



## **12 – File I/O**

COMP 125 Programming with Python

# Recording Disclaimer



The synchronous sessions are recorded (audiovisual recordings). The students are not required to keep their cameras on during class.

The audiovisual recordings, presentations, readings and any other works offered as the course materials aim to support remote and online learning. They are only for the personal use of the students. Further use of course materials other than the personal and educational purposes as defined in this disclaimer, such as making copies, reproductions, replications, submission and sharing on different platforms including the digital ones or commercial usages are strictly prohibited and illegal.

The persons violating the above-mentioned prohibitions can be subject to the administrative, civil, and criminal sanctions under the Law on Higher Education Nr. 2547, the By-Law on Disciplinary Matters of Higher Education Students, the Law on Intellectual Property Nr. 5846, the Criminal Law Nr. 5237, the Law on Obligations Nr. 6098, and any other relevant legislation.

The academic expressions, views, and discussions in the course materials including the audio-visual recordings fall within the scope of the freedom of science and art.

# How can we save data for later use?

- **Word processors:** write letters, reports, etc.
- **Image editors:** draw graphics and edit images
- **Spreadsheets:** insert numbers and mathematical formulas
- **Games:** list of player names, scores, current game status
- **Web browsers:** contents of shopping carts, etc.

## You will learn

- How to open a file
- How to read from a file
- How to write into a file
- How to close a file

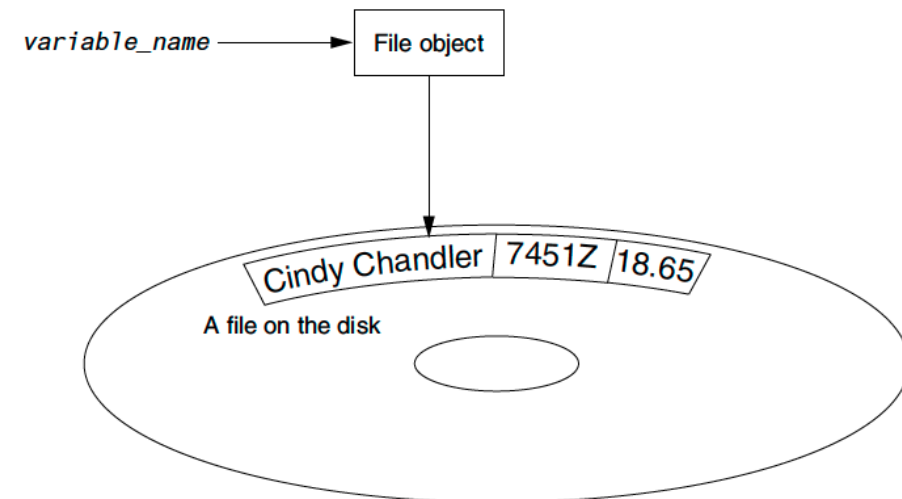


# Filenames and file objects

- **Files** are identified by a **filename**
  - Examples: cat.jpg , notes.txt , resume.docx
- **Filename extensions** represent the type of data stored
  - Examples: .jpg, .txt , .docx, .py
- The program must create a **file object** in memory to work with a file on the computer's disk



**Figure 6-4** A variable name references a file object that is associated with a file



# Opening a file

*file\_variable* = open(*filename*, *mode*)

name of the variable  
referencing the file object

string specifying the  
name of the file

Mode in which the file will  
be opened (reading, writing, etc.)

**Table 6-1** Some of the Python file modes

Mode	Description
'r'	Open a file for reading only. The file cannot be changed or written to.
'w'	Open a file for writing. If the file already exists, erase its contents. If it does not exist, create it.
'a'	Open a file to be written to. All data written to the file will be appended to its end. If the file does not exist, create it.

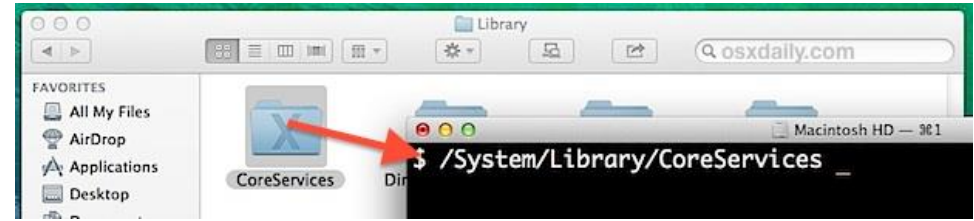
```
f1 = open('num.txt', 'r')  
f2 = open('num.txt', 'w')  
f3 = open('num.txt', 'a')
```

# Closing a file

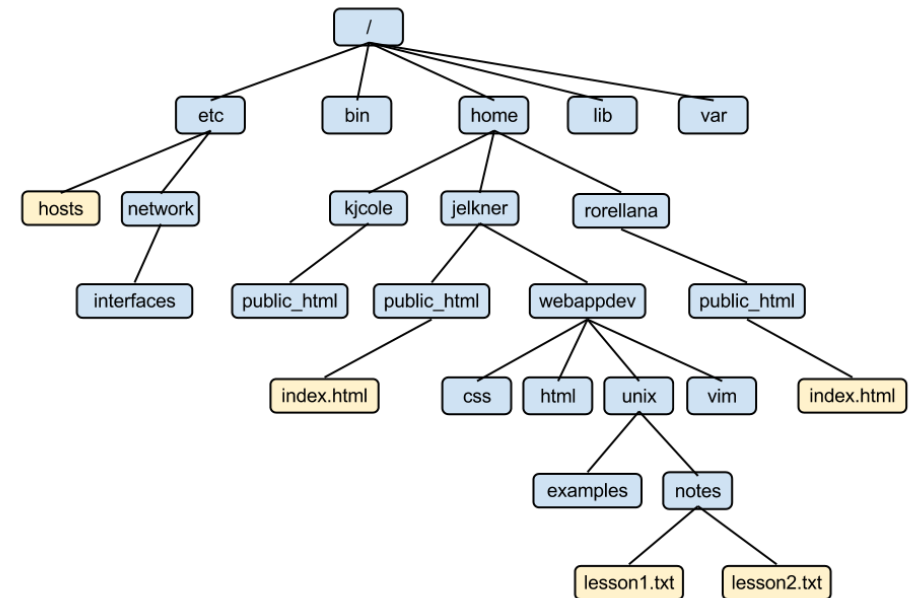
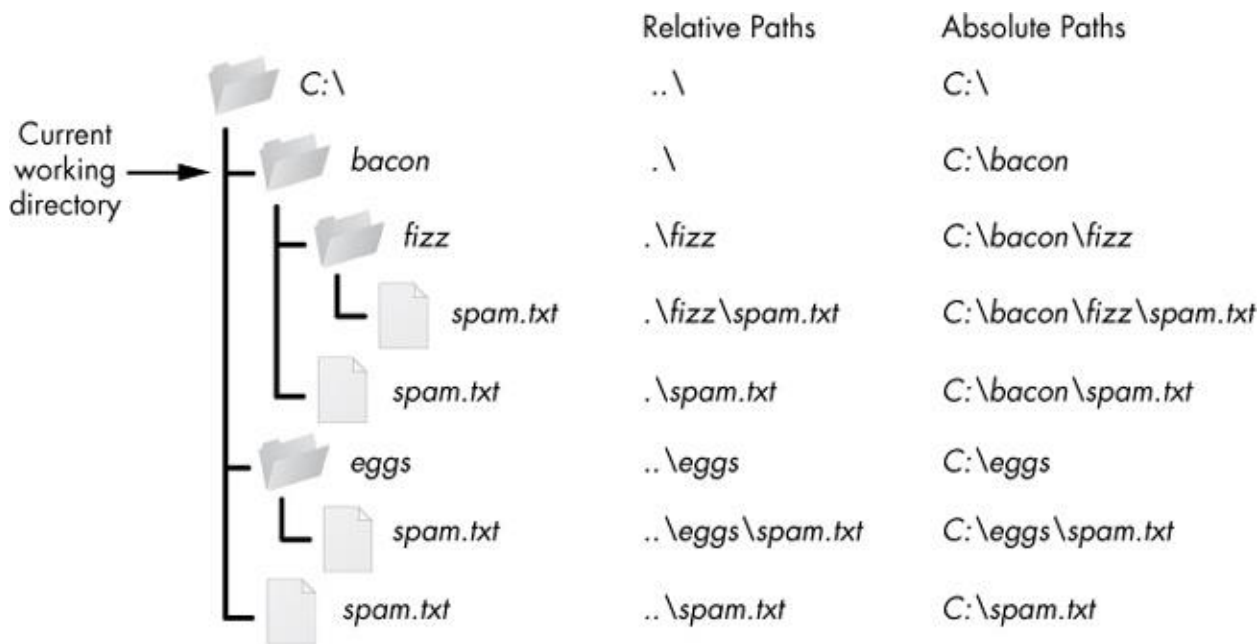
- Do not forget to close the file once you are done
- `close()` disconnects the program from the file to prevent data loss

```
f1 = open('num.txt', 'r')  
f2 = open('num.txt', 'w')  
f3 = open('num.txt', 'a')  
  
...  
  
f1.close()  
f2.close()  
f3.close()
```

# File locations - paths



- Files are kept on the secondary storage using file systems
- Almost all common file systems organize files into hierarchies (which form a tree) using drives and folders



# File locations - paths

- Need to specify the path of the file if it is not in the same directory/folder as our program
- These can be relative to the current directory or absolute
- **Hint:** to get the current directory of a Python program

```
import os
print(os.getcwd())
```

```
In [1]: import os
In [2]: print(os.getcwd())
/Users/cigdem/Desktop
In [3]: file_1 = open('/Users/cigdem/Documents/ex1.txt', 'w')
In [4]: file_2 = open('../Documents/ex2.txt', 'w')
In [5]: file_3 = open('./comp125/ex3.txt', 'w')
In [6]: file_4 = open('comp125/ex4.txt', 'w')
In [7]: file_5 = open('ex5.txt', 'w')
In [8]: file_1.close()
In [9]: file_2.close()
In [10]: file_3.close()
In [11]: file_4.close()
In [12]: file_5.close()
```

*These are for Mac*

```
test_file = open(r'C:\Users\Blake\temp\test.txt', 'w')
```

Specifies that the string is a raw string (backslash characters are read as literal backslashes)



# Writing data into a file

*file\_variable.write(string)*

It accepts a single string as an argument

```
no = 1
```

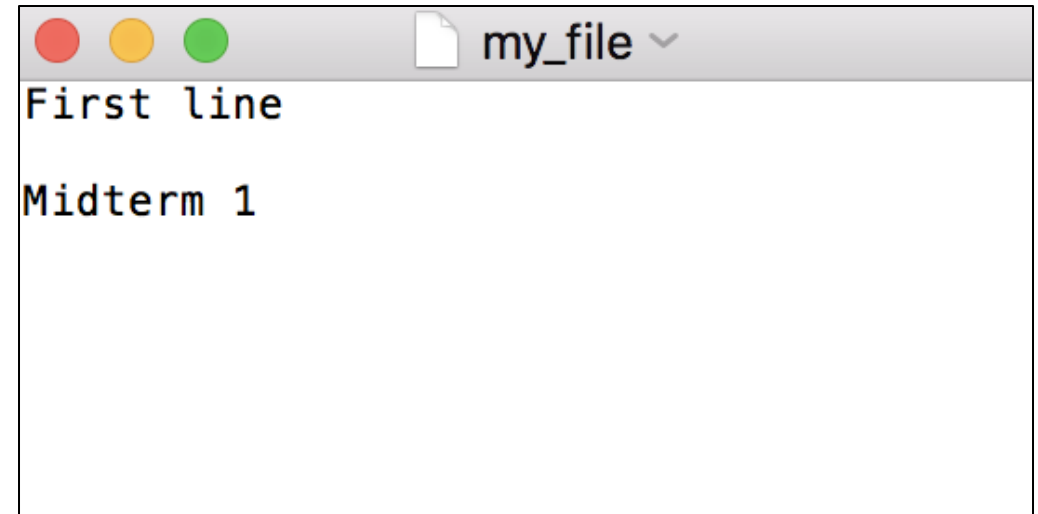
```
s = 'Midterm'
```

```
fid = open('my_file', 'w')
```

```
fid.write('First line\n\n')
```

```
fid.write(s + ' ' + str(no))
```

```
fid.close()
```

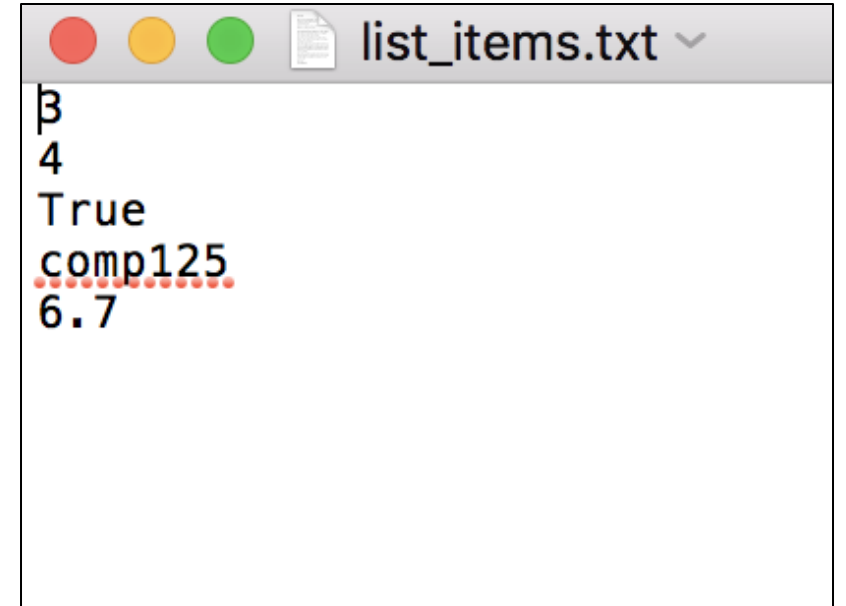


# Example

- Write the items of the following list into a file. Write one list item in a single line.

```
L = [ 3, 4, True, 'comp125', 6.7 ]
```

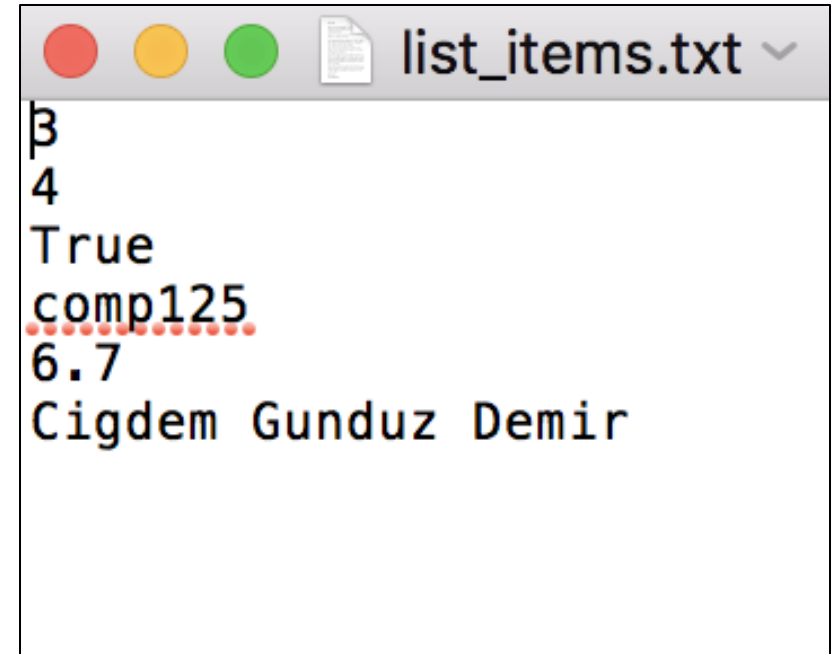
```
fvar = open('list_items.txt', 'w')
for i in range(len(L)):
    fvar.write(str(L[i]) + '\n')
fvar.close()
```



# Example

- Add your name at the end of the file that you created in the previous example.

```
fvar = open('list_items.txt', 'a')  
fvar.write('Cigdem Gunduz Demir')  
fvar.close()
```



```
3  
4  
True  
comp125  
6.7  
Cigdem Gunduz Demir
```

# Using string formatting with write

```
name = 'John'  
age = 22
```

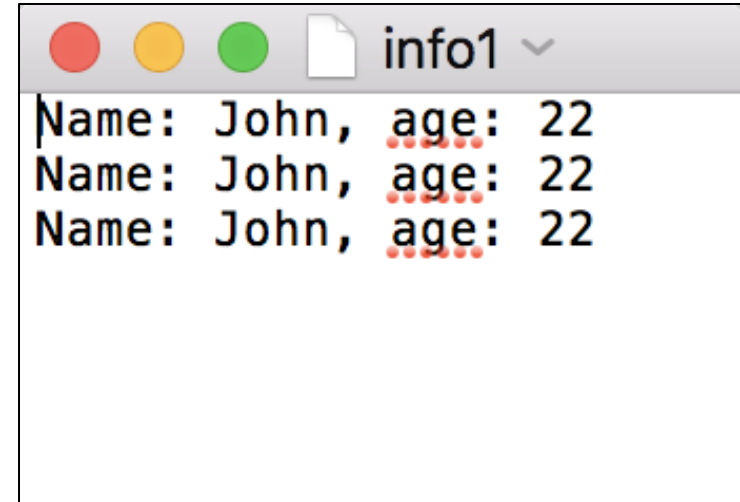
```
file1 = open('info1', 'w')
```

```
file1.write('Name: %s, age: %d\n' % (name, age))
```

```
file1.write('Name: {0}, age: {1}\n'.format(name, age))
```

```
file1.write(f'Name: {name}, age: {age}\n')
```

```
file1.close()
```



# Using string formatting with write

```
floatValue = 123456.789
```

```
integerValue = 4237
```

```
file2 = open('info2', 'w')
```

```
file2.write('One way %.8d' % integerValue)
```

```
file2.write('Another way: %.2d\t' % integerValue)
```

```
file2.write('(1) %.2f\n' % floatValue)
```

```
file2.write('(1) %.5f\n' % floatValue)
```

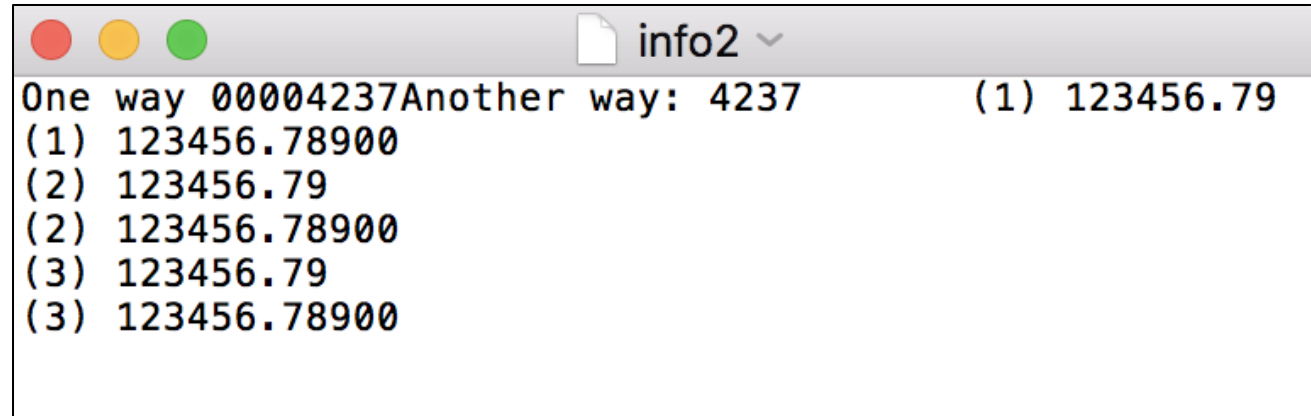
```
file2.write('(2) {:.2f}\n'.format(floatValue))
```

```
file2.write('(2) {:.5f}\n'.format(floatValue))
```

```
file2.write(f'(3) {floatValue:.2f}\n')
```

```
file2.write(f'(3) {floatValue:.5f}\n')
```

```
file2.close()
```



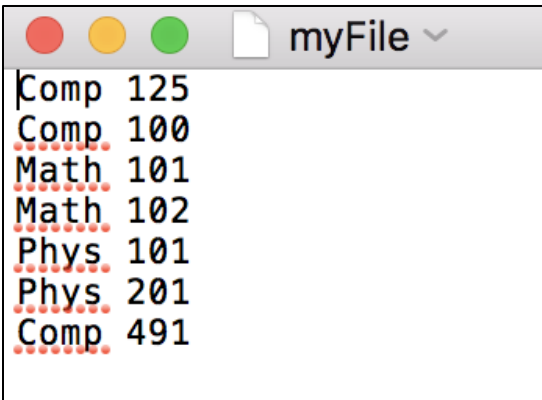
```
info2
One way 00004237Another way: 4237      (1) 123456.79
(1) 123456.78900
(2) 123456.79
(2) 123456.78900
(3) 123456.79
(3) 123456.78900
```

# Reading data from a file

- If you open your file for reading (with 'r' method), you can use the file object's **read**, **readline**, and **readlines** methods
- **read**: File object function that reads entire file contents into memory
  - Only works if the file has been opened for reading
  - Contents returned as a string
- **readline**: File object function that reads a line from the file
  - Line returned as a string, including ' \n '
- **readlines**: Read and return a list of lines from the stream file.
  - Lines returned as a list of strings

# Example

- Suppose that you have the following file that contains the department abbreviations and the course codes separated with white space.



```
Comp 125
Comp 100
Math 101
Math 102
Phys 101
Phys 201
Comp 491
```

- Write a function that takes the name of this file as its input, reads its contents, and returns a list of course tuples. You may assume that the file content is always valid.

```
def read_courses(filename):
    infile = open(filename, 'r')
    contents = infile.read()
    items = contents.split()
    i = 0
    L = []
    while i < len(items):
        L.append((items[i], items[i+1]))
        i += 2
    infile.close()
    return L

def main():
    fname = input('Enter the filename: ')
    L = read_courses(fname)
    print(L)

main()
```

```
In [6]: runfile('/Users/cigdem/Desktop/example2.py',
wdir='/Users/cigdem/Desktop')

Enter the filename: myFile
[('Comp', '125'), ('Comp', '100'), ('Math', '101'),
('Math', '102'), ('Phys', '101'), ('Phys', '201'),
('Comp', '491')]
```

# Using readline

- Reads the file contents, one line at a time

```
fid = open('myFile2', 'r')
```

```
l1 = fid.readline()
```

```
l2 = fid.readline()
```

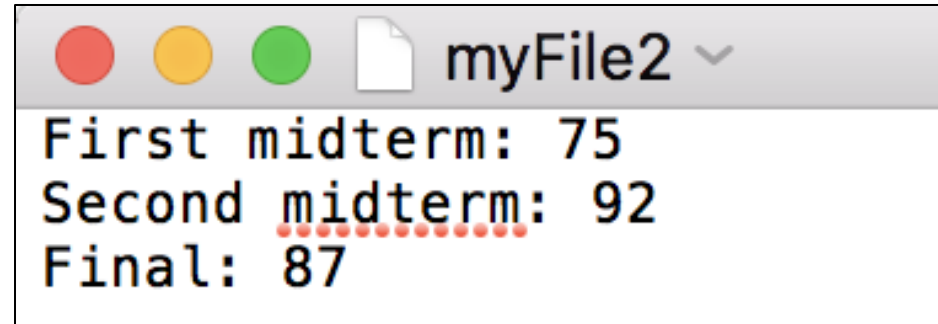
```
l3 = fid.readline()
```

```
print('line 1: ', l1)
```

```
print('line 2: ', l2)
```

```
print('line 3: ', l3)
```

```
fid.close()
```



```
In [1]: runfile('/Users/cigdem/De  
line 1: First midterm: 75  
  
line 2: Second midterm: 92  
  
line 3: Final: 87
```



# Using a for loop

- **for** loop automatically reads the lines in a file
  - without requiring a *priming read* operation
  - without testing a special condition that signals the end of the file

```
for variable in file_object:  
    statement  
    statement  
    etc.
```

# Example

```
def read_courses(filename):
    infile = open(filename, 'r')
    contents = infile.read()
    items = contents.split()
    i = 0
    L = []
    while i < len(items):
        L.append((items[i], items[i+1]))
        i += 2
    infile.close()
    return L

def main():
    fname = input('Enter the filename: ')
    L = read_courses(fname)
    print(L)

main()
```

```
def read_courses(filename):
    infile = open(filename, 'r')
    L = []
    for line in infile:
        items = line.split()
        L.append((items[0], items[1]))
    infile.close()
    return L

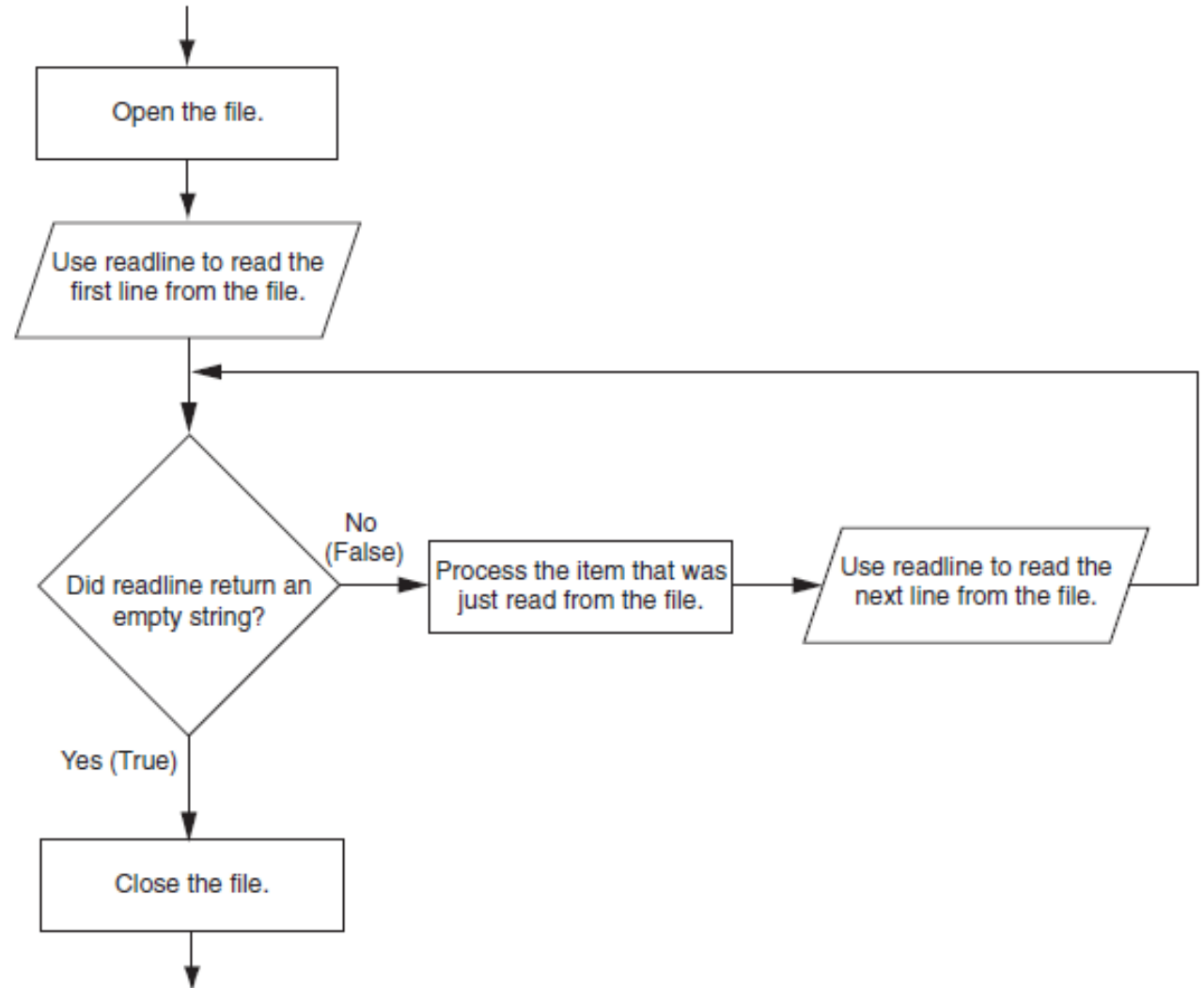
def main():
    fname = input('Enter the filename: ')
    L = read_courses(fname)
    print(L)

main()
```

# Using a while loop

- Need to know when the end of the file has been reached
- The `readline` function return an empty string (") when it tries to read beyond the end of a file

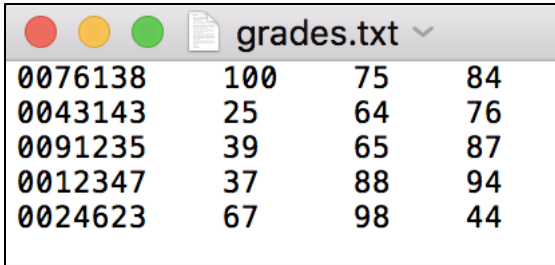
Figure 6-17 General logic for detecting the end of a file



**NOTE:** In this algorithm, we call the `readline` method just before entering the `while` loop. The purpose of this method call is to get the first line in the file, so it can be tested by the loop. This initial read operation is called a *priming read*.

# Example

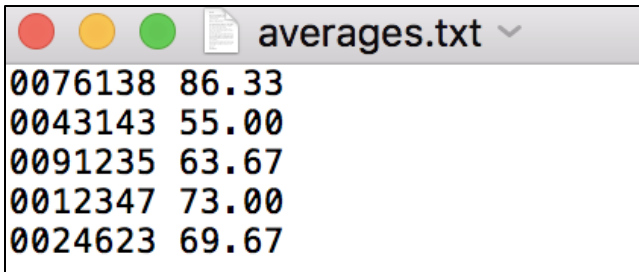
- Suppose that you have the following file that contains the exam grades of students



0076138	100	75	84
0043143	25	64	76
0091235	39	65	87
0012347	37	88	94
0024623	67	98	44

- Write two functions:

- The first function takes the name of this file as its input, reads its contents, and returns a list of student tuples. The first item in each tuple is the student id as a string and the second item is a list of three integer grades. You may assume that the file always contains valid content.
- The second function takes the list of student tuples together with an output file name as its inputs. For each student, it calculates the average of the grades and saves these averages in the output file together with the students ids.



0076138	86.33
0043143	55.00
0091235	63.67
0012347	73.00
0024623	69.67

```
def read_grades(input_file_name):

    infile = open(input_file_name, 'r')
    line = infile.readline()
    L = []
    while line != '':
        line = line.split()
        g1 = int(line[1])
        g2 = int(line[2])
        g3 = int(line[3])
        L.append((line[0], [g1, g2, g3]))
        line = infile.readline()

    infile.close()
    return L

def save_averages(L, output_file_name):
    outfile = open(output_file_name, 'w')
    for i in range(len(L)):
        avg = L[i][1][0] + L[i][1][1] + L[i][1][2]
        outfile.write(L[i][0] + '\t')
        outfile.write('%.2f\n' % (avg/3))

    outfile.close()

def main():
    L = read_grades('grades.txt')
    save_averages(L, 'averages.txt')

main()
```

# Alternative way to open a file: 'with' statement

```
infile = open('info3', 'r')  
l1 = infile.readline()  
l2 = infile.readline()  
l3 = infile.readline()  
infile.close()
```

```
with open('info3', 'r') as infile:  
    l1 = infile.readline()  
    l2 = infile.readline()  
    l3 = infile.readline()
```

`infile.close()` is not required

# Some useful functions to process strings

**rstrip()** : returns a copy of the string with all trailing whitespace characters are removed (*at the end, i.e., Right side*)

- *Useful to get rid of the newline character*

**lstrip()** : returns a copy of the string with all leading whitespace characters are removed (*at the beginning, i.e., Left side*)

**strip()** : returns a copy of the string with all leading and trailing whitespace characters are removed

```
infile = open('info3', 'r')

line = infile.readline()
print(line)
print('next line')
print(line.rstrip())
print('next line')

infile.close()
```

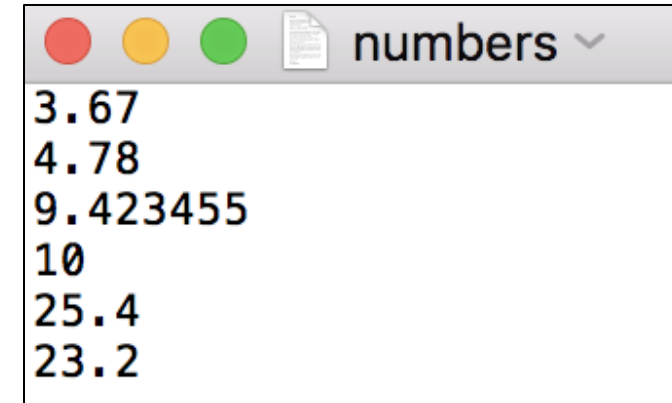
```
In [1]: runfile('/Users/cigdem/Desktop/
10

next line
10
next line
```

# Example

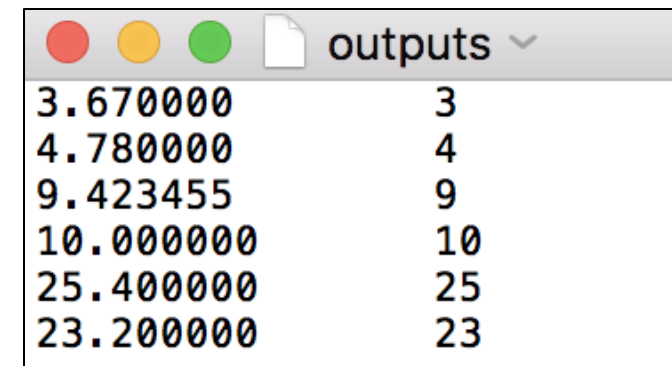
- Write a function that takes two filenames as its inputs. It reads the floating point numbers from the input file, truncates each number, and writes the original number together with its truncated form into the output file.

```
def process_numbers(inname, outname):  
    infile = open(inname, 'r')  
    outfile = open(outname, 'w')  
  
    line = infile.readline()  
    while line != '':  
        num = float(line.rstrip())  
        num_int = int(num)  
        outfile.write('%f\t%d\n' % (num, num_int))  
        line = infile.readline()  
  
    infile.close()  
    outfile.close()  
  
process_numbers('numbers', 'outputs')
```



A screenshot of a text editor window titled "numbers". The window contains the following text:

```
3.67  
4.78  
9.423455  
10  
25.4  
23.2
```



A screenshot of a text editor window titled "outputs". The window contains the following text:

```
3.670000      3  
4.780000      4  
9.423455      9  
10.000000     10  
25.400000     25  
23.200000     23
```



# Back to exceptions

- What happens if your input file does not exist?

```
f1 = open('hello.txt', 'r')  
s = f1.read()  
f1.close()
```

```
In [1]: runfile('/Users/cigdem/Desktop/example6.py', wdir='/Users/cigdem/Desktop')  
Traceback (most recent call last):  
  
  File "/Users/cigdem/Desktop/example6.py", line 1, in <module>  
    f1 = open('hello.txt', 'r')  
  
FileNotFoundError: [Errno 2] No such file or directory: 'hello.txt'
```

# Back to exceptions

- What happens if the path of your output file does not exist?

```
f2 = open('/Users/demir/Desktop', 'w')  
f2.write('My string')  
f2.close()
```

```
In [2]: runfile('/Users/cigdem/Desktop/example6.py', wdir='/Users/cigdem/Desktop')  
Traceback (most recent call last):
```

```
File "/Users/cigdem/Desktop/example6.py", line 2, in <module>  
    f2 = open('/Users/demir/Desktop', 'w')
```

```
FileNotFoundError: [Errno 2] No such file or directory: '/Users/demir/Desktop'
```

# Back to exceptions

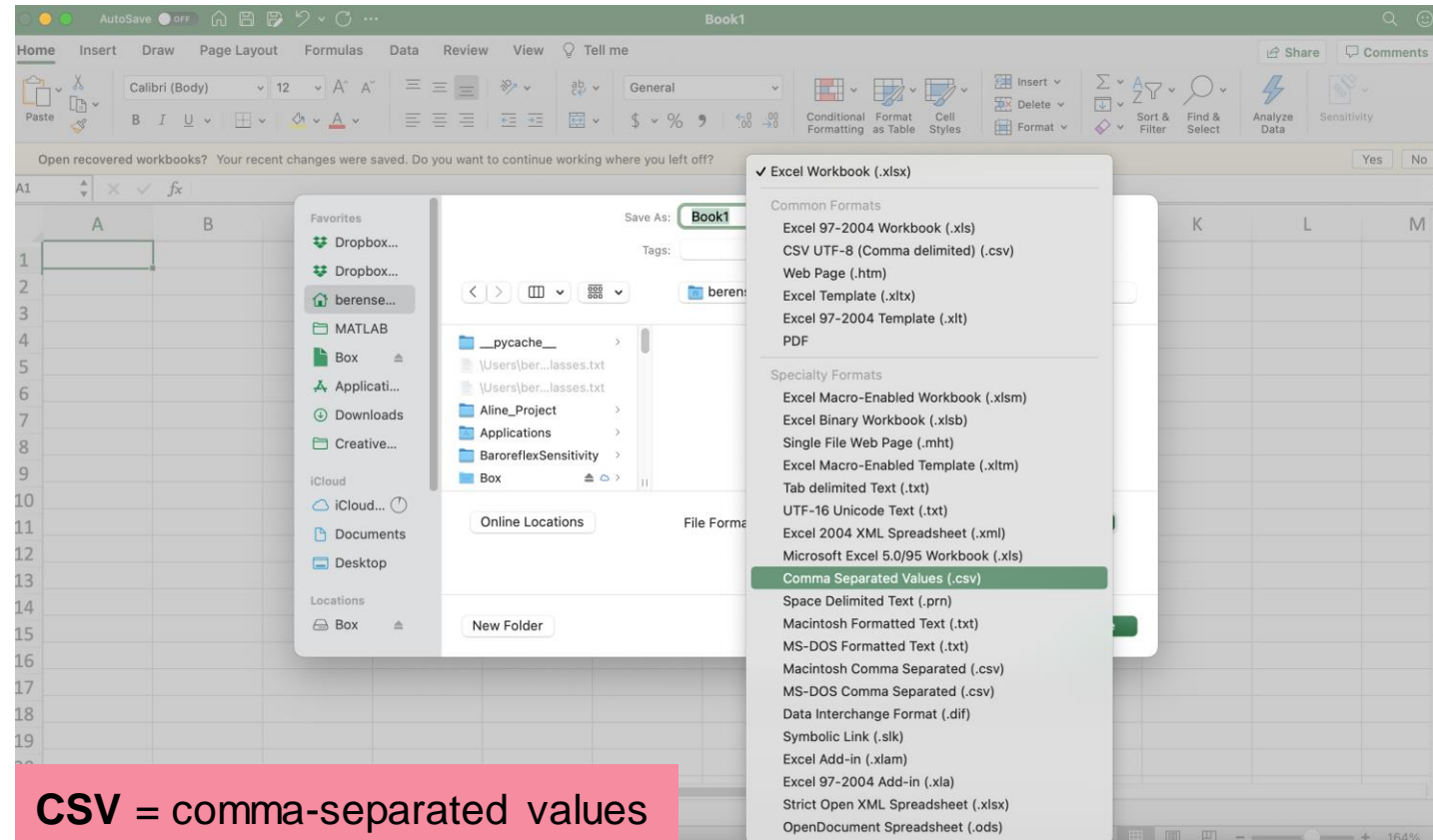
- What happens if your input file does not exist?

```
f1 = open('hello.txt', 'r')  
s = f1.read()  
f1.close()
```

```
try:  
    f1 = open('hello.txt', 'r')  
    s = f1.read()  
    f1.close()  
except FileNotFoundError:  
    print('File does not exist')
```

# Working with CSV files

- CSV file Wikipedia definition
  - “A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.”
- How to edit CSV files?
  - Spreadsheet Apps
    - MS Excel, Apple Numbers, Google Sheets, OpenOffice or LibreOffice
  - Text Editing Apps
    - WordPad, TextEdit, Vim, Emacs, etc.
  - Programming Editor Apps
    - Spyder, PyCharm, VS Code, etc.



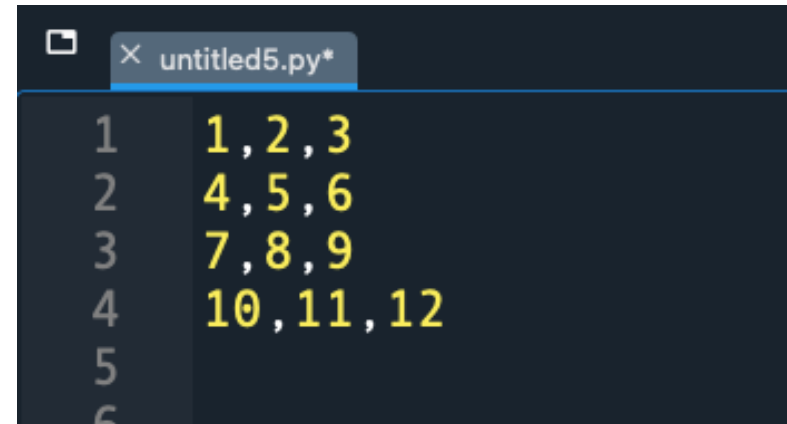
**CSV** = comma-separated values

# CSV files

Spreadsheet view:

	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12

Text editor view:



The screenshot shows a text editor window titled 'untitled5.py\*'. The editor displays a CSV file with 4 rows of data. The first row contains '1, 2, 3', the second row contains '4, 5, 6', the third row contains '7, 8, 9', and the fourth row contains '10, 11, 12'. The text is highlighted in yellow.

```
1 1, 2, 3
2 4, 5, 6
3 7, 8, 9
4 10, 11, 12
```

# Writing data into a .csv file

- Writing into the same column

	A	
1	apple	
2	banana	
3	grapes	
4	pear	
5	watermelon	
6		

```
1 apple
2 banana
3 grapes
4 pear
5 watermelon
6
```

```
my_list=['apple','banana', 'grapes', 'pear', 'watermelon']
new_file = open('fruits.csv','w')
for i in my_list:
    new_file.write(i + '\n')
new_file.close()
```

**If not string, use str  
to convert to a string**

- Writing into the same row

	A	B	C	D	E	
1	apple	banana	grapes	pear	watermelon	
2						
3						

```
1 apple,banana,grapes,pear,watermelon
2
3
```

```
my_list=['apple','banana', 'grapes', 'pear', 'watermelon']
new_file = open('fruits.csv','w')
row = my_list[0]
for i in my_list[1:]:
    row = row + ',' + i
new_file.write(row)
new_file.close()
```

# Reading data from a .csv file

- Reading the items from a column

	A	
1	apple	
2	banana	
3	grapes	
4	pear	
5	watermelon	
6		

```
1 apple
2 banana
3 grapes
4 pear
5 watermelon
6
```

```
new_file = open('fruits.csv','r')
for line in new_file:
    line = line.rstrip()
    print(line)
new_file.close()
```

```
apple
banana
grapes
pear
watermelon
```

- Reading the items from a row

	A	B	C	D	E	
1	apple	banana	grapes	pear	watermelon	
2						
3						

```
1 apple,banana,grapes,pear,watermelon
2
3
```

```
file = open('fruits.csv', 'r')
for line in file:
    line = line.rstrip()
    print(line)
    parts = line.split(',')
    print(parts)
file.close()
```

```
apple,banana,grapes,pear,watermelon
['apple', 'banana', 'grapes', 'pear', 'watermelon']
```

**If you have just one line, you can use `readline()` without a loop**

# csv module

- Writing into the same column

	A	
1	apple	
2	banana	
3	grapes	
4	pear	
5	watermelon	
6		

```
1 apple
2 banana
3 grapes
4 pear
5 watermelon
6
```

```
import csv
my_list=['apple','banana', 'grapes', 'pear', 'watermelon']
file = open('fruits2.csv','w')
writer = csv.writer(file)
for i in my_list:
    writer.writerow([i])
file.close()
```

- Writing into the same row

	A	B	C	D	E	
1	apple	banana	grapes	pear	watermelon	
2						
3						

```
1 apple,banana,grapes,pear,watermelon
2
3
```

```
import csv
my_list=['apple','banana', 'grapes', 'pear', 'watermelon']
file = open('fruits2.csv','w')
writer = csv.writer(file)
writer.writerow(my_list)
file.close()
```



# csv module

- Reading from the csv file

```
import csv
file=open('fruits2.csv', 'r')
reader = csv.reader(file)
for row in reader:
    print(row)
file.close()
```

	A	
1	apple	
2	banana	
3	grapes	
4	pear	
5	watermelon	
6		

```
['apple']
['banana']
['grapes']
['pear']
['watermelon']
```

	A	B	C	D	E	
1	apple	banana	grapes	pear	watermelon	
2						
3						

```
['apple', 'banana', 'grapes', 'pear', 'watermelon']
```

# csv module

**writerow** : One-dimensional

**writerows**: Two-dimensional

```
import csv

my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]

file1 = open('data1.csv', 'w')
writer = csv.writer(file1)
for row in my_list:
    writer.writerow(row)
file1.close()

file1 = open('data2.csv', 'w')
writer = csv.writer(file1)
writer.writerows(my_list)
file1.close()
```

	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12

# csv module

**writerow** : One-dimensional

**writerows**: Two-dimensional

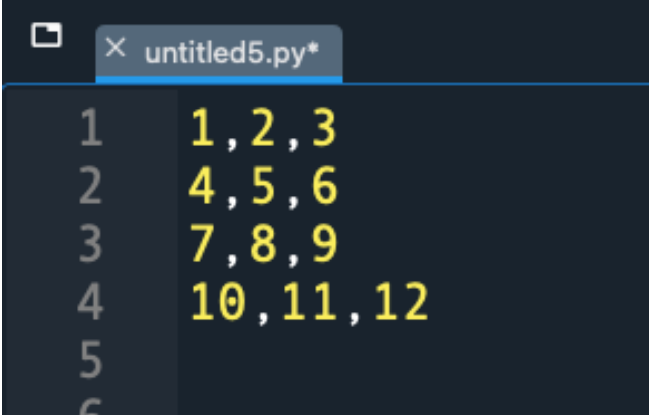
```
import csv

my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]

file1 = open('data1.csv', 'w')
writer = csv.writer(file1)
for row in my_list:
    writer.writerow(row)
file1.close()

file1 = open('data2.csv', 'w')
writer = csv.writer(file1)
writer.writerows(my_list)
file1.close()
```


	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12



```
untitled5.py*
1  1, 2, 3
2  4, 5, 6
3  7, 8, 9
4  10, 11, 12
5
6
```

# Example

- The file **1ubq-mod.pdb** contains atom information for the protein **1ubq**
- From the lines starting with **ATOM** keyword,  
copy the atomid, atom name, and x, y, z coordinates to a csv file
- Note the extra lines at the end of the file

 1ubq-mod.pdb - Notepad

File Edit Format View Help

ATOM	13	CB	GLN	A	2	26.733	30.148	2.905	1.00	14.46		C
ATOM	14	CG	GLN	A	2	26.882	31.546	3.409	1.00	17.01		C
ATOM	15	CD	GLN	A	2	26.786	32.562	2.270	1.00	20.10		C
ATOM	16	OE1	GLN	A	2	27.783	33.160	1.870	1.00	21.89		O
ATOM	17	NE2	GLN	A	2	25.562	32.733	1.806	1.00	19.49		N
ATOM	18	N	ILE	A	3	26.849	29.656	6.217	1.00	5.87		N
ATOM	19	CA	ILE	A	3	26.235	30.058	7.497	1.00	5.07		C
ATOM	20	C	ILE	A	3	26.882	31.428	7.862	1.00	4.01		C
ATOM	21	O	ILE	A	3	27.906	31.711	7.264	1.00	4.61		O
ATOM	22	CB	ILE	A	3	26.344	29.050	8.645	1.00	6.55		C
ATOM	23	CG1	ILE	A	3	27.810	28.748	8.999	1.00	4.72		C

ATOM	601	O	GLY	A	76	38.933	40.525	35.687	0.25	36.13		O
ATOM	602	OX	GLY	A	76	40.862	39.575	36.251	0.25	36.27		O
TER	603		GLY	A	76							
HETATM	604	O	HOH	A	77	45.747	30.081	19.708	1.00	12.43		O
HETATM	605	O	HOH	A	78	19.168	31.868	17.050	1.00	12.65		O
HETATM	606	O	HOH	A	79	32.010	38.387	19.636	1.00	12.83		O
HETATM	607	O	HOH	A	80	42.084	27.361	21.953	1.00	22.27		O
HETATM	608	O	HOH	A	81	21.314	20.644	8.719	1.00	18.33		O
HETATM	609	O	HOH	A	82	31.965	38.637	3.699	1.00	31.69		O
HETATM	610	O	HOH	A	83	27.707	15.908	4.653	1.00	20.30		O

# pdb2csv

```
import csv


#Read the file and close after the with body commands finish
with open("C://Users/msayar/Downloads/1ubq-mod.pdb","r") as f:
    data = f.readlines()

#Store the information here
datalist = []
for line in data:
    if line.startswith('ATOM'):
        line = line.split()
        datalist += [line[1:3]+line[6:9]]

outfile = open("atoms.csv","w")
csvw = csv.writer(outfile, lineterminator='\n')

for i in range(len(datalist)):
    csvw.writerow(datalist[i])

outfile.close()
```

 atoms.csv - Notepad

File Edit Format View Help

```
1,N,27.340,24.430,2.614
2,CA,26.266,25.413,2.842
3,C,26.913,26.639,3.531
4,O,27.886,26.463,4.263
5,CB,25.112,24.880,3.649
6,CG,25.353,24.860,5.134
7,SD,23.930,23.959,5.904
8,CE,24.447,23.984,7.620
9,N,26.335,27.770,3.258
10,CA,26.850,29.021,3.898
```