



# COMP2720: Automating Tools for New Media

Files: How to read them, write them, and process them.



# Files

- Files are these named large collections of bytes.
- Files typically have a base name and a suffix.
  - “barbara.jpg” has a base name of “barbara” and a suffix of “.jpg”
- Files exist in directories (sometimes called folders).



# Directories / Folders

- Directories can contain files or other directories.
- There is a base directory on your computer, sometimes called the *root* directory.
- A complete description of what directories to visit to get to your file is called a *path*.

# Paths

```
>>> file=pickAFile()
```

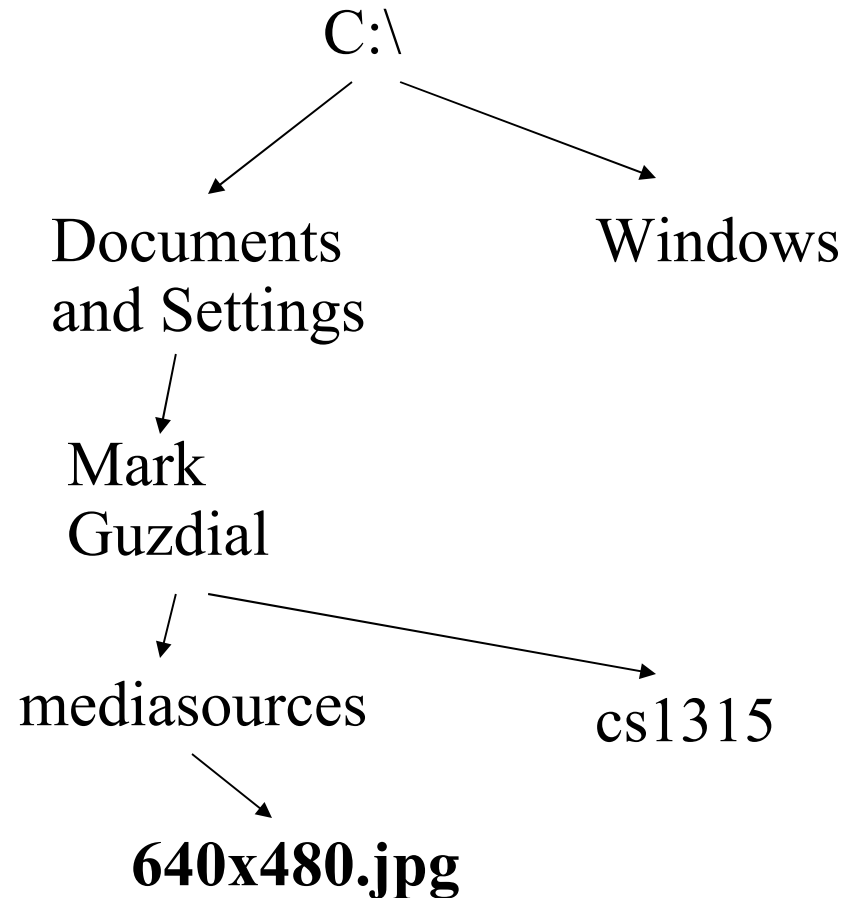
```
>>> print file
```

```
C:\Documents and Settings\Mark Guzdial\My  
Documents\mediasources\640x480.jpg
```

- This says that, on my computer, the file “640x480.jpg” resides in the “mediasources” directory, which is in “My Documents” which is in “Mark Guzdial” which is in “Documents and Settings” which is in the base directory on the drive (hard disk) “C:\”

# We call this structure a *tree*

- “C:\” is the *root* of the tree.
- It has *branches*, each of which is a directory.
- Any directory (branch) can contain more directories (branches) and files (*leaves*)



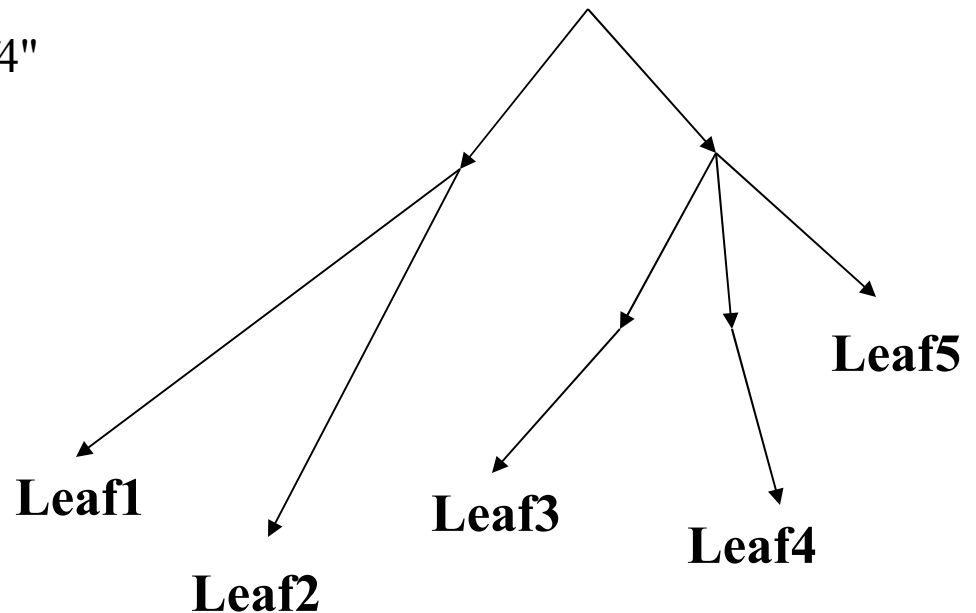


# Why do I care about all this?

- If you're going to process files, you need to know where they are (directories) and how to specify them (paths).
- If you're going to do movie or sound processing, which can involve lots of files, you need to be able to write programs that process all the files in a directory (or even several directories) without having to write down each and every name of the files.

# Using lists to represent trees

```
>>> tree =  
    [[["Leaf1", "Leaf2"], [{"Leaf3"}, {"Leaf4"},  
        ], "Leaf5"]]  
>>> print tree  
[['Leaf1', 'Leaf2'], [{"Leaf3"}, ['Leaf4'],  
    'Leaf5']]  
>>> print tree[0]  
['Leaf1', 'Leaf2']  
>>> print tree[1]  
[['Leaf3'], ['Leaf4'], 'Leaf5']  
>>> print tree[1][0]  
['Leaf3']  
>>> print tree[1][1]  
['Leaf4']  
>>> print tree[1][2]  
Leaf5
```



The Point: Lists allow us to represent complex relationships, like trees

# How to open a file in Python

- For *reading* or *writing* a file (getting characters out or putting characters in), you need to use `open()`.
- `open(filename, how)` opens the file with *filename*.
  - If you don't provide a full path, the filename is assumed to be in the same directory as JES.
- `how` is a two character string that says what you want to do with the file.
  - “rt” means “*read text*”
  - “wt” means “*write text*”
  - “rb” and “wb” means read or write bytes
    - We won't do much of that

# Methods on files: `open()` returns a file object

- `open()` returns a file object that you use to manipulate the file.
  - Example: `file=open("myfile","wt")`
- `file.read()` reads the whole file as a single string.
- `file.readlines()` reads the whole file into a list where each element is one line.
  - `read()` and `readlines()` can only be used once without closing and reopening the file.
- `file.write(something)` writes something to the file.
- `file.close()` closes the file — writes it out to the disk, and won't let you do any more to it without re-opening it.

# Reading a file

```
>>> program=pickAFile()
>>> print program
C:\Documents and Settings\Mark Guzdial\My Documents\py-programs\littlepicture.py
>>> file=open(program,"rt")
>>> contents=file.read()
>>> print contents
def littlepicture():
    canvas=makePicture(getMediaPath("640x480.jpg"))
    addText(canvas,10,50,"This is not a picture")
    addLine(canvas,10,20,300,50)
    addRectFilled(canvas,0,200,300,500,yellow)
    addRect(canvas,10,210,290,490)
    return canvas
>>> file.close()
```

# Reading a file by lines

```
>>> file=open(program,"rt")
>>> lines=file.readlines()
>>> print lines
['def littlepicture():\n', '
  canvas=makePicture(getMediaPath("640x480.jpg"))\n', '
  addText(canvas,10,50,"This is not a picture")\n', '
  addLine(canvas,10,20,300,50)\n', '
  addRectFilled(canvas,0,200,300,500,yellow)\n', '
  addRect(canvas,10,210,290,490)\n', ' return canvas']
>>> file.close()
```

# Silly example of writing a file

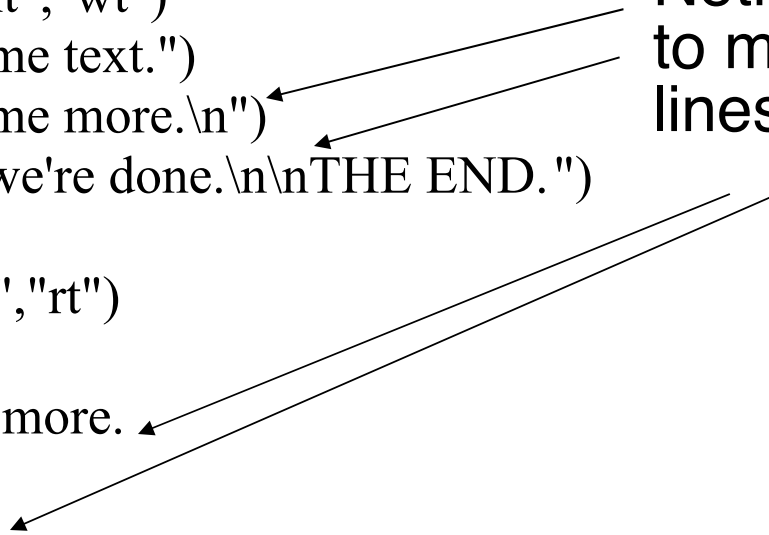
```
>>> writefile = open("myfile.txt","wt")
>>> writefile.write("Here is some text.")
>>> writefile.write("Here is some more.\n")
>>> writefile.write("And now we're done.\n\nTHE END.")
>>> writefile.close()
>>> writefile=open("myfile.txt","rt")
>>> print writefile.read()
```

```
Here is some text.Here is some more.
And now we're done.
```

```
THE END.
```

```
>>> writefile.close()
```

Notice the `\n`  
to make new  
lines





# Now let's start playing with files!

- First, a function that will automatically change the text string that the program `littlepicture.py` draws.
- As input, we'll take a new filename and a new string.
- We'll `find()` the `addText`, then look for the first double quote, and then the final double quote.
- Then we'll write out the program as a new string to a new file.

# Changing the little program automatically

```
def changeLittle(filename,newstring):
    # Get the original file contents
    programfile=r"C:\Documents and Settings\Mark Guzdial\My Documents\py-
        programs\littlePicture.py"
    file = open(programfile,"rt")
    contents = file.read()
    file.close()
    # Now, find the right place to put our new string
    addtext = contents.find("addText")
    firstquote = contents.find('"',addtext) # Double quote after addText
    endquote = contents.find('"',firstquote+1) # Double quote after firstquote
    # Make our new file
    newfile = open(filename,"wt")
    newfile.write(contents[:firstquote+1]) # Include the quote
    newfile.write(newstring)
    newfile.write(contents[endquote:])
    newfile.close()
```

# changeLittle("sample.py", "Here is a sample of changing a program")

## Original:

```
def littlePicture():  
  
    canvas=makePicture(getMediaPath("640x480.jpg"))  
    addText(canvas,10,50,"This is not a picture")  
    addLine(canvas,10,20,300,50)  
  
    addRectFilled(canvas,0,200,300,500, yellow)  
    addRect(canvas,10,210,290,490)  
    return canvas
```

## Modified:

```
def littlePicture():  
  
    canvas=makePicture(getMediaPath("640x480.jpg"))  
    addText(canvas,10,50,"Here is a sample of changing a program")  
    addLine(canvas,10,20,300,50)  
  
    addRectFilled(canvas,0,200,300,500, yellow)  
    addRect(canvas,10,210,290,490)  
    return canvas
```



# That's how vector-based drawing programs work!

- Editing a line in AutoCAD doesn't change the pixels.
- It changes the underlying representation of what the line should look like.
- It then runs the representation and creates the pixels all over again.
- Is that slower?
  - Who cares? (Refer to Moore's Law...)

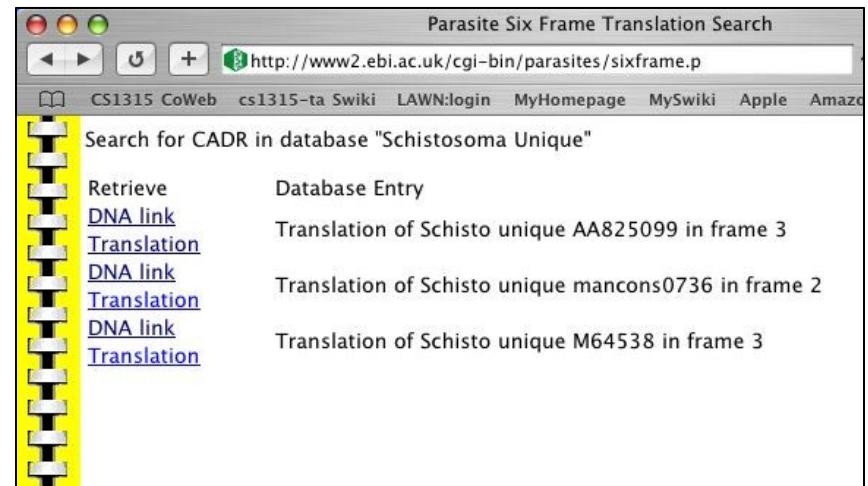


# Finding data on the Internet

- The Internet is filled with wonderful data, and almost all of it is in text!
- Later, we'll write functions that directly grab files from the Internet, turn them into strings, and pull information out of them.
- For now, let's assume that the files are on your disk, and let's process them from there.

# Example: Finding the nucleotide sequence

- There are places on the Internet where you can grab DNA sequences of things like parasites.
- What if you're a biologist and want to know if a sequence of nucleotides that you care about is in one of these parasites?
- We not only want to know "yes" or "no," but which parasite.



# What the data looks like

>Schisto unique AA825099

```
gcttagatgtcagattgagcacgatgatcgattgaccgtgagatcgacga  
gatgcgcagatcgagatctgcatacagatgatgaccatagtgtacg
```

>Schisto unique mancons0736

```
ttctcgctcacactagaagcaagacaattacactattattattatt  
accattattattattattactattattattattactattatta  
ctacgtcgcttttactccctttattctcaaattgtgtatccttcctt
```



# How are we going to do it?

- First, we get the sequences in a big string.
- Next, we find where the small subsequence is in the big string.
- From there, we need to work backwards until we find “>” which is the beginning of the line with the sequence name.
- From there, we need to work forwards to the end of the line. From “>” to the end of the line is the name of the sequence.
  - Yes, this is hard to get just right. Lots of debugging prints.

# The code that does it

```
def findSequence(seq):
    sequencesFile = getMediaPath("parasites.txt")
    file = open(sequencesFile,"rt")
    sequences = file.read()
    file.close()
    # Find the sequence
    seqloc = sequences.find(seq)
    #print "Found at:",seqloc
    if seqloc <> -1:
        # Now, find the ">" with the name of the sequence
        nameloc = sequences.rfind(">",0,seqloc)
        #print "Name at:",nameloc
        endlines = sequences.find("\n",nameloc)
        print "Found in ",sequences[nameloc:endlines]
    if seqloc == -1:
        print "Not found"
```

# Why -1?

- If `.find()` or `.rfind()` don't find something, they return -1
  - If they return 0 or more, then it's the index of where the search string is found.
- What's “<>”?
  - That's notation for “not equals”
  - You can also use “!=“

# Running the program

```
>>> findSequence("tagatgtcagattgagcacgatgatcgattgacc")
```

```
Found in >Schisto unique AA825099
```

```
>>> findSequence("agtcactgtctggttgaaagtgaatgcttccaccgatt")
```

```
Found in >Schisto unique mancons0736
```

# Example: Get the temperature

- The weather is always available on the Internet.
- Can we write a function that takes the forecast temperature out of a source like <http://www.bom.gov.au/weather/act/> ?

Forecast for the ACT - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.bom.gov.au/cgi-bin/wrap\_fwo.pl?IDN10035.txt

Forecast for the ACT phorum - comp272@talk

Home | About Us | Contacts | Help | Feedback

SEARCH

Global | Australia | NSW | Vic. | Old | WA | SA | Tas. | ACT | NT | Ant.

Learn About Meteorology | Weather & Warnings | Climate | Hydrology | Numerical Prediction | About Services | Registered User Services

See [Glossary](#) to find out what the weather terms mean.

IDN10035  
FORECAST FOR THE ACT  
Issued at 10:00am on Tuesday the 20th of September 2005  
BUREAU OF METEOROLOGY  
CANNBERRA METEOROLOGICAL OFFICE

**SITUATION:**  
A weak cold front that crossed the region yesterday is now over the Tasman Sea, and a ridge of high pressure is developing over NSW. Fine conditions with the high over the next few days, becoming warm on Wednesday and Thursday as winds turn northerly with the high moving over the Tasman Sea. On Friday and over the weekend, a trough will move into NSW, bringing some more unsettled weather, with afternoon showers and possible thunderstorms about.

**LOCAL AREA:**  
Partly cloudy this morning with some fog patches about and isolated showers in the far south. Winds were light and variable.

**WARNINGS:**  
Nil.

**A.C.T. FORECAST:** For the rest of today and Wednesday  
Fine, partly cloudy today with light to moderate west to northwest winds. Early fog patches again on Wednesday before a warm, fine and mostly sunny day with light northwesterly winds.

**PRECIS:**  
Max Today: 16  
Wednesday Min Tonight: 2 Max: 19

**UV Index:** UV Index 4 [Moderate].

**Headline :** Fine, partly cloudy.

**Probability of rain:** 20%.

**Fire Danger Ratings:**

**Winds on Canberra lakes:** Variable winds 5 to 10 km/h overnight and in the mornings. Westerly winds at 15 to 25 km/h today, tending northwesterly to 15 km/h on Wednesday.

Done

Work Idis [Calendar - KOrganizer] Inbox for peter.christen Forecast for the ACT - N Tue Sep 20 3°C 10:25 AM

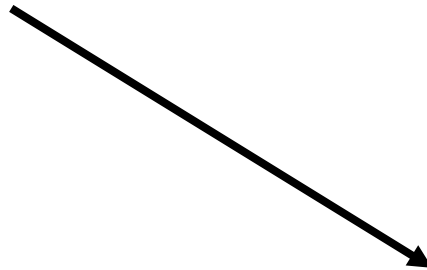


# The Internet is mostly text

- Text is the other unimedia.
- Web pages are actually text in the format called HTML (HyperText Markup Language)
  - HTML isn't a programming language, it's an encoding language.
  - It defines a set of meanings for certain characters, but one can't program in it.
- We can ignore the HTML meanings for now, and just look at patterns in the text.

# Where's the temperature?

- The word “temperature” doesn't really show up.
- But the forecast temperature follows the word “Max”



Precis: Windy, chance afternoon shower.

City: Max 17

Tuggeranong: Max 16




## We can use the same *algorithm* we've seen previously

- Grab the content out of a file in a big string.
  - (We've saved the HTML page previously.)
  - Soon, we'll see how to grab it directly.)
- Find the starting indicator (“Max ”)
- Find the ending indicator (a new line “\n”)
- Read the previous characters.

# Finding the temperature

```
def findTemperature():
    weatherFile = getMediaPath("bom_act_forecast.htm")
    file = open(weatherFile,"rt")
    weather = file.read()
    file.close()
    # Find the Temperature
    curloc = weather.find("Max ")
    if curloc <> -1:
        # Now, find the new line "\n" following the temp
        tempstart = curloc+4 # Position after "Max "
        tempend = weather.find("\n", tempstart)
        print "Current temperature:",weather[tempstart+1:tempend]
    if curloc == -1:
        print "They must have changed the page format -- can't find the temp"
```



# What to do now

- Read sections 10.1, 10.2 and 10.3 in text book.
- **Assignment 1 due next week!**