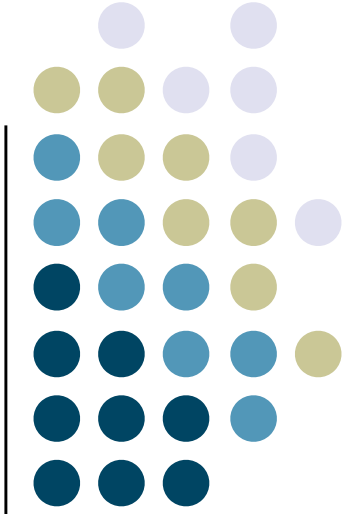
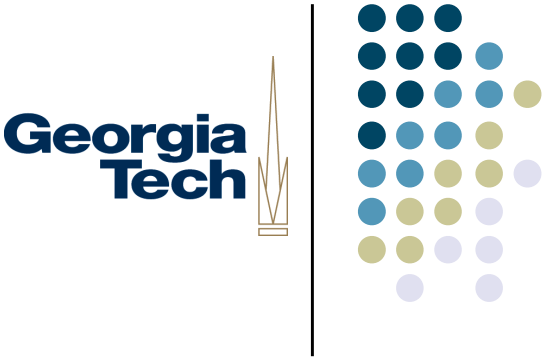
Manipulating Data Files in Python



Georgia Tech

Learning Objectives



- Working with CSV files
  - Reading and writing
  - Moving into and out of data structures
- Accessing files in other folders
- JSON files
  - Reading and writing
- Regular expressions

Data Files

- Last time we learned how to open, read from, and write to files
- Today we focus on different types of data files

with Statement

- Handy command to help with file ops
- Had code like

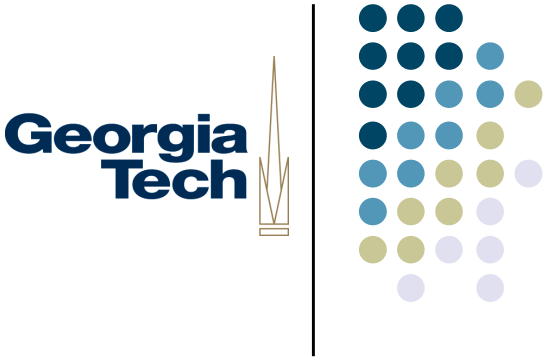
```
try:
    infile = open('sales_data.txt', 'r')
    for line in infile:
        # do something
    infile.close()

except IOError:
    print('An error occurred trying to read the file.')
```

- Can do

```
with open('sales_data.txt', 'r') as f:
    for line in f.readlines():
        # do something
```

- Does all useful close(), exception stuff

CSV Files

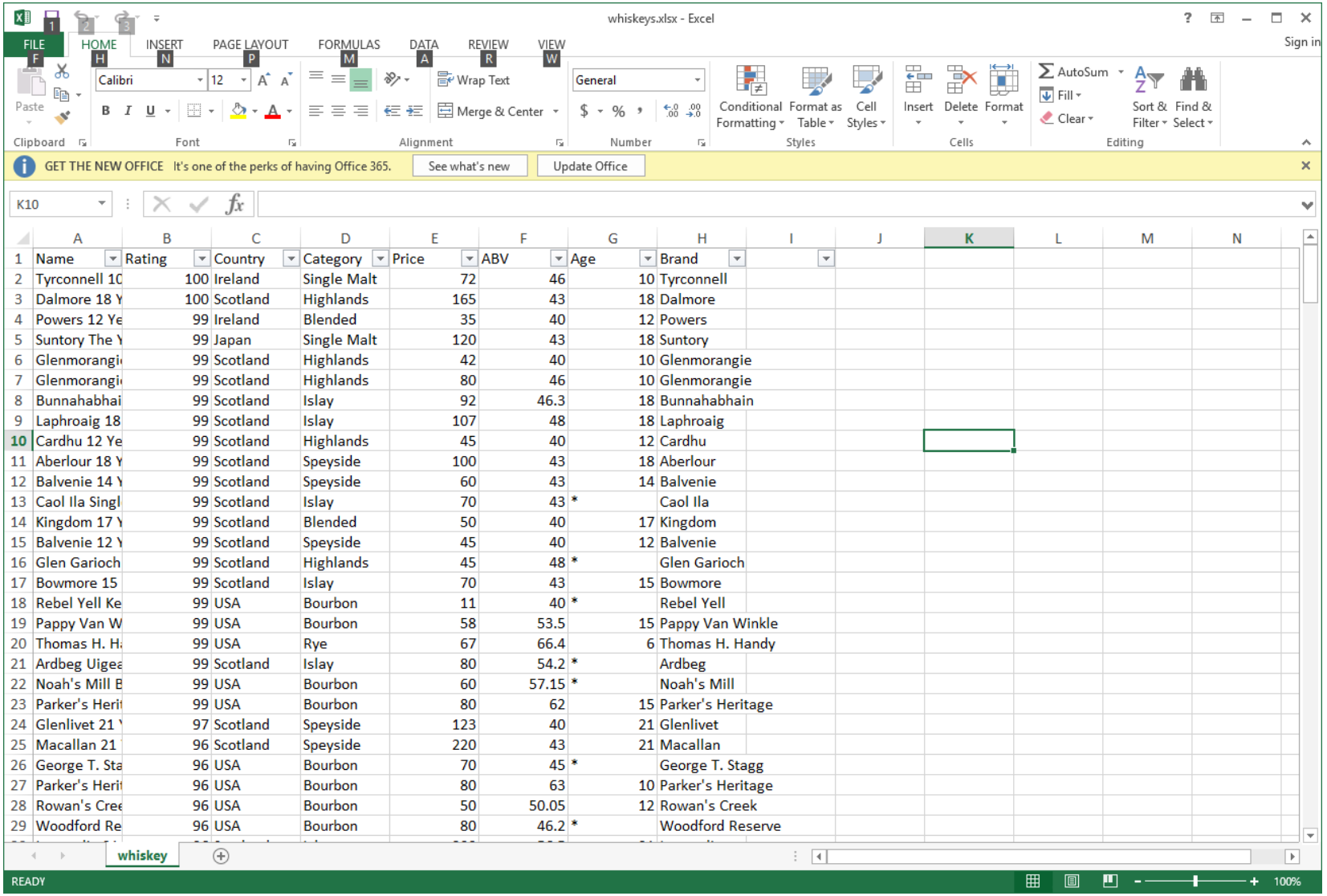


- Comma-separated values

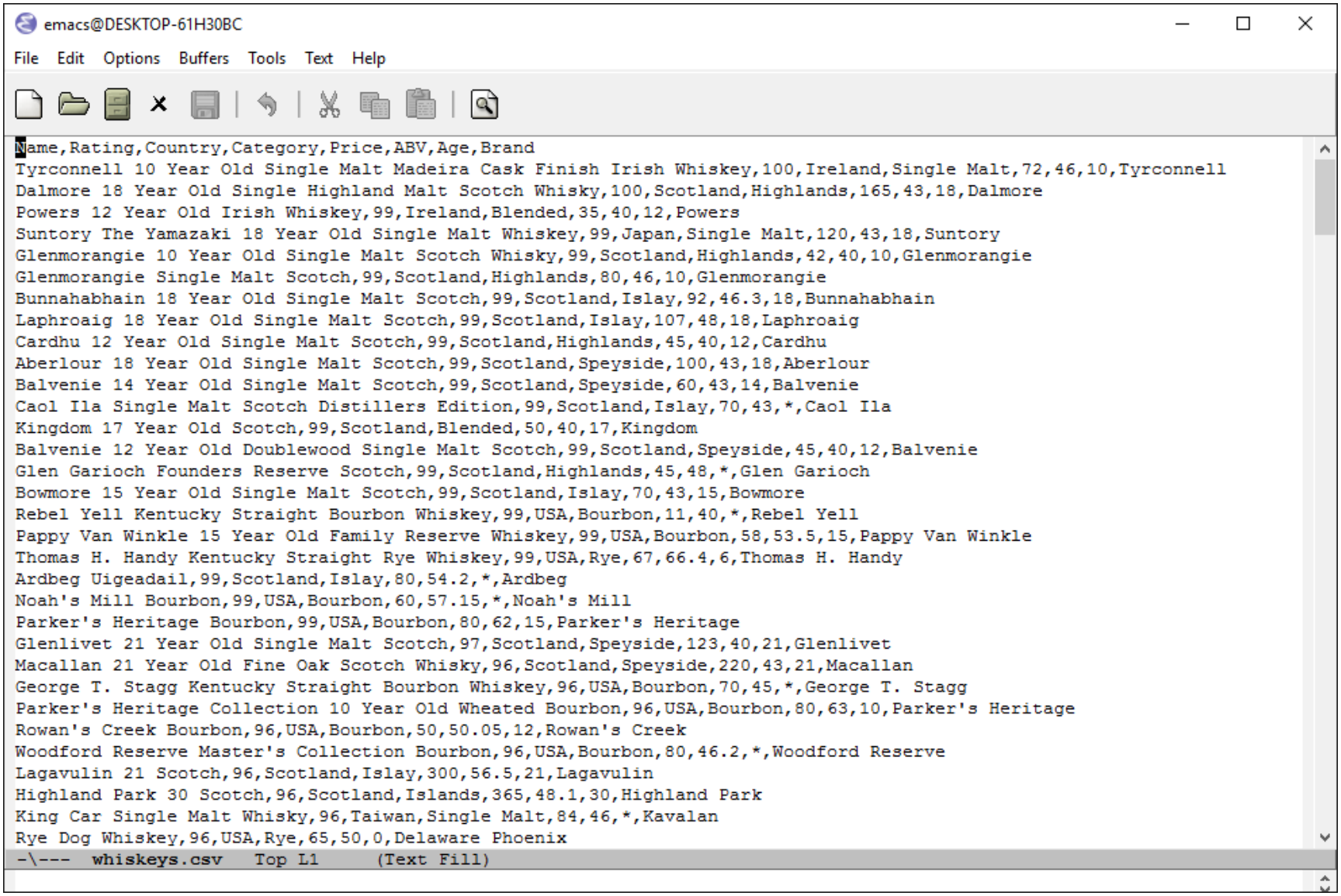
```
"Ford","Ranger","17.2","340"
"Hyundai","Genesis","23.8","260"
```

(quotes optional)

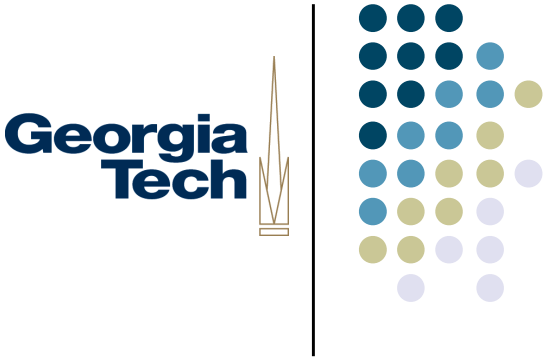
- Very common for tabular data
- Can be generated by spreadsheets such as Excel



	Name	Rating	Country	Category	Price	ABV	Age	Brand
2	Tyrconnell 10	100	Ireland	Single Malt	72	46	10	Tyrconnell
3	Dalmore 18 Y	100	Scotland	Highlands	165	43	18	Dalmore
4	Powers 12 Ye	99	Ireland	Blended	35	40	12	Powers
5	Suntory The Y	99	Japan	Single Malt	120	43	18	Suntory
6	Glenmorangi	99	Scotland	Highlands	42	40	10	Glenmorangie
7	Glenmorangi	99	Scotland	Highlands	80	46	10	Glenmorangie
8	Bunnahabhai	99	Scotland	Islay	92	46.3	18	Bunnahabhain
9	Laphroaig 18	99	Scotland	Islay	107	48	18	Laphroaig
10	Cardhu 12 Ye	99	Scotland	Highlands	45	40	12	Cardhu
11	Aberlour 18 Y	99	Scotland	Speyside	100	43	18	Aberlour
12	Balvenie 14 Y	99	Scotland	Speyside	60	43	14	Balvenie
13	Caol Ila Singl	99	Scotland	Islay	70	43	*	Caol Ila
14	Kingdom 17 Y	99	Scotland	Blended	50	40	17	Kingdom
15	Balvenie 12 Y	99	Scotland	Speyside	45	40	12	Balvenie
16	Glen Garioch	99	Scotland	Highlands	45	48	*	Glen Garioch
17	Bowmore 15	99	Scotland	Islay	70	43	15	Bowmore
18	Rebel Yell Ke	99	USA	Bourbon	11	40	*	Rebel Yell
19	Pappy Van W	99	USA	Bourbon	58	53.5	15	Pappy Van Winkle
20	Thomas H. Ha	99	USA	Rye	67	66.4	6	Thomas H. Handy
21	Ardbeg Uigea	99	Scotland	Islay	80	54.2	*	Ardbeg
22	Noah's Mill B	99	USA	Bourbon	60	57.15	*	Noah's Mill
23	Parker's Herit	99	USA	Bourbon	80	62	15	Parker's Heritage
24	Glenlivet 21 Y	97	Scotland	Speyside	123	40	21	Glenlivet
25	Macallan 21	96	Scotland	Speyside	220	43	21	Macallan
26	George T. Sta	96	USA	Bourbon	70	45	*	George T. Stagg
27	Parker's Herit	96	USA	Bourbon	80	63	10	Parker's Heritage
28	Rowan's Cree	96	USA	Bourbon	50	50.05	12	Rowan's Creek
29	Woodford Re	96	USA	Bourbon	80	46.2	*	Woodford Reserve

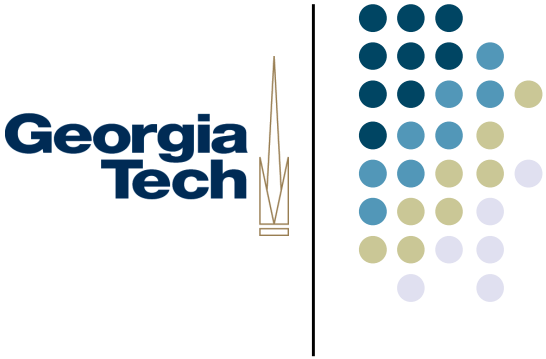

emacs@DESKTOP-61H30BC
File Edit Options Buffers Tools Text Help
Name,Rating,Country,Category,Price,ABV,Age,Brand
Tyrconnell 10 Year Old Single Malt Madeira Cask Finish Irish Whiskey,100,Ireland,Single Malt,72,46,10,Tyrconnell
Dalmore 18 Year Old Single Highland Malt Scotch Whisky,100,Scotland,Highlands,165,43,18,Dalmore
Powers 12 Year Old Irish Whiskey,99,Ireland,Blended,35,40,12,Powers
Suntory The Yamazaki 18 Year Old Single Malt Whiskey,99,Japan,Single Malt,120,43,18,Suntory
Glenmorangie 10 Year Old Single Malt Scotch Whisky,99,Scotland,Highlands,42,40,10,Glenmorangie
Glenmorangie Single Malt Scotch,99,Scotland,Highlands,80,46,10,Glenmorangie
Bunnahabhain 18 Year Old Single Malt Scotch,99,Scotland,Islay,92,46.3,18,Bunnahabhain
Laphroaig 18 Year Old Single Malt Scotch,99,Scotland,Islay,107,48,18,Laphroaig
Cardhu 12 Year Old Single Malt Scotch,99,Scotland,Highlands,45,40,12,Cardhu
Aberlour 18 Year Old Single Malt Scotch,99,Scotland,Speyside,100,43,18,Aberlour
Balvenie 14 Year Old Single Malt Scotch,99,Scotland,Speyside,60,43,14,Balvenie
Caol Ila Single Malt Scotch Distillers Edition,99,Scotland,Islay,70,43,*,Caol Ila
Kingdom 17 Year Old Scotch,99,Scotland,Blended,50,40,17,Kingdom
Balvenie 12 Year Old Doublewood Single Malt Scotch,99,Scotland,Speyside,45,40,12,Balvenie
Glen Garioch Founders Reserve Scotch,99,Scotland,Highlands,45,48,*,Glen Garioch
Bowmore 15 Year Old Single Malt Scotch,99,Scotland,Islay,70,43,15,Bowmore
Rebel Yell Kentucky Straight Bourbon Whiskey,99,USA,Bourbon,11,40,*,Rebel Yell
Pappy Van Winkle 15 Year Old Family Reserve Whiskey,99,USA,Bourbon,58,53.5,15,Pappy Van Winkle
Thomas H. Handy Kentucky Straight Rye Whiskey,99,USA,Rye,67,66.4,6,Thomas H. Handy
Ardbeg Uigeadail,99,Scotland,Islay,80,54.2,*,Ardbeg
Noah's Mill Bourbon,99,USA,Bourbon,60,57.15,*,Noah's Mill
Parker's Heritage Bourbon,99,USA,Bourbon,80,62,15,Parker's Heritage
Glenlivet 21 Year Old Single Malt Scotch,97,Scotland,Speyside,123,40,21,Glenlivet
Macallan 21 Year Old Fine Oak Scotch Whisky,96,Scotland,Speyside,220,43,21,Macallan
George T. Stagg Kentucky Straight Bourbon Whiskey,96,USA,Bourbon,70,45,*,George T. Stagg
Parker's Heritage Collection 10 Year Old Wheated Bourbon,96,USA,Bourbon,80,63,10,Parker's Heritage
Rowan's Creek Bourbon,96,USA,Bourbon,50,50.05,12,Rowan's Creek
Woodford Reserve Master's Collection Bourbon,96,USA,Bourbon,80,46.2,*,Woodford Reserve
Lagavulin 21 Scotch,96,Scotland,Islay,300,56.5,21,Lagavulin
Highland Park 30 Scotch,96,Scotland,Islands,365,48.1,30,Highland Park
King Car Single Malt Whisky,96,Taiwan,Single Malt,84,46,*,Kavalan
Rye Dog Whiskey,96,USA,Rye,65,50,0,Delaware Phoenix
-\--- whiskeys.csv Top L1 (Text Fill)

Read In?



- How would we read that file in?

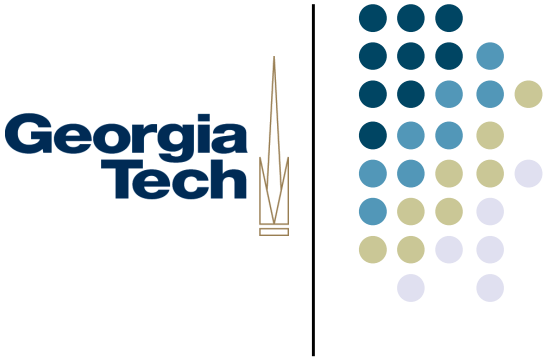
Simple Access



```
def readCSV(filename):
    file = open(filename, "r")
    lines = file.readlines()
    l = list()
    for line in lines:
        parts = line.split(",")
        l.append(parts)
        print(parts[0], parts[1])
    return l
```

Returns a list of lists

Tricky Stuff



- Potential issues?
  - Does it work with quoted items?
  - What if there are spaces between items?
  - What if an item has a comma inside it?

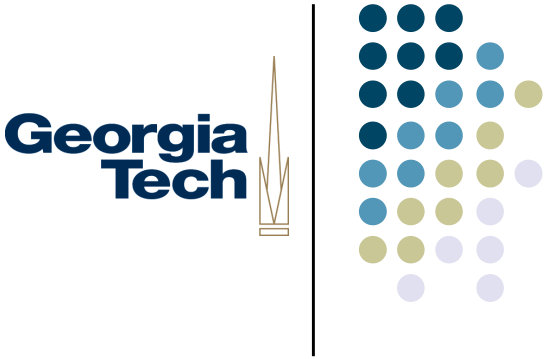
- Let's test

Getting the Files

- Might want to look into directories/folders on the local machine
- How do we explore them (inside a program) and possibly grab all the csv files in a folder?

- Need help from Python libraries

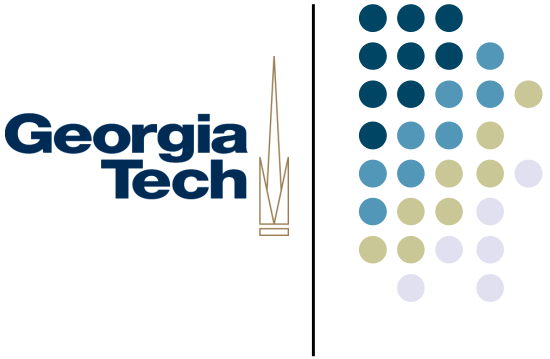
Useful Module




```
import os
```

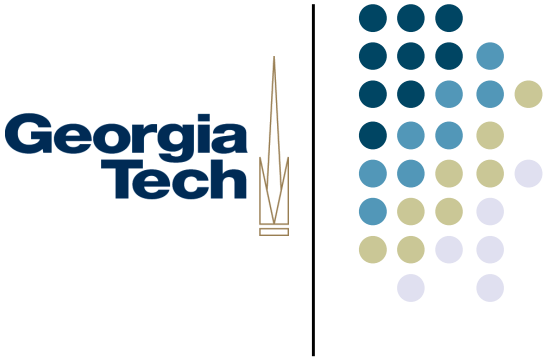
`os.listdir(dir)` – returns list of files in directory dir
`os.chdir(dir)` – change "active" directory to dir
`os.walk(dir)` – walk file system starting at dir

Get all the CSV's



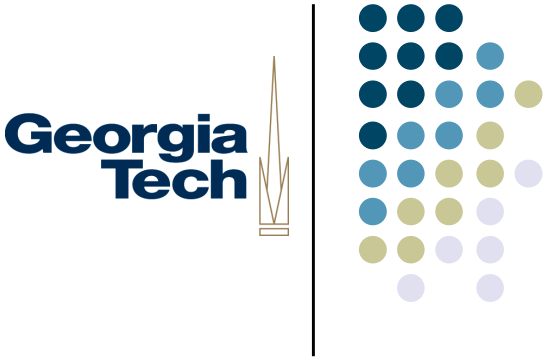

```
import os

files = os.listdir()
for item in files:
    if item.endswith(".csv"):
        csvFile = open(item, "r")
        # work on the file
        csvFile.close()
```

Walking through Folders



```
import os

for root, dirs, files in os.walk("data"):
    print(root, dirs, files)
    for filename in files:
        # create full name with path
        curr_file = os.path.join(root, filename)
        if curr_file.endswith("csv"):
            # work on the file
        else:
            continue
```

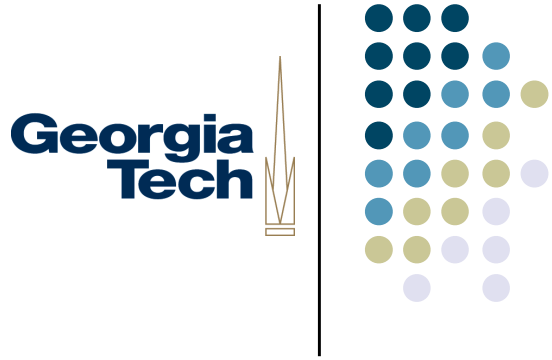
Reading CSV Files



- Don't need to do it ourself
- Python has module for that called…

csv

Using the Module



```
def readacsv(name):
    file = open(name,"r")
    csvfile = csv.reader(file)
    for row in csvfile:
        # do something
    file.close()
```

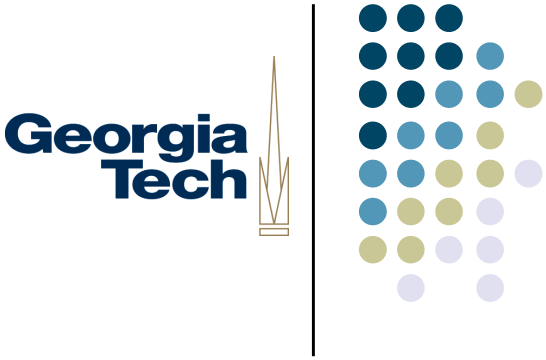
OR

```
def readacsv(name):
    with open(name) as f:
        csvfile = csv.reader(f)
        for row in csvfile:
            # do something
```

Why use the Module?

- Remember those earlier formatting problems
- The module handles them

Simple Access - Module

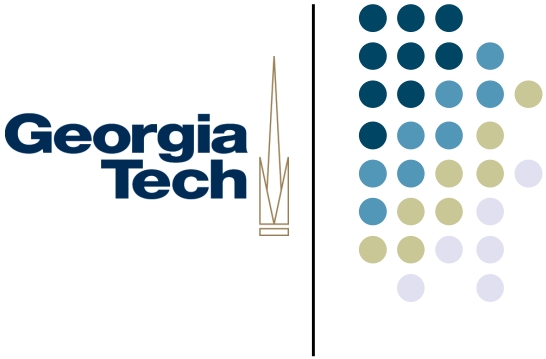


```
import csv

def readCSVbuiltin(filename):
    file = open(filename, "r")
    csvfile = csv.reader(file)
    l = list()
    for row in csvfile:
        l.append(row)
        print(row[0], row[1])
    return l
```

Returns a list of lists

Access as Dictionary

- Module has converter to dictionary
- If your file has a header row, that can be used
- Each row then will be a dictionary with key as the header field

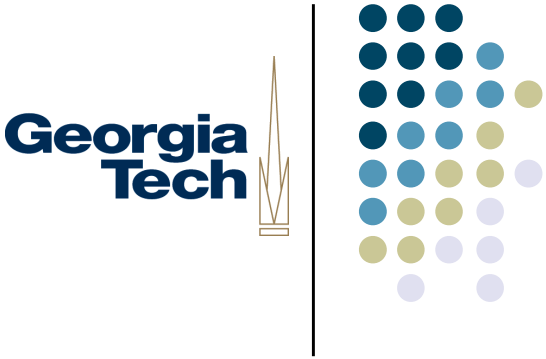


```
import csv

reader = csv.DictReader(open("students.csv"))

# check out the headers
print(reader.fieldnames)

# put them all in a list
myList = list(reader)
#  OR (but cant do both of these) Why?
# process them individually
for row in reader:
    print(row)
    print(row['age'])
```

Writing

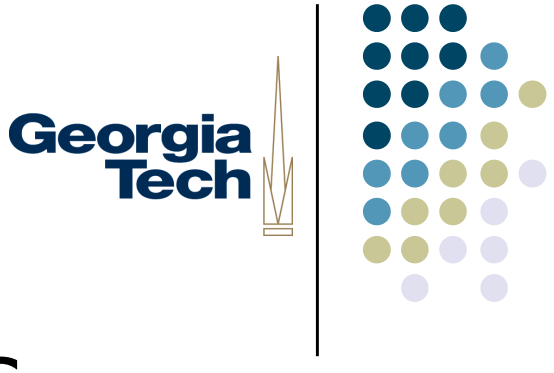
- What if you have a set (list) of dictionaries and you want to create a csv file?
- Handy `DictWriter` function for helping to do that
- Need to get the keys from the dictionary to use as the first row of the csv file

Write Example



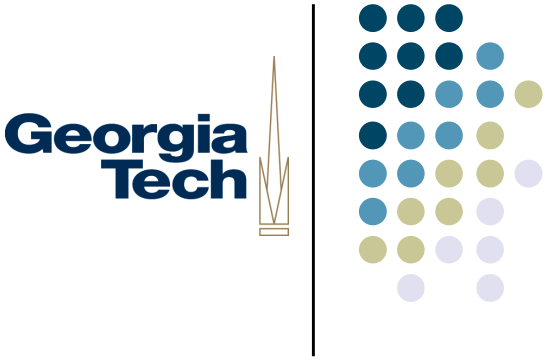
```
import csv

myDicts = [{"name":"bob", "age":23, "gender":"male"},
           {"name":"sue", "age":37, "gender":"female"}]

with open("people.csv", "w", newline='') as f:
    colnames = list(myDicts[0].keys())
    # for readability
    colnames.sort()
    writer = csv.DictWriter(f, fieldnames = colnames)
    writer.writeheader()
    for n in myDicts:
        writer.writerow(n)
```

Arguments



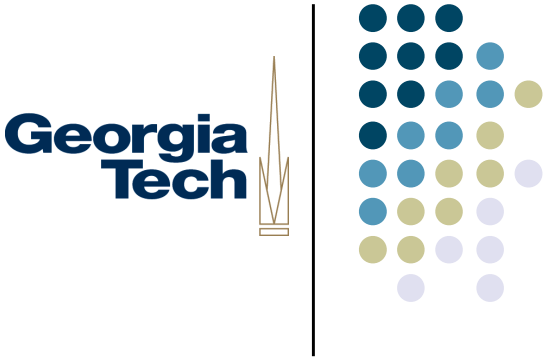
- csv reader has useful arguments
  - dialect: What type of csv file it is (default is 'excel'
  - delimiter: Items in file are usually comma separated but that can be changed
  - quotechar: The default is double quotes but that can be changed

JSON Files



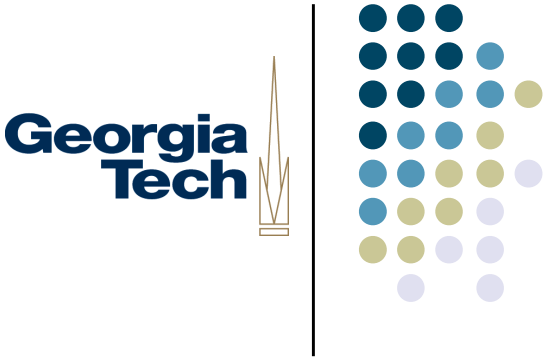
- JavaScript Object Notation
- Data exchange format
- Easy for people to read & write
- Easy for computers to parse & generate
- List of data objects (attribute, value) pairs

JSON Example



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
     "streetAddress": "21 2nd Street",
     "city": "New York",
     "state": "NY",
     "postalCode": "10021-3100"
   },
   "phoneNumbers": [
     {
       "type": "home",
       "number": "212 555-1234"
     },
     {
       "type": "office",
       "number": "646 555-4567"
     }
   ],
   "children": [],
   "spouse": null
}
```

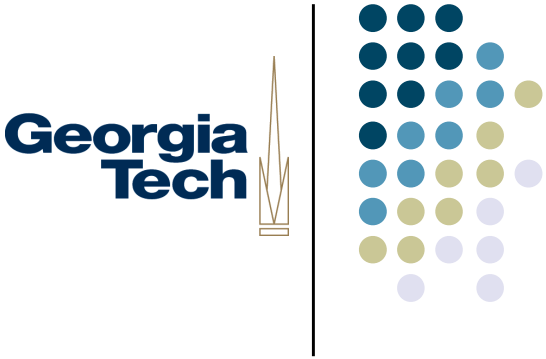
Writing JSON



Writing out to a JSON file from a list of dictionaries

```
import json

myDicts = [{"name":"bob", "age":23, "gender":"male"},
           {"name":"sue", "age":37, "gender":"female"}]

with open("people.json", "w") as f:
   json.dump(myDicts, f)
```

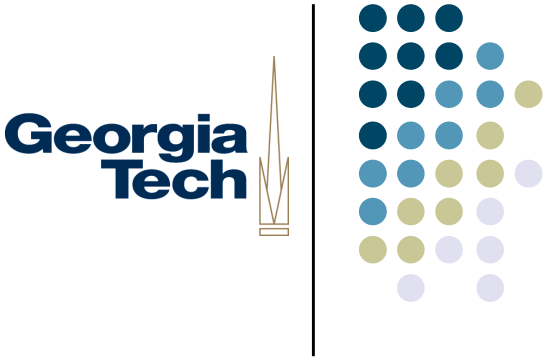
Reading JSON



Reading in a JSON file

```
import json

with open("people.json", "r") as f:
    myPeople = json.load(f)
```

Regular Expressions



Pattern matching on strings

```
import re
```

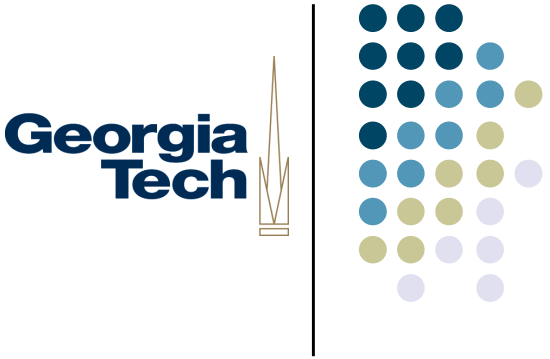
Bring in that module

```
re.split(pattern, string)
re.findall(pattern, string)
re.sub(pattern, replacement, string)
```

Useful functions

pattern should be `r'stuff'`

Symbols



`a` – the actual character a

`.` – match any single character except for newline

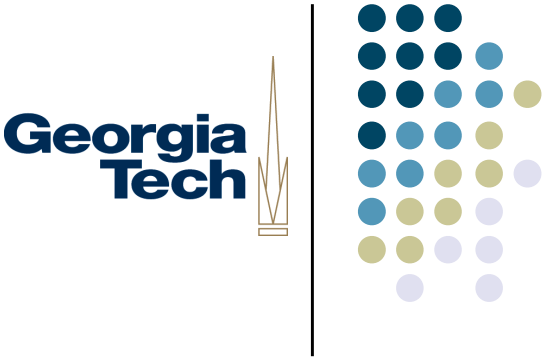
`+` – one or more occurrences of the pattern

`?` – zero or one occurrence of the pattern

`*` – zero or more repetitions of the pattern

`+?*` – operate on the character before then in the pattern

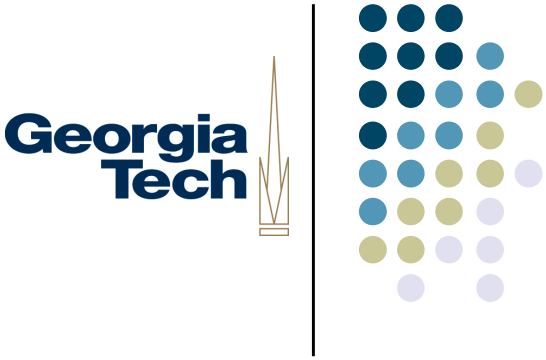
`a` – the actual character a
`.` – match any single character except for newline
`+` – one or more occurrences of the pattern
`?` – zero or one occurrence of the pattern
`*` – zero or more repetitions of the pattern



```
import re
re.split(r'a', 'Flatland')
['Fl', 'tl', 'nd']

re.split(r'txt', 'abc.txt')
['abc', '']

re.findall(r'a.', 'Flatland')
['at', 'an']

re.findall(r'.?a', 'Flatland')
['la', 'la']

re.findall(r'a.*', 'Flatland')
['atland']
```

For these two

```
re.findall(r'.?a', 'Flatland')
['la', 'la']

re.findall(r'a.*', 'Flatland')
['atland']
```

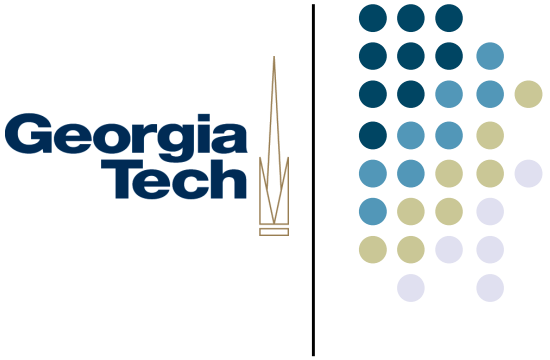


Would the following technically be right?

```
re.findall(r'.?a', 'Flatland')
['a', 'a']

re.findall(r'a.*', 'Flatland')
['and']
```

Python regular expressions are greedy by default
They try to match as many characters as possible

Special Patterns

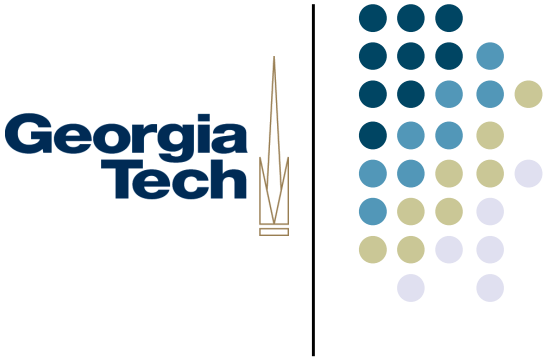


`\d` – decimal digit
`\s` – a whitespace
`\w` – an alphanumeric character

Capitals are opposites
`\D` – anything but a digit
`\S` – anything but a whitespace
`\W` – anything but alphanumeric chars

`a|b` – either a or b
`[ab]` – match both character a and b
`[1-5]` – any numbers in range 1 to 5
`^` – negation

Special Patterns



Assume

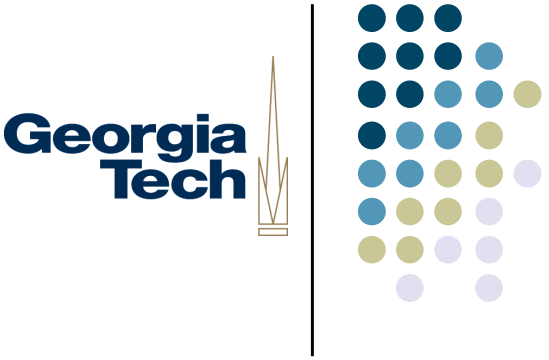
```
str = "3 Bacon  \n14 Eggs"
```

```
re.sub(r'Bacon|Eggs', 'Butter', str)
'3 Butter  \n14 Butter'

re.sub(r'[34]', '9', str)
'9 Bacon  \n19 Eggs'

re.sub(r'^[0-5]', '*', str)
'3*********14*****'
```

Review

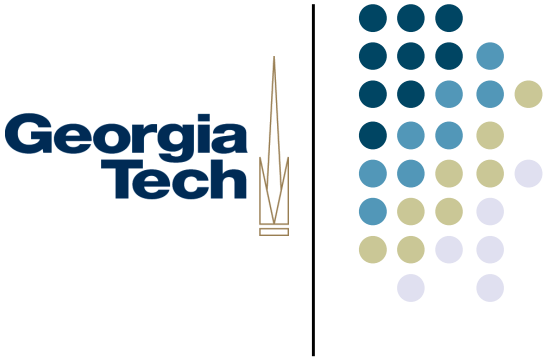
- Did you get the programming challenge?
- Print a sorted, counted list of all words in a document

Learning Objectives



- Working with CSV files
  - Reading and writing
  - Moving into and out of data structures
- Accessing files in other folders
- JSON files
  - Reading and writing
- Regular expressions

Next Time



- Accessing web data
  - Let's now go get datafiles from the web and work with them