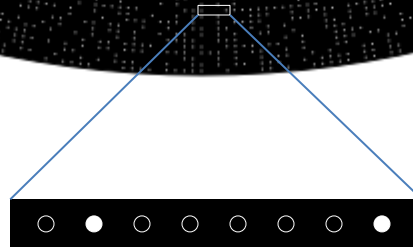
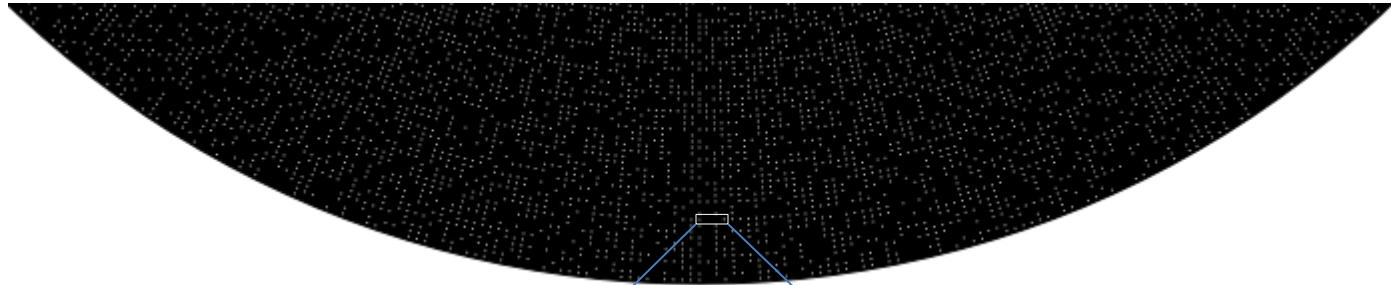


Review

- Algorithms
- Sorting
 - Selection Sort
 - Bubble Sort
- Searching
 - Linear Search
 - Binary Search
- Worst Case Running Time



0 1 0 0 0 0 0 1

$0+2^6+0+0+0+0+0+2^0$

64 + 1

65

st schoolboy is now familiar with truths for which Archimedes would have sacrificed his life.

ASCII - American Standard Code for Information Interchange

Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec
(nul)	0	(dc4)	20	(40	<	60	P	80	d	100	x	120
(soh)	1	(nak)	21)	41	=	61	Q	81	e	101	y	121
(stx)	2	(syn)	22	*	42	>	62	R	82	f	102	z	122
(etx)	3	(etb)	23	+	43	?	63	S	83	g	103	{	123
(eot)	4	(can)	24	,	44	@	64	T	84	h	104		124
(enq)	5	(em)	25	-	45	A	65	U	85	i	105	}	125
(ack)	6	(sub)	26	.	46	B	66	V	86	j	106	~	126
(bel)	7	(esc)	27	/	47	C	67	W	87	k	107	(del)	127
(bs)	8	(fs)	28	0	48	D	68	X	88	l	108		
(ht)	9	(gs)	29	1	49	E	69	Y	89	m	109		
(nl)	10	(rs)	30	2	50	F	70	Z	90	n	110		
(vt)	11	(us)	31	3	51	G	71	[91	o	111		
(np)	12	(sp)	32	4	52	H	72	\	92	p	112		
(cr)	13	!	33	5	53	I	73]	93	q	113		
(so)	14	"	34	6	54	J	74	^	94	r	114		
(si)	15	#	35	7	55	K	75	_	95	s	115		
(dle)	16	\$	36	8	56	L	76	`	96	t	116		
(dc1)	17	%	37	9	57	M	77	a	97	u	117		
(dc2)	18	&	38	:	58	N	78	b	98	v	118		
(dc3)	19	'	39	;	59	O	79	c	99	w	119		

Files

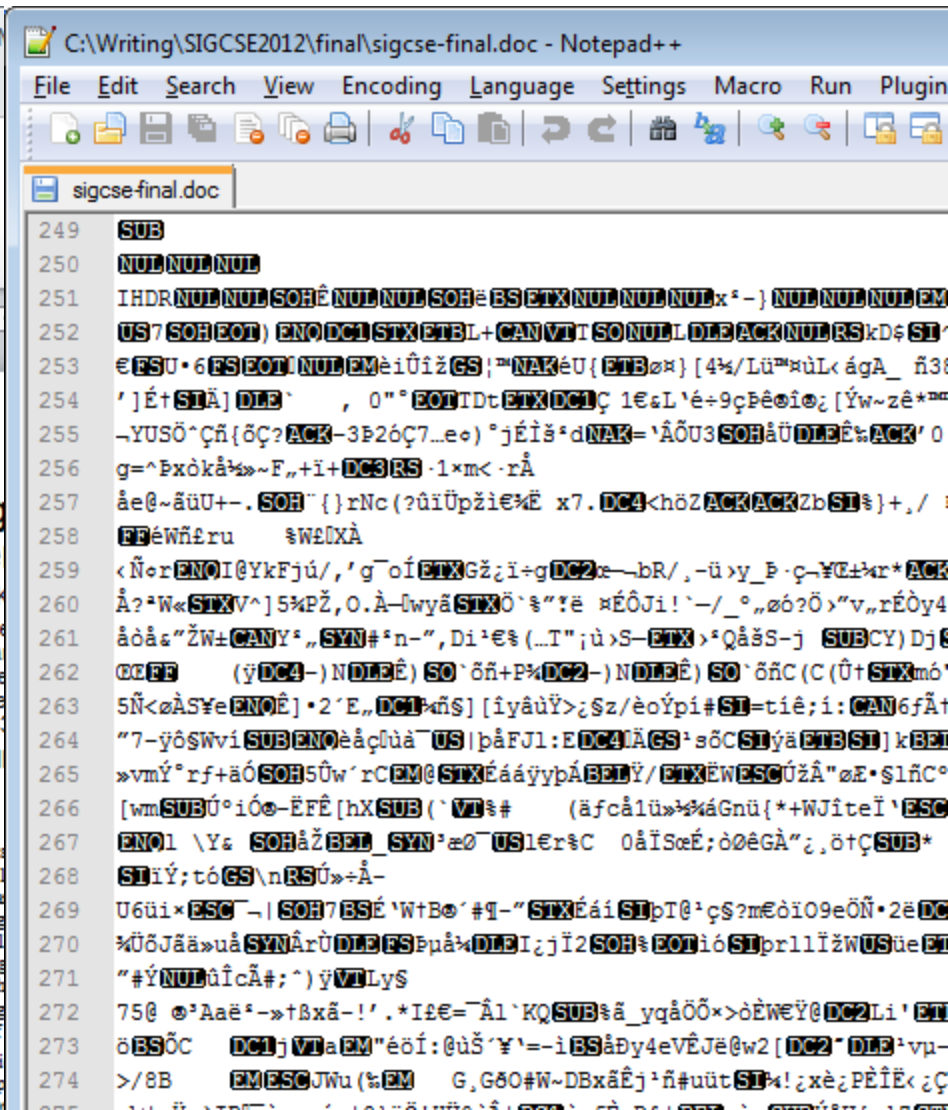
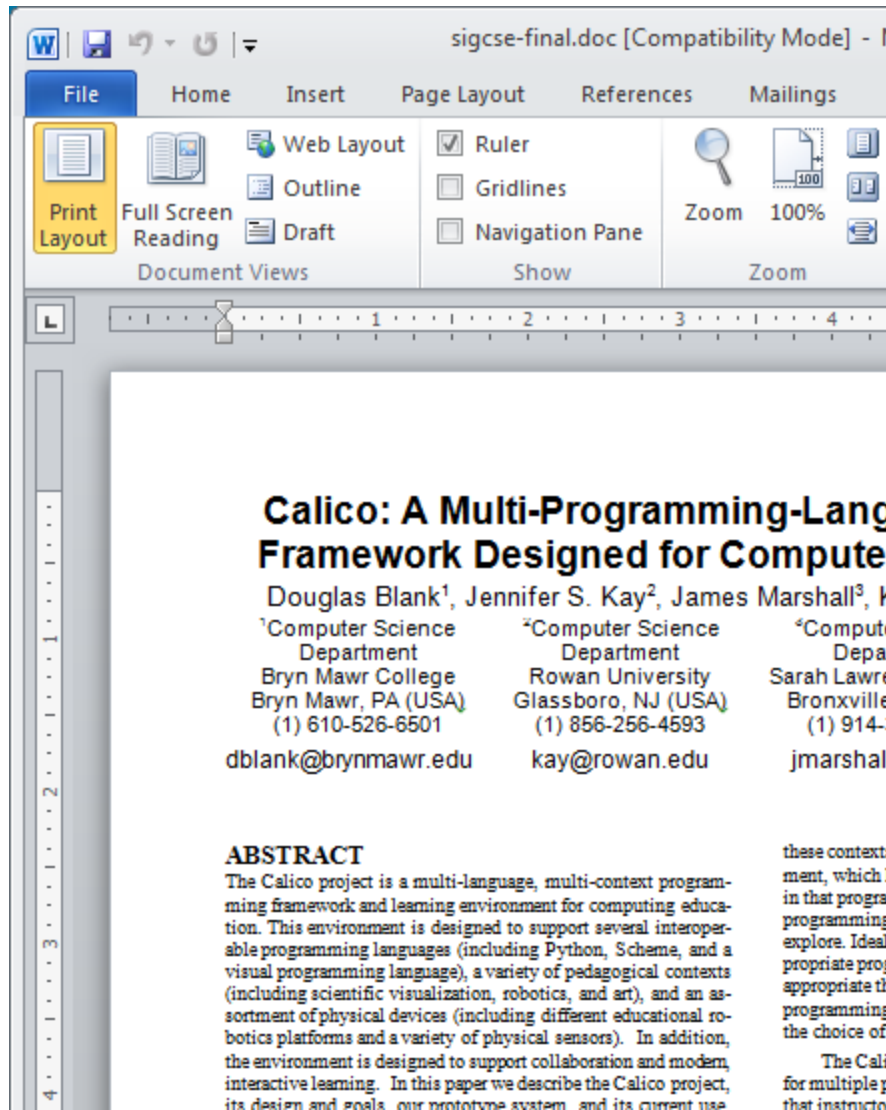
- A sequence of bytes
- Usually stored in some durable form
- Have a beginning, end, and length (size)
- Have an encoding – how to interpret data
 - US-ASCII (7-bits per glyph)
 - ISO-8859-1 (8-bits per glyph)
 - Unicode (Combines several encodings)
 - UTF-8 (variable width per glyph, >1,100,000)
 - ...
- Compression
 - Lossless vs. Lossy (ex. PNG vs. JPEG)
 - Run-length Encoding (253 white pixels, 29 reds, ...)
 - Lempel–Ziv (dictionary coders, used by zip)
 - ...



Unicode Character
'PILE OF POO'
(U+1F4A9)

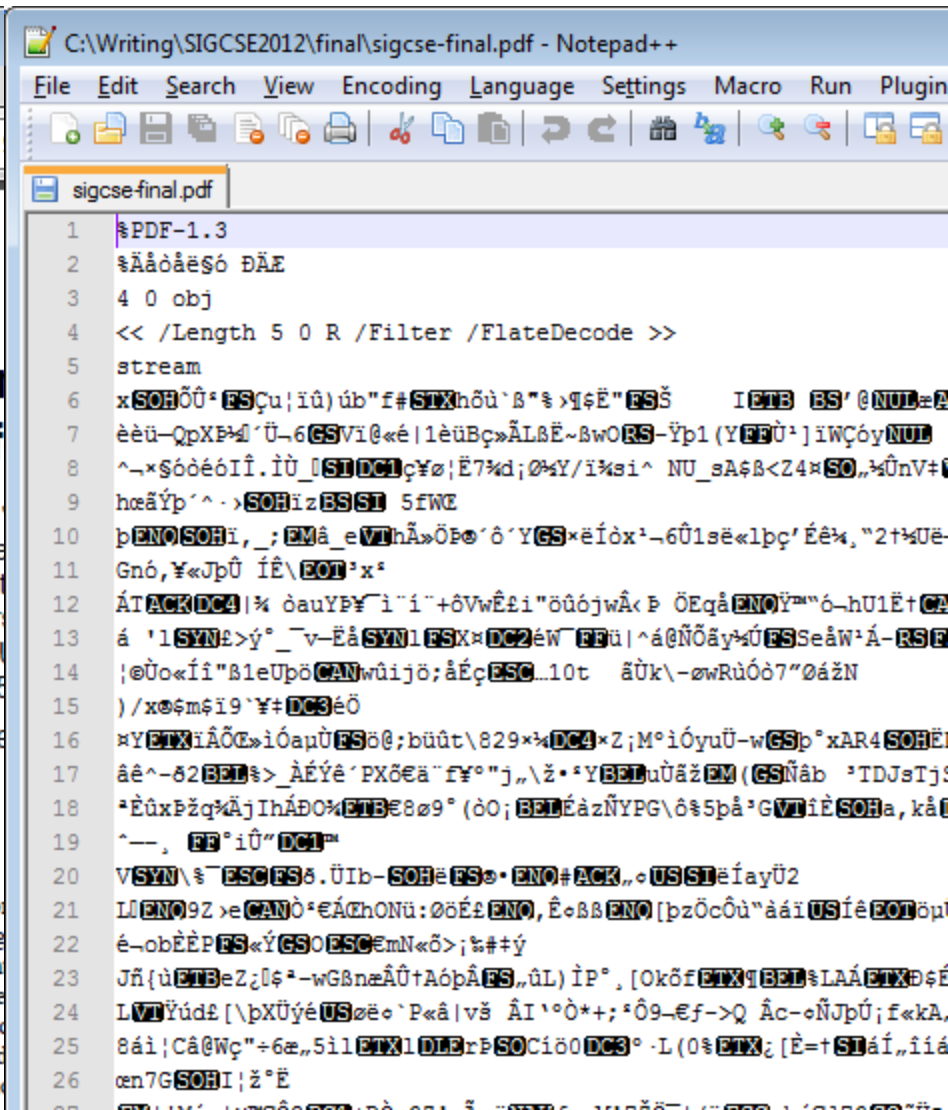
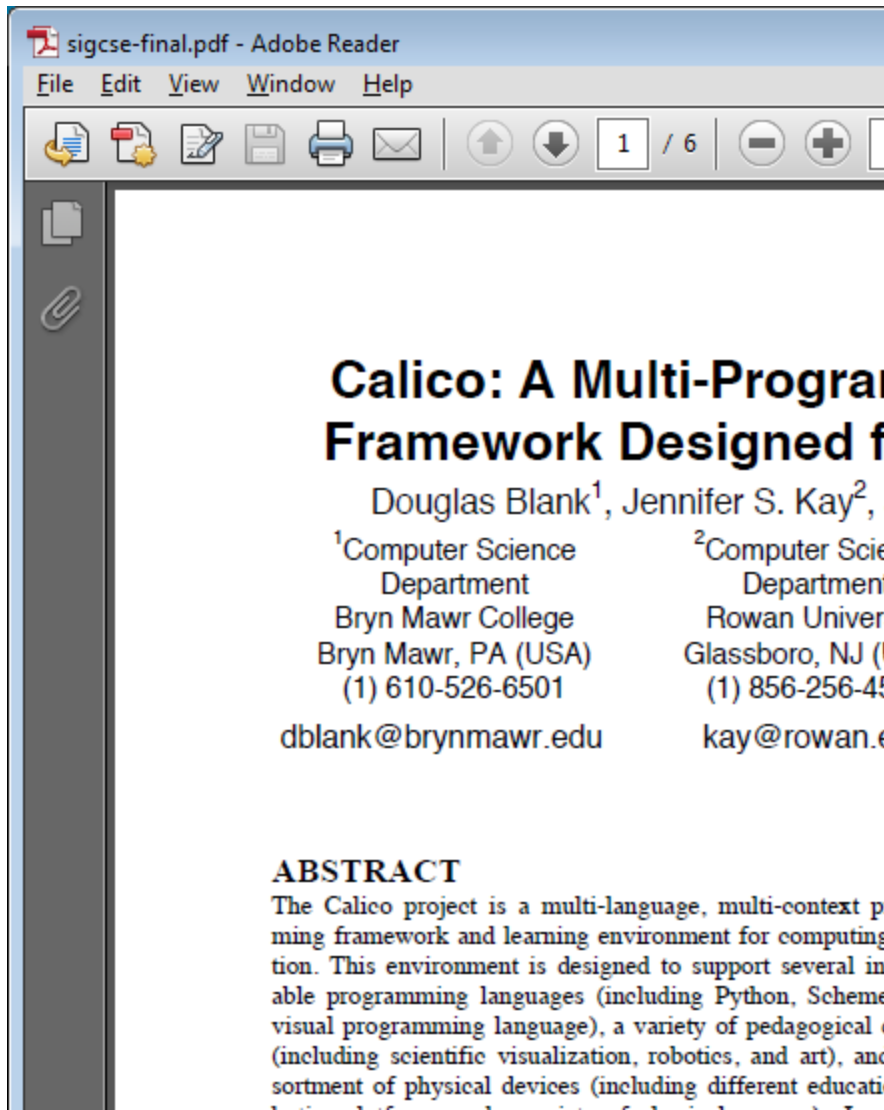
Files

- Proprietary encodings and compression means file data may not directly encode what you read



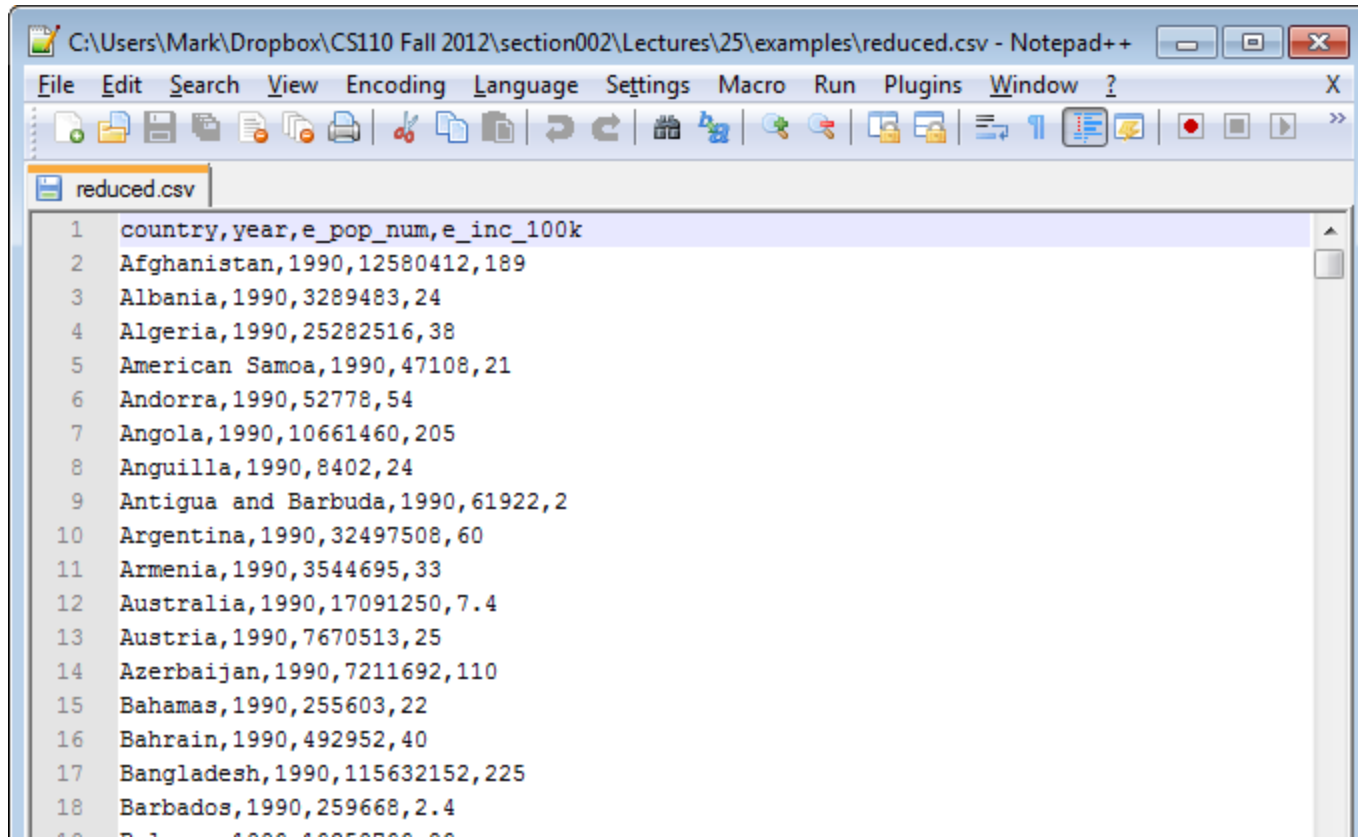
Files

- Proprietary encodings and compression means file data may not directly encode what you read



“Simple Text Files”

- All file data translates to readable content
 - no formatting, no compression
 - standard, non-proprietary encoding
 - each data item encodes a glyph

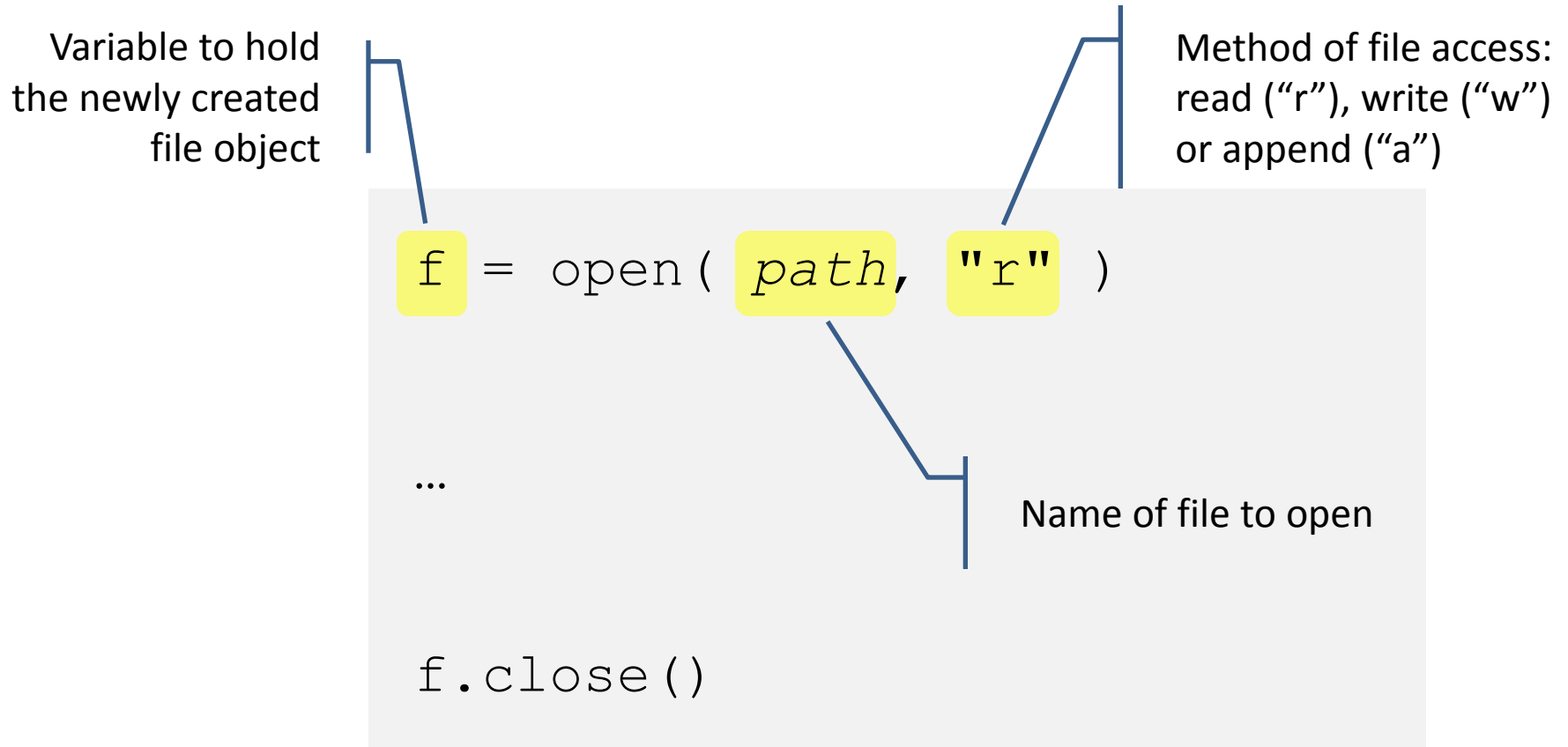


The screenshot shows a Notepad++ window with the title bar 'C:\Users\Mark\Dropbox\CS110 Fall 2012\section002\Lectures\25\examples\reduced.csv - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations and editing. The text area displays a CSV file named 'reduced.csv' with the following content:

```
1 country,year,e_pop_num,e_inc_100k
2 Afghanistan,1990,12580412,189
3 Albania,1990,3289483,24
4 Algeria,1990,25282516,38
5 American Samoa,1990,47108,21
6 Andorra,1990,52778,54
7 Angola,1990,10661460,205
8 Anguilla,1990,8402,24
9 Antigua and Barbuda,1990,61922,2
10 Argentina,1990,32497508,60
11 Armenia,1990,3544695,33
12 Australia,1990,17091250,7.4
13 Austria,1990,7670513,25
14 Azerbaijan,1990,7211692,110
15 Bahamas,1990,255603,22
16 Bahrain,1990,492952,40
17 Bangladesh,1990,115632152,225
18 Barbados,1990,259668,2.4
```

Opening/Closing Files

- To programmatically access data in a file, it must first be opened
- When finished, it must be closed



File Reading Methods

(Executed between `open()` and `f.close()`)

- `f.read()`
 - Read all data until end of file (EOF) is reached and return as a string object
- `f.readline()`
 - Read one entire line from the file (keeps the trailing newline character) and return as a string object
- `f.readlines()`
 - Read until EOF using `readline()` and return a list containing the lines thus read

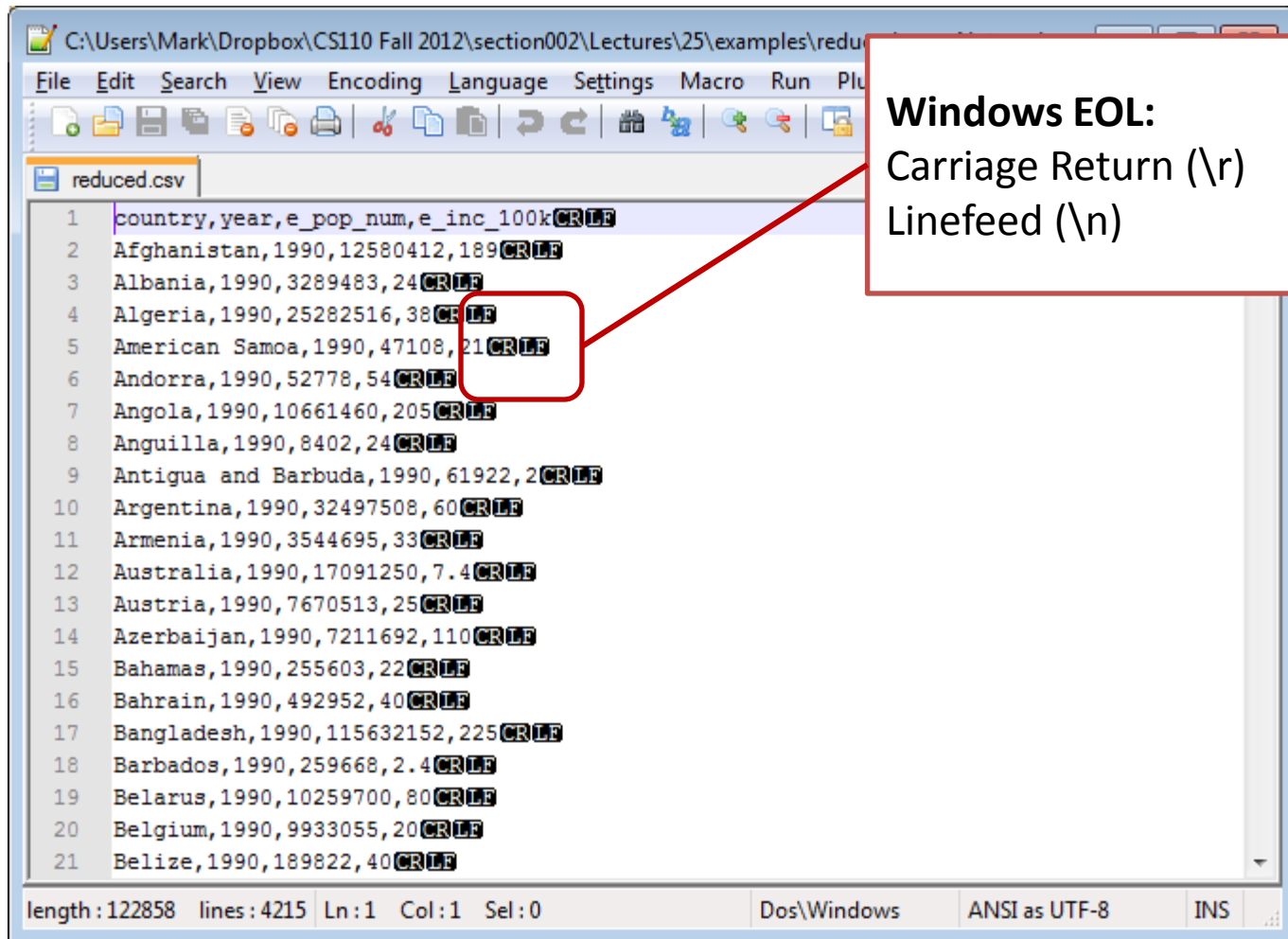
End-of-line (EOL) Markers

- The end-of-line in a file is marked by special non-glyph characters
- Varies by Operating System
 - Unix and Linux (and Mac OS X)
 - Use newline: `\n`
 - DOS and Windows
 - Use return + newline: `\r\n`
 - Old Mac OSs
 - Use return: `\r`



`\` is used as a special “escape” character

End-of-line (EOL) Markers



The screenshot shows a text editor window with a menu bar (File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plug-ins) and a toolbar. The file 'reduced.csv' is open. The text content is a CSV file with 21 lines. Each line ends with a highlighted 'CR LF' marker. A red box highlights the 'CR LF' markers on lines 4, 5, and 6. A red arrow points from this box to a callout box on the right. The callout box contains the text: 'Windows EOL: Carriage Return (\r) Linefeed (\n)'. The status bar at the bottom shows 'length: 122858 lines: 4215 Ln: 1 Col: 1 Sel: 0' and 'Dos\Windows ANSI as UTF-8 INS'.

```
1 country,year,e_pop_num,e_inc_100kCR LF
2 Afghanistan,1990,12580412,189CR LF
3 Albania,1990,3289483,24CR LF
4 Algeria,1990,25282516,38CR LF
5 American Samoa,1990,47108,21CR LF
6 Andorra,1990,52778,54CR LF
7 Angola,1990,10661460,205CR LF
8 Anguilla,1990,8402,24CR LF
9 Antigua and Barbuda,1990,61922,2CR LF
10 Argentina,1990,32497508,60CR LF
11 Armenia,1990,3544695,33CR LF
12 Australia,1990,17091250,7.4CR LF
13 Austria,1990,7670513,25CR LF
14 Azerbaijan,1990,7211692,110CR LF
15 Bahamas,1990,255603,22CR LF
16 Bahrain,1990,492952,40CR LF
17 Bangladesh,1990,115632152,225CR LF
18 Barbados,1990,259668,2.4CR LF
19 Belarus,1990,10259700,80CR LF
20 Belgium,1990,9933055,20CR LF
21 Belize,1990,189822,40CR LF
```

length: 122858 lines: 4215 Ln: 1 Col: 1 Sel: 0 Dos\Windows ANSI as UTF-8 INS

Windows EOL:
Carriage Return (\r)
Linefeed (\n)

End-of-line (EOL) Markers

- Python automatically translates Windows EOLs when reading and writing files on Windows platforms

WHO Tuberculosis Data

The screenshot shows a Mozilla Firefox browser window with the address bar displaying <http://www.who.int/tb/country/data/download/en/index.html>. The page features the WHO logo and a navigation menu with links to Health topics, Data and statistics, Media centre, Publications, Countries, Programmes and projects, and About WHO. The main heading is "Tuberculosis (TB)". On the left, a sidebar lists various topics including Tuberculosis, TB Topics index, Stop TB Strategy, DOTS expansion, TB diagnostics and laboratories, TB/HIV, MDR/XDR-TB, Health systems, Public-Private Mix, Affected people, TB research, TB data, TB publications, and About us. The main content area is titled "Download data as CSV files" and includes an "E-mail" link. It provides information about the data provided by countries to WHO and estimates of TB burden generated by WHO for *Global Tuberculosis Control 2010*. The text states that CSV files can be imported into many spreadsheet and database programs, and some spreadsheets such as Excel can open CSV files directly. It also mentions that the first row in each CSV file contains variable names; find the definition of each variable in the data dictionary. There are two download buttons: "» Download the data dictionary" and "» Download WHO estimates [760 KB]". The text below the second button states: "This includes WHO-generated estimates of TB mortality, prevalence, incidence (including incidence of HIV+TB) and case detection rate."

WHO | Download data as CSV files - Mozilla Firefox

File Edit View History Bookmarks Tools Help

WHO | Download data as CSV files

<http://www.who.int/tb/country/data/download/en/index.html>

Google

World Health Organization

Health topics Data and statistics Media centre Publications Countries Programmes and projects About WHO

Tuberculosis (TB)

Tuberculosis
TB Topics index
Stop TB Strategy
DOTS expansion
TB diagnostics and laboratories
TB/HIV
MDR/XDR-TB
Health systems
Public-Private Mix
Affected people
TB research
TB data
TB publications
About us

Download data as CSV files

E-mail

Data provided by countries to WHO and estimates of TB burden generated by WHO for *Global Tuberculosis Control 2010* are available for download as [comma-separated value \(CSV\)](#) files. CSV files can be imported into many spreadsheet and database programs, and some spreadsheets such as Excel can open CSV files directly.

The first row in each CSV file contains variable names; find the definition of each variable in the data dictionary.

Definition of variables: [» Download the data dictionary](#)

WHO TB burden estimates: [» Download WHO estimates \[760 KB\]](#)

This includes WHO-generated estimates of TB mortality, prevalence, incidence (including incidence of HIV+TB) and case detection rate.

<http://www.who.int/tb/country/data/download/en/index.html>

Microsoft Excel

Home Insert Page Layout Formulas Data Review View Developer

Normal Page Layout Page Break Preview Custom Views Full Screen

Workbook Views

☒ Ruler ☒ Formula Bar ☒ Gridlines ☒ Headings ☐ Message Bar

Show/Hide

Zoom 100% Zoom to Selection

New Window Arrange All Freeze Panes Split Hide View Side by Side Synchronous Scrolling Reset Window Position

Window

Save Workspace Switch Windows Macros

Macros

A1 country

Book1:2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	country	iso2	iso3	iso_numeric	g_whoregion	year	e_pop_num	e_prev_100k	e_prev_100k_lo	e_prev_100k_hi	e_prev_num	e_prev_num_lo	e_prev_num_hi	e_mort_exe	e_mort_exc_tbt
2	Afghanistan	AF	AFG	4	EMR	1990	12580412	452	196	754	57000	25000	95000	66	
3	Afghanistan	AF	AFG	4	EMR	1991	13427960	452	196	754	61000	26000	100000	66	
4	Afghanistan	AF	AFG	4	EMR	1992	14572340	452	196	754	66000	28000	110000	66	
5	Afghanistan	AF	AFG	4	EMR	1993	15861049	452	196	754	72000	31000	120000	66	
6	Afghanistan	AF	AFG	4	EMR	1994	17081664	452	196	754	77000	33000	130000	66	
7	Afghanistan	AF	AFG	4	EMR	1995	18083748	452	196	754	82000	35000	140000	66	
8	Afghanistan	AF	AFG	4	EMR	1996	18807500	452	196	754	85000	37000	140000	66	
9	Afghanistan	AF	AFG	4	EMR	1997	19303252	452	196	754	87000	38000	150000	66	
10	Afghanistan	AF	AFG	4	EMR	1998	19665668	452	196	754	89000	38000	150000	66	
11	Afghanistan	AF	AFG	4	FMR	1999	20041026	443	196	734	89000	39000	150000	63	

Sheet1 Sheet2 Sheet3

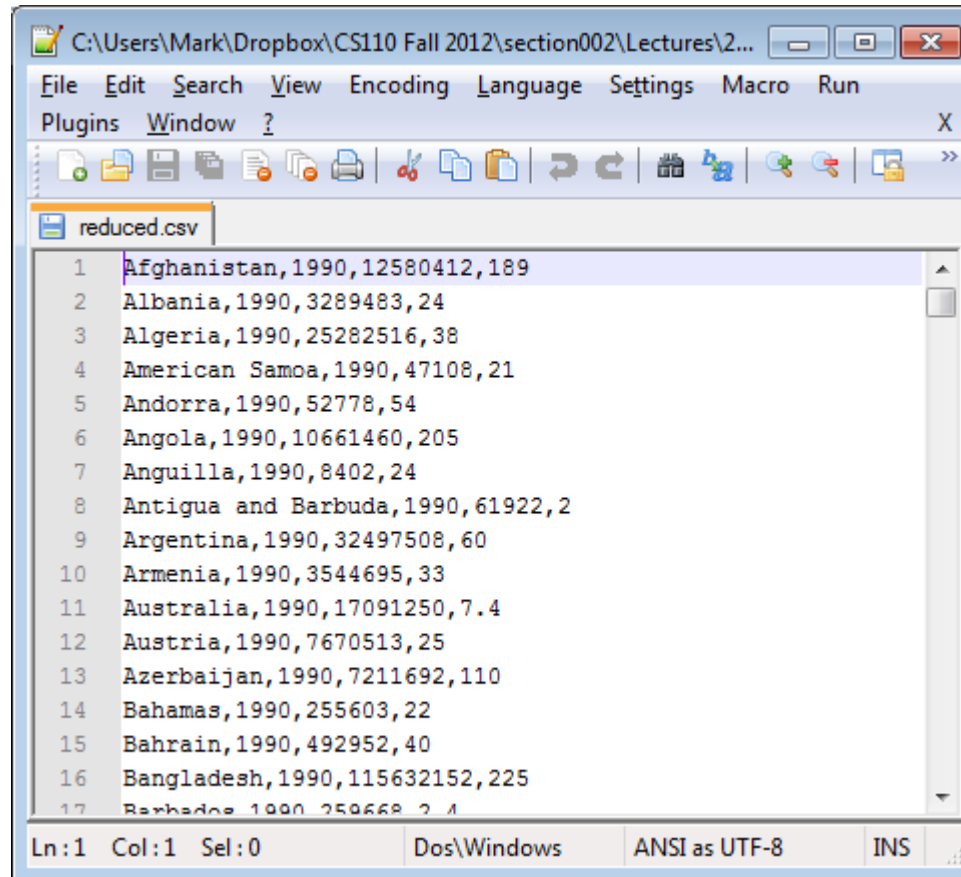
Book1:1

	A	B	C	D	E	F	G	H
1	variable_name	dataset	code_list	definition				
2	country	Country identification		Country or territory name				
3	iso_numeric	Country identification		ISO numeric country/territory code				
4	iso2	Country identification		ISO 2-character country/territory code				
5	iso3	Country identification		ISO 3-character country/territory code				
6	c_cdr	Estimates		Case detection rate (all forms), percent				
7	c_cdr_hi	Estimates		Case detection rate (all forms), percent, high bound				
8	c_cdr_lo	Estimates		Case detection rate (all forms), percent, low bound				
9	e_inc_100k	Estimates		Estimated incidence (all forms) per 100 000 population				
10	e_inc_100k_hi	Estimates		Estimated incidence (all forms) per 100 000 population, high bound				
11	e_inc_100k_lo	Estimates		Estimated incidence (all forms) per 100 000 population, low bound				

Sheet1 Sheet2 Sheet3

Ready Scroll Lock 100%

reduced.csv



1	Afghanistan,1990,12580412,189
2	Albania,1990,3289483,24
3	Algeria,1990,25282516,38
4	American Samoa,1990,47108,21
5	Andorra,1990,52778,54
6	Angola,1990,10661460,205
7	Anguilla,1990,8402,24
8	Antigua and Barbuda,1990,61922,2
9	Argentina,1990,32497508,60
10	Armenia,1990,3544695,33
11	Australia,1990,17091250,7.4
12	Austria,1990,7670513,25
13	Azerbaijan,1990,7211692,110
14	Bahamas,1990,255603,22
15	Bahrain,1990,492952,40
16	Bangladesh,1990,115632152,225
17	Barbados,1990,250668,2.4

In Excel: Select File, Save As, CSV (Comma delimited) (*.csv)

File Reading

```
f = open( path, "r" )
```

```
all      = f.read()
```

```
# Read entire file
```

```
line     = f.readline()
```

```
# Read one line
```

```
lines    = f.readlines()
```

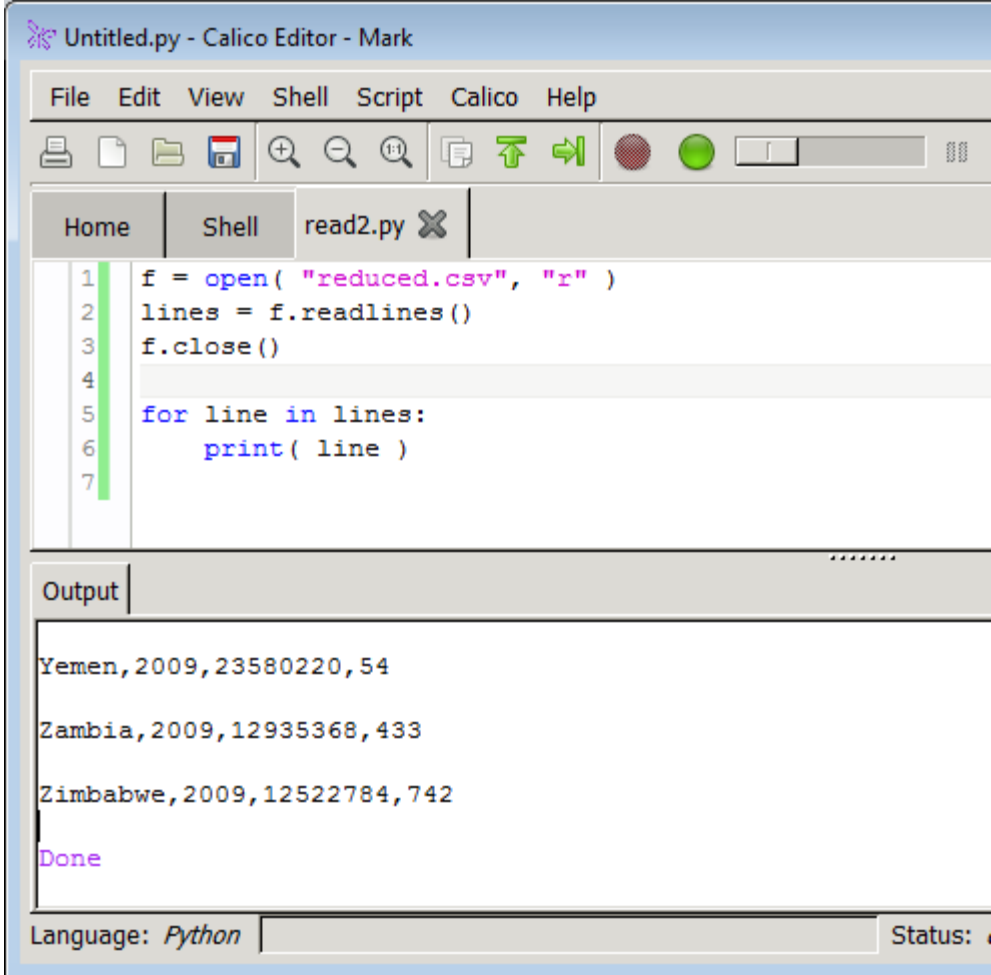
```
# Read a list of lines
```

```
f.close()
```

Reading Files – All lines at once

```
f = open( "reduced.csv", "r" )  
lines = f.readlines()  
f.close()
```

```
for line in lines:  
    print( line )
```



The screenshot shows the Calico Editor interface. The title bar reads "Untitled.py - Calico Editor - Mark". The menu bar includes "File", "Edit", "View", "Shell", "Script", "Calico", and "Help". The toolbar contains icons for file operations and execution. The editor window shows a Python script with the following code:

```
1 f = open( "reduced.csv", "r" )  
2 lines = f.readlines()  
3 f.close()  
4  
5 for line in lines:  
6     print( line )  
7
```

The "Output" panel at the bottom displays the results of the script execution:

```
Yemen,2009,23580220,54  
  
Zambia,2009,12935368,433  
  
Zimbabwe,2009,12522784,742  
  
Done
```

The status bar at the bottom indicates "Language: Python" and "Status: ".

read1.py

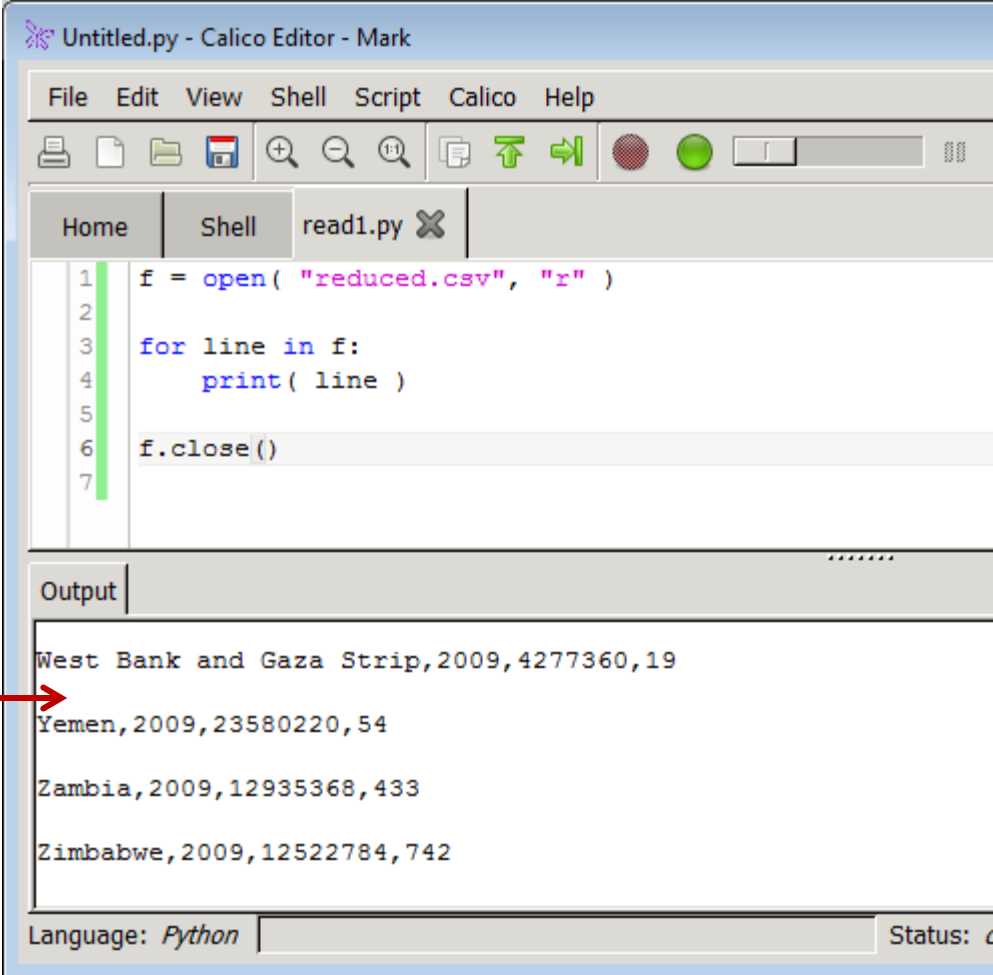
Reading Files with a for-loop

```
f = open( "reduced.csv", "r" )
```

```
for line in f:  
    print( line )
```

```
f.close()
```

Why the extra blank line?



The screenshot shows the Calico Editor interface. The top menu bar includes File, Edit, View, Shell, Script, Calico, and Help. Below the menu is a toolbar with icons for file operations and execution. The main editor window displays a Python script named 'read1.py' with the following code:

```
1 f = open( "reduced.csv", "r" )  
2  
3 for line in f:  
4     print( line )  
5  
6 f.close()  
7
```

Below the editor is an 'Output' pane showing the results of running the script. The output consists of four lines of text, each representing a record from the 'reduced.csv' file:

```
West Bank and Gaza Strip,2009,4277360,19  
Yemen,2009,23580220,54  
Zambia,2009,12935368,433  
Zimbabwe,2009,12522784,742
```

A red arrow points from the text 'Why the extra blank line?' to the first line of the output, 'West Bank and Gaza Strip,2009,4277360,19', indicating the question about the blank line preceding the first output line.

Language: Python Status: a

Reading Files with a for-loop

- Find the longest line in a file

```
f = open( "reduced.csv", "r" )

longest = ""
for line in f:
    if len(line) > len(longest):
        longest = line

f.close()
print( longest )
```

Splitting/Joining Strings

```
# A single line read from a file
```

```
line = "United States of America,1996,274066816,8.9"
```

```
# Split string into a list if substrings
```

```
# Can use any delimiter
```

```
items = line.split("," )
```

```
print( items )
```

```
>>> ['United States of America', '1996', '274066816', '8.9']
```

```
# Join a list of strings into a single combined string
```

```
joined = ",".join(items)
```

```
print( joined )
```

```
>>> United States of America,1996,274066816,8.9
```

Converting Strings to Numbers

```
syear = '1996'  
year = int(syear)  
print( type( year ) )
```

```
>>> <type 'int'>
```

```
stb100k = '8.9'  
tb100k = float(stb100k)  
print( type( tb100k ) )
```

```
>>> <type 'float'>
```

Reading Files and Converting to Usable Format

1. Read lines from a file, one at a time
2. Split lines on an appropriate separator character into a list of items (list of strings)
3. Convert individual list items into appropriate data types
4. Store converted data in lists or objects

Reading Files and Converting to Usable Format

```
# load.py
items = []

f = open( "reduced.csv", "r" ) # Open file

for line in f:                # Loop over all lines in file

    items = line.split(',')    # Split items

    country = items[0]         # Convert to usable data types
    year     = int( items[1] )
    pop      = int( items[2] )
    tb       = float( items[3] ) * 100000
                                # Store data together in a dictionary
    item = {'country':country, 'year':year, 'population':pop, 'tb':tb}

    items.append( item )      # Add to master list

f.close()                    # Close file
```

Removing whitespace chars with strip()

Whitespace characters:

- not associated with glyphs
- Include: space, tab, newline, carriage return, ...
- " \t\n\r"

Whitespace can be removed from the ends of a string using the `strip()` string method

```
s1 = "  abc"  
s2 = s1.strip()  
print( s1 )  
print( s2 )
```

```
>>>      abc  
>>> abc
```

String Interpolation (Substitution)

- Format one string and insert it into another string
- Uses '%' notation
- Substitution template on left, items on right

`<template string> % items`

- Template substitution specifiers:

<code>%s</code>	: substitute a string
<code>%d</code>	: substitute an integer
<code>%5d</code>	: substitute an integer formatted with 5 places
<code>%3.2f</code>	: substitute a float, 3 places to the left of integer, : and 2 places to the right

String Interpolation

```
>>> "Hello %s %s, you may have already won $%d" % ("Mr.", "Smith", 10000)
'Hello Mr. Smith, you may have already won $10000'

>>> 'This int, %5d, was placed in a field of width 5' % 7
'This int,      7, was placed in a field of width 5'

>>> 'This int, %10d, was placed in a field of width 10' % 10
'This int,      10, was placed in a field of width 10'

>>> 'This int, %10d, was placed in a field of width 10' % 7
'This int,      7, was placed in a field of width 10'

>>> 'This float, %10.5f, has width 10 and precision 5.' % 3.1415926
'This float,    3.14159, has width 10 and precision 5.'

>>> 'This float, %0.5f, has width 0 and precision 5.' % 3.1415926
'This float, 3.14159, has width 0 and precision 5.'

>>> 'Compare %f and %0.20f' % (3.14, 3.14)
'Compare 3.140000 and 3.14000000000000000010000'
```

String Interpolation (Substitution)

‘items’ can be a dictionary, and substitution specifiers can be dictionary keys in parentheses.

```
<template> % dictionary
```

```
msg = "The winner of %(award)s is %(name)s!"  
dsub1 = {'award': 'first place', 'name': 'Fido'}  
dsub2 = {'award': 'second place', 'name': 'Spot'}
```

```
print( msg % dsub1 )  
print( msg % dsub2 )
```

```
>>> The winner of first place is Fido!  
>>> The winner of second place is Spot!
```

File write methods

- `f.write(str)`
 - Write a string to the file
 - Note: Due to buffering, the string may not actually show up in the file until the `f.flush()` or `f.close()` method is called
- `f.writelines(sequence)`
 - Write a sequence of strings to the file
 - Note: Does not add line separators, but this can be done using the string `join` operator

Writing Files

- Open for write "w" overwrites file.
- Open for append "a" adds to what is already there.

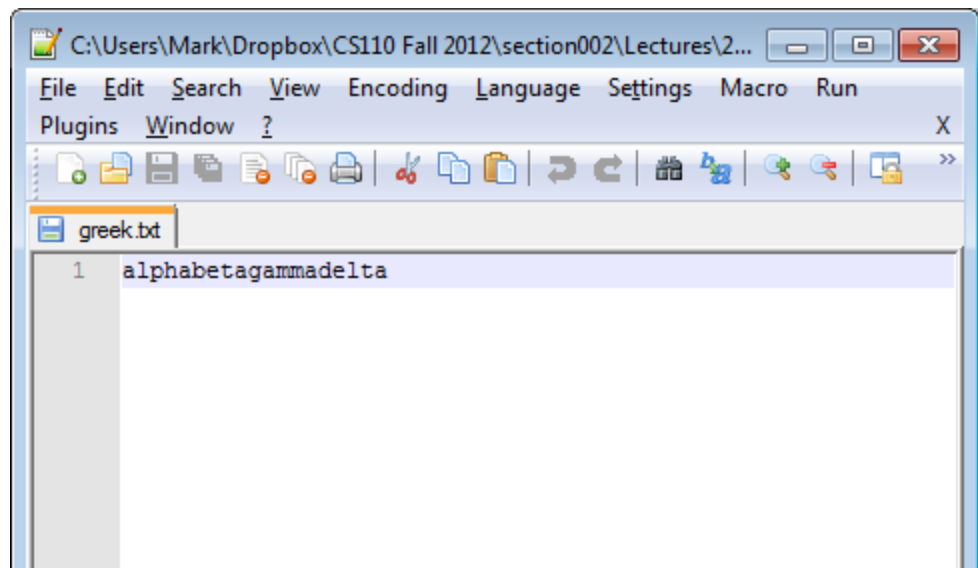
```
# writel.py
```

```
f = open("greek.txt", "w")
```

```
f.write( "alpha" )
```

```
f.writelines( ["beta", "gamma", "delta"] )
```

```
f.close()
```



Writing Files

- Write EOL characters by adding '\n'

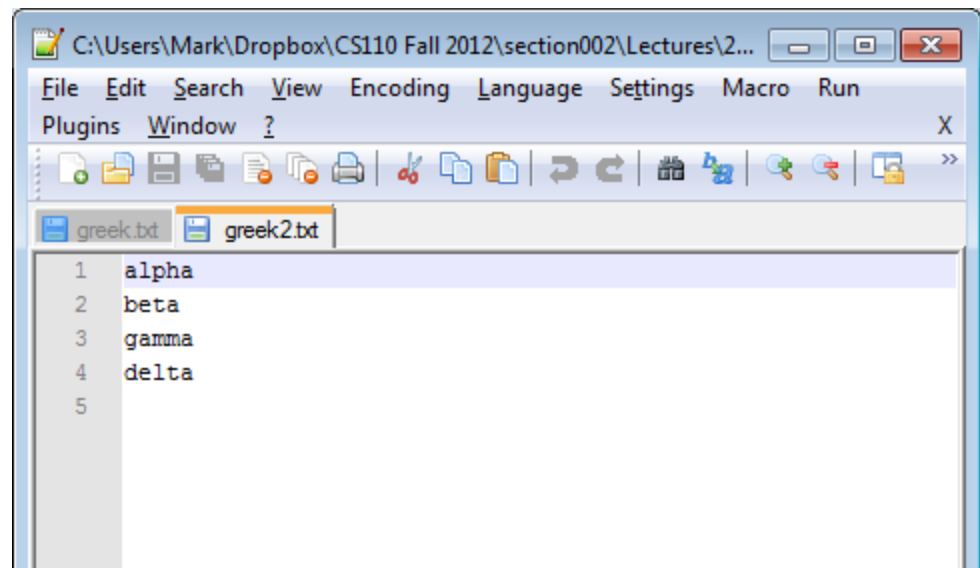
```
# write2.py
```

```
f = open("greek2.txt", "w")
```

```
f.write( "alpha\n" )
```

```
f.writelines( ["beta\n", "gamma\n", "delta\n"] )
```

```
f.close()
```



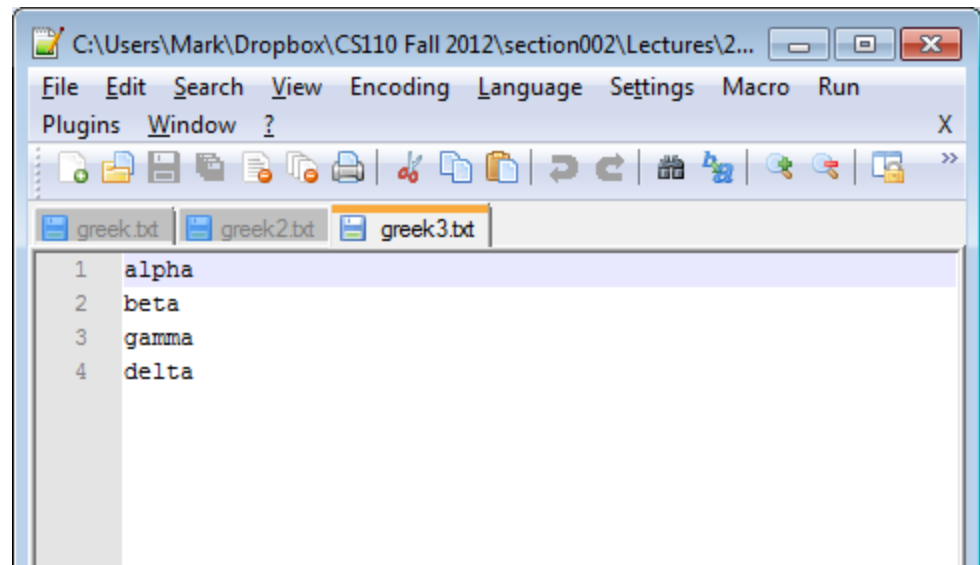
Writing Files

- Use join method of os.linesep string to build file contents

```
# write3.py
import os
f = open("greek3.txt", "w")

f.write( "alpha\n" )
items = ["beta", "gamma", "delta"]
sitems = os.linesep.join( items )
f.write( sitems )

f.close()
```



The **os** module

- Provides generic operating system (OS) support and a standard, platform-independent OS interface
- Includes tools for environments, processes, files, shell commands, and much more

File and directory commands

- `os.getcwd()`
 - Returns the name of the current working directory as a string
- `os.chdir(path)`
 - Changes the current working directory for this process to *path*, a directory name string
- `os.listdir(path)`
 - Returns a list of names of all the entries in the directory *path*

File and directory commands

The Calico Project, Version 2.1.1

```
python>>> import os
```

Ok

```
python>>> os.getcwd()
```

```
'C:\\CS110 Fall 2012\\section002\\Lectures\\25\\examples'
```

Ok

```
python>>> os.listdir( os.getcwd() )
```

```
['greek.txt', 'greek2.txt', 'greek3.txt', 'load.py',  
'longest.py', 'path.py', 'read1.py', 'read2.py',  
'reduced.csv', 'TB_burden_countries_2010.csv',  
'TB_data_dictionary_2011-04-10.csv', 'write1.py',  
'write2.py', 'write3.py']
```

Ok

Portability constants

- `os.curdir`
 - String for the current directory
- `os.pardir`
 - String for the parent directory
- `os.sep`
 - String used to separate directories
- `os.linesep`
 - String used to terminate lines

Portability constants

The Calico Project, Version 2.1.1

```
python>>> import os
```

Ok

```
python>>> os.curdir
```

```
'.'
```

Ok

```
python>>> os.pardir
```

```
'..'
```

Ok

```
python>>> os.sep
```

```
'\\'
```

Ok

```
python>>> os.linesep
```

```
'\r\n'
```

Ok