outfile = open("filename", "a")
   ```

2. Write lines into the file in a loop
   ```python
   outfile.write(line + "\n")
   ```

3. Close the file
   ```python
   outfile.close()
   ```