Programming for Engineers in Python

Fall 2022-2023

I/O

Topics

- Input / Output (I/O, for short)
  - Keyboard input
  - Working with Files

Input/Output in Python

- I/O operations allow our program to interact with its environment by receiving and sending information in various ways.
- The `print` command sends output to the screen.
- The `input` command receives input from the keyboard.
  - Returns a <u>String</u> containing text typed by the user (the program stops when the user presses ENTER).
  - You can specify a prompt message.

input: Receiving input from the keyboard

Try it yourself:
Run the following example using IDLE

```
>>> s = input('Please enter a number:')
Please enter a number: 6
>>> s
"6"
>>> print(int(s)+5)
11
```

File I/O in Python

Working with files

File and Folders

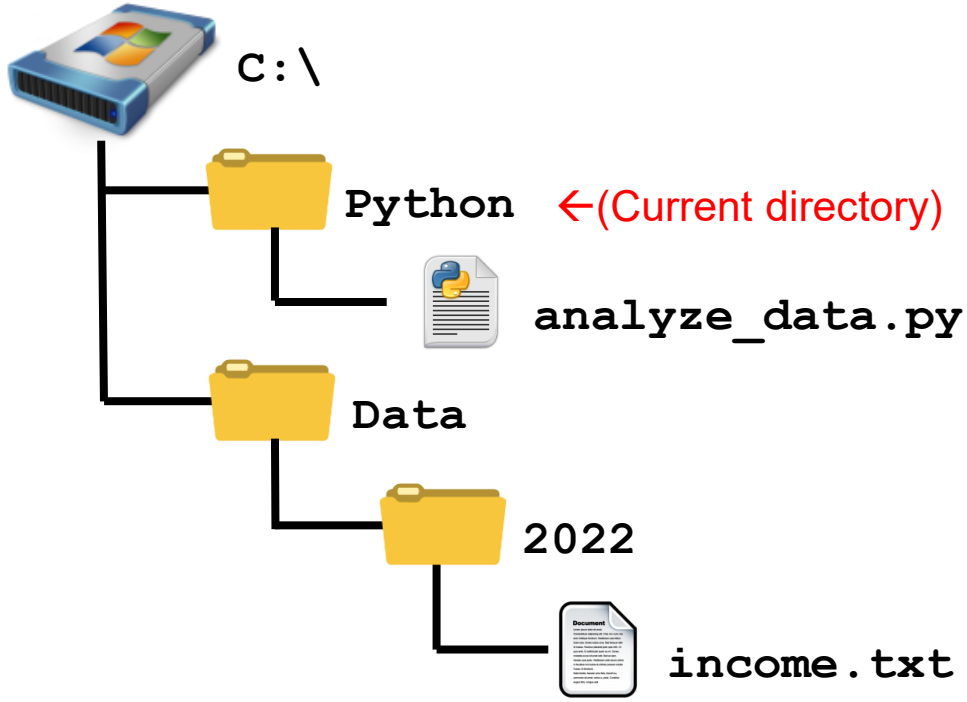
- A **Computer-File** is a resource for storing information; A “digital” document
- Files are organized in folders (=directories).
- Every file has an address (path) in the computer’s file system
  - Example: ***C:\Desktop\ta5_code\test_file.txt***

DO NOT USE NON-ENGLISH PATHS!!!(!!!)

- The file’s extension represents its content type (txt, exe, mp3, avi, jpg)
- We shall next show how to deal with **textual files**.
  - Files can also contain arbitrary, “binary” information (we will not discuss these files though)

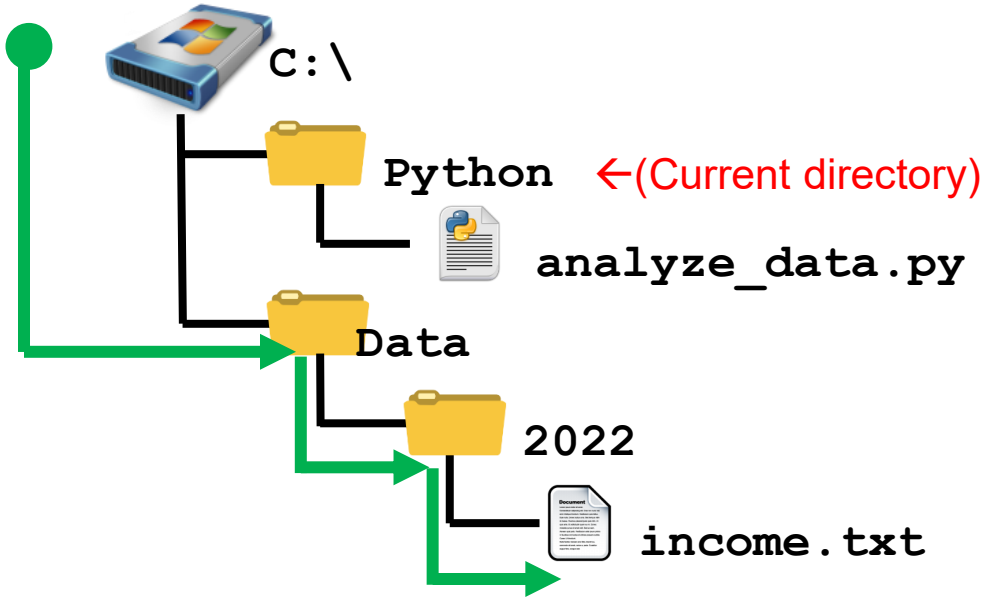
The file system hierarchy

Consider the following file system hierarchy: (windows example)



The file system hierarchy

Consider the following file system hierarchy:

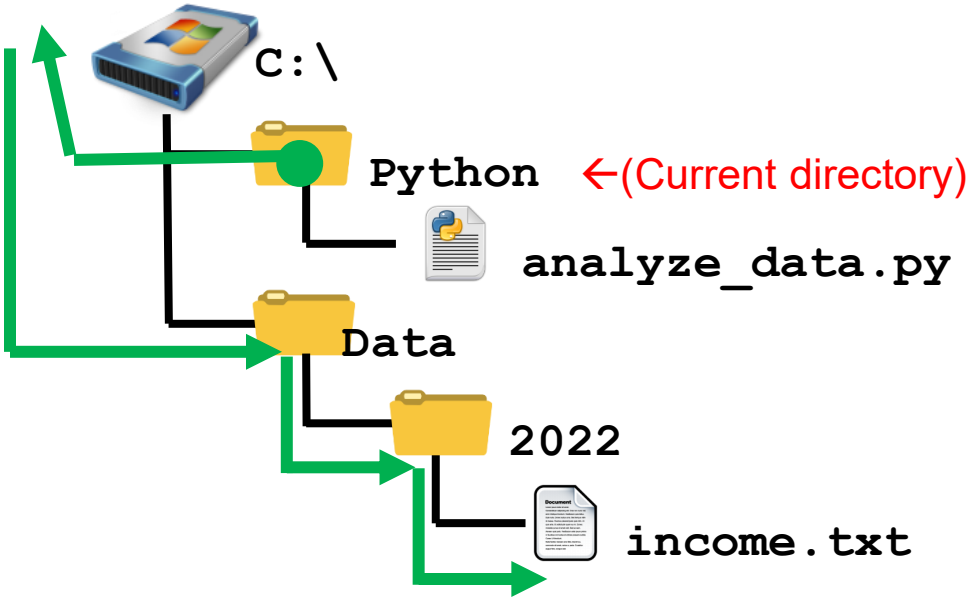


To access the file 'income.txt' from the code inside 'analyze_data.py' we can use two options:

Option 1: **Absolute path:** `C:\Data\2022\income.txt`

The file system hierarchy

Consider the following file system hierarchy:



"Go one level up"

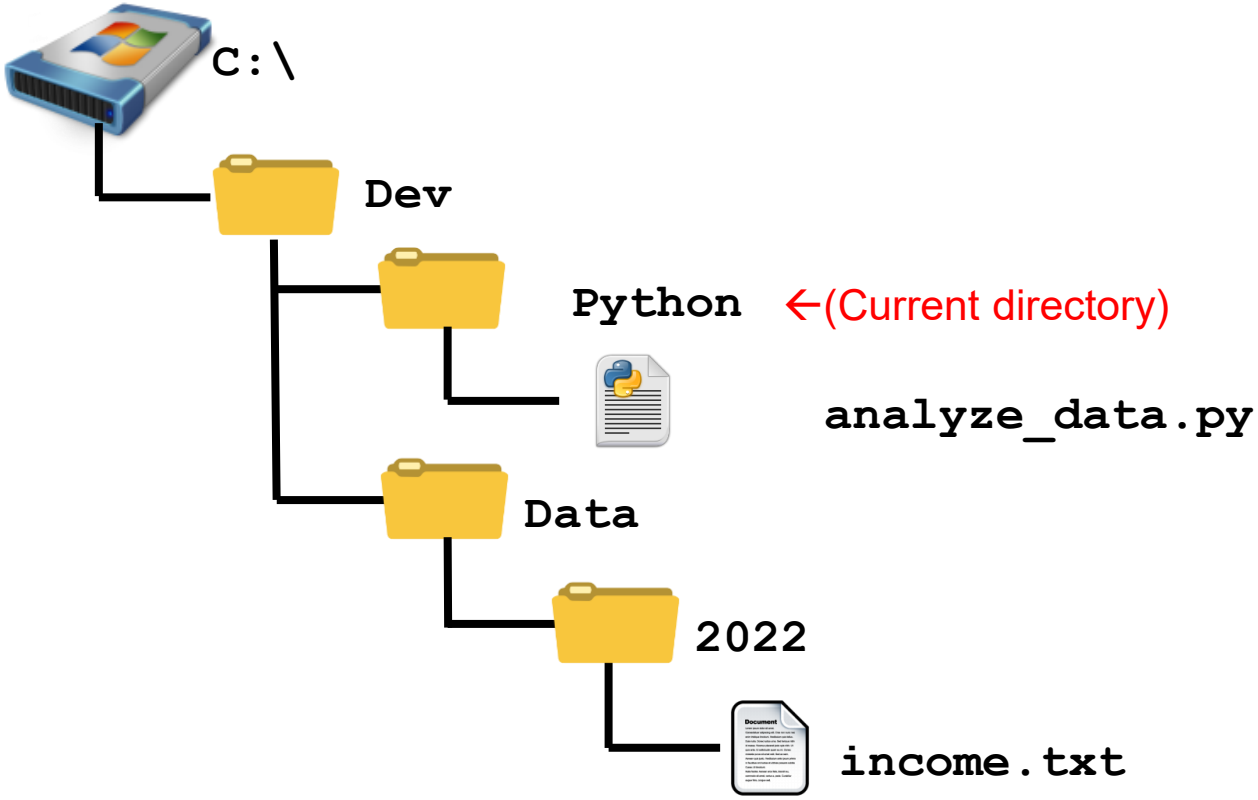
To access the file 'income.txt' from the code inside 'analyze_data.py' we can use two options:

Option 2: **Relative path:** `..\Data\2022\income.txt`

Note that the current directory is 'Python'

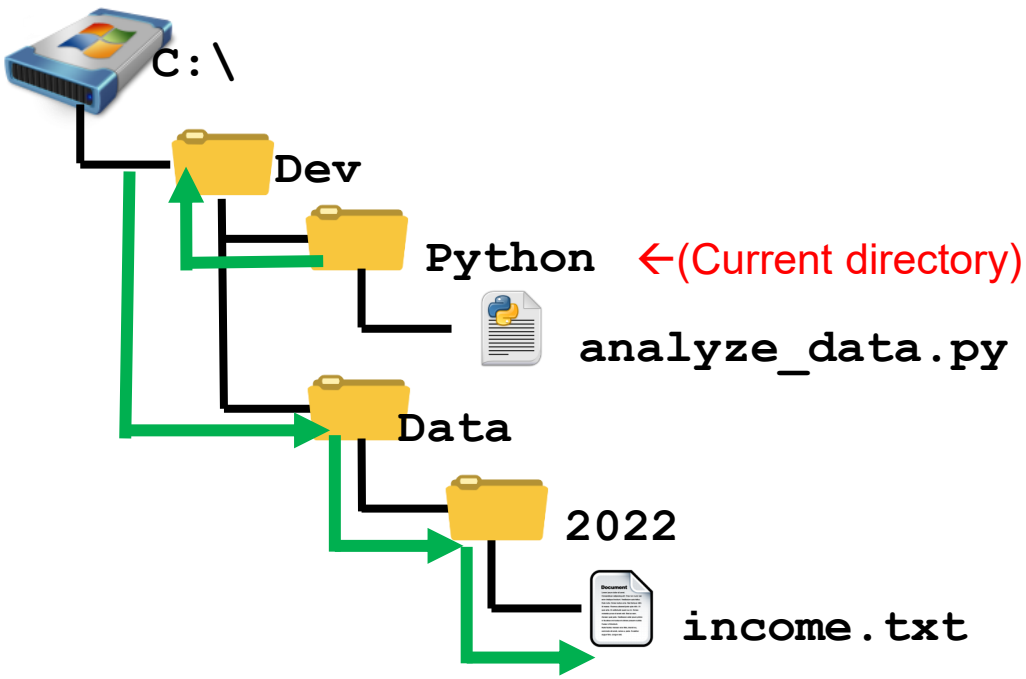
The file system hierarchy

And what would be the **relative path** to access 'income.txt' from the code inside 'analyze_data.py' in the following example?



The file system hierarchy

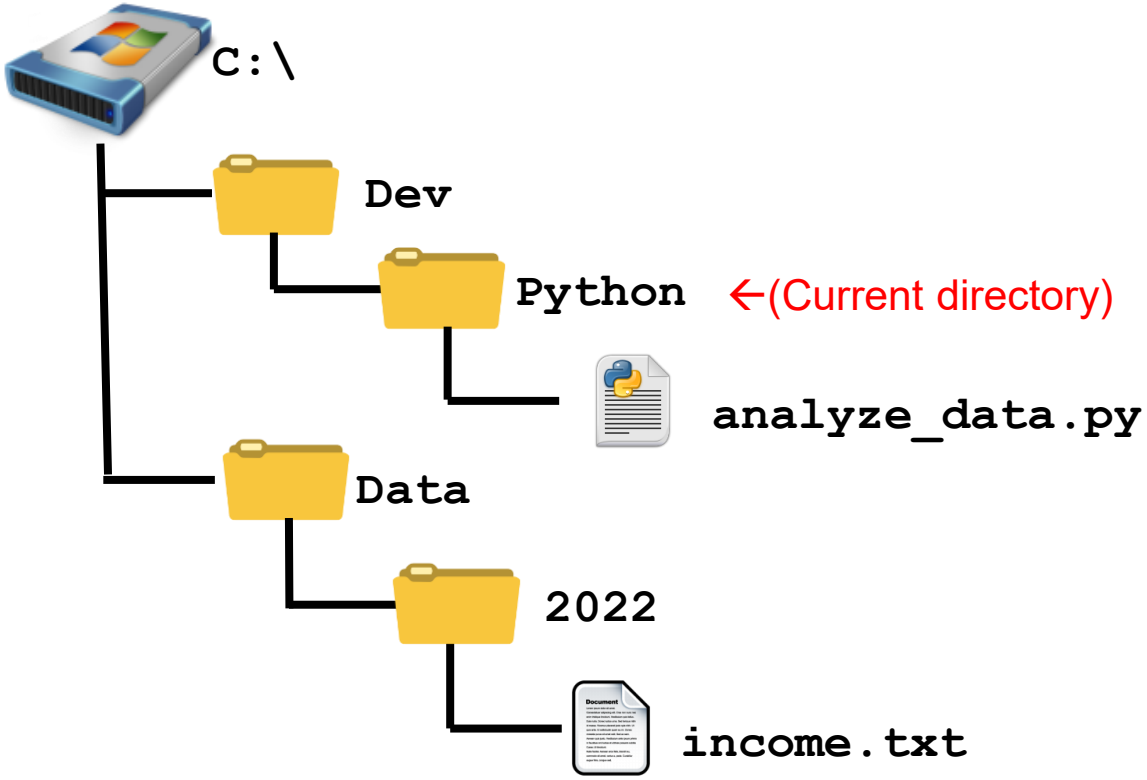
And what would be the **relative path** to access 'income.txt' from the code inside 'analyze_data.py' in the following example?



Answer: **Relative path:** `..\Data\2022\income.txt`
(The same as before!)

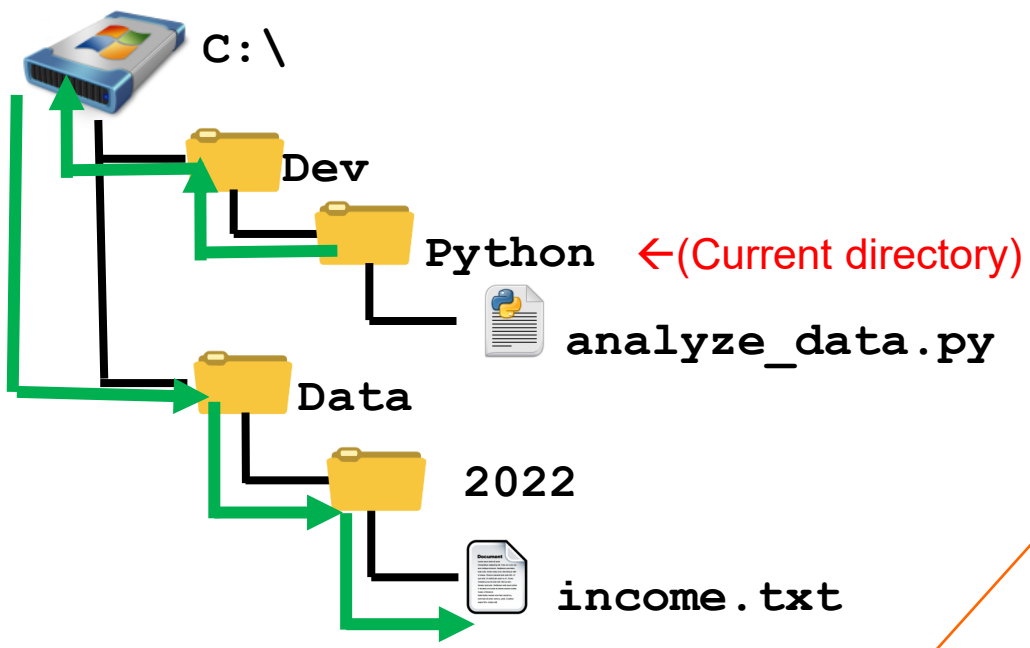
The file system hierarchy

And in this third example?



The file system hierarchy

And in this third example?

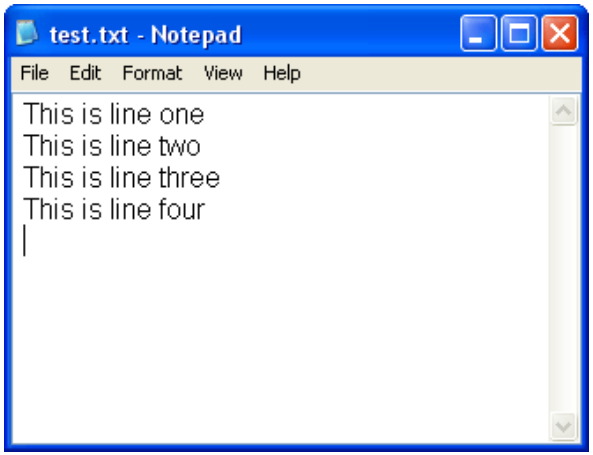


"Go two levels up"

Answer: **Relative path:** `..\..\Data\2022\income.txt`

Text files

- Text files are composed of lines of text.
- Every line is composed of a sequence of characters.
- The last character on every line is the special 'end of line' character ('\n'), usually hidden by most text editors.
- To get rid of trailing newline characters, the string method rstrip() can be used (more details):




```
>>> lines
['This is line one\n', 'This is line two\n']
>>> for i in range(len(lines)):
        lines[i] = lines[i].rstrip()
>>> lines
['This is line one', 'This is line two']
```

Opening a file

- `open()` returns a file object.
- Commonly used with two arguments: `open(filename, mode)`.
  - *filename*: an address of a file (that is, an absolute or relative path to a file)
  - *mode*:
    - 'r' – read
    - 'w' – write - overwrites (“deletes”) prior data
    - 'a' – append - adds at end of prior data

Reading a whole file

Relative path, file is in the same directory

```
>>> f = open('test_file.txt', 'r') # returns a file object
>>> s = f.read() # reads the entire file, returns its content as a string
>>> print(s)
'This is a file'
>>> f.close() # releases the file lock, frees resources

# in case the file is located in a different directory, write an explicit
path to the file
>>> f = open('C:/Users/Desktop/ta5_code/test_file.txt', 'r')

>>> f
<_io.TextIOWrapper name='C:/Users/Desktop/ta5_code/test_file.txt'
mode='r' encoding='cp1252'>
>>> f.read()
'This is a file'
>>> f.close()
```

Path can be absolute or relative

Note about \

```
>>> f = open('C:/Users/Desktop/ta5_code/test_file.txt', 'r')
```

Be careful when using \ in strings representing file paths!
\ is normally used to indicate control characters.
For example, '\n' in a string represents a new line!

Alternatives:

- 'C:/Users/Desktop/ta5_code/test_file.txt'
- 'C:\\Users\\Desktop\\ta5_code\\test_file.txt‘

Reading line by line

- The for loop is used to read line after line from a file object:

```
f = open('test_file_2.txt', 'r')
for line in f:
    print(line, end="")
f.close()
```

The empty string

⟹

```
this is line 1
this is line 2
the end
```

- Storing read lines as a list of strings:

```
f = open('test_file_2.txt', 'r')
lines = []
for line in f:
    lines.append(line)
f.close()
```

Two common ways for reading from a file

- f.**read()** – reads an entire file as single string
- Iterating a file using ***for*** loop:

```
for line in f:
    …
```

Two common ways for reading from a file

```
f = open('test_file.txt', 'r')
```

```
lines =  f.read()
```

```
lines = []
for line in f:
    lines.append(line)
```

```
f.close()  # releases the file lock, frees resources
```

lines = 'this is line 1\n this is line 2\n the end'

lines = ['this is line 1\n', 'this is line 2\n', 'the end']

test_file.txt
this is line 1 this is line 2 the end

Remarks

- To start reading a file from its beginning, we need to re-open it.
- Remember to close the file using **f.close**() to free up resources!
- In a string, '\n' represents the new line character.

```
>>> a_line = 'This is a line\nSecond line'
>>> a_line
'This is a line\nSecond line'
>>> print(a_line)
This is a line
Second line
```

File output: writing to a file

- Use the **f.write(str)**

```
# open a file (or create it if no such file exists)
f = open ('test_file_3.txt', 'w')
f.write("This is a test")
# newline
f.write('\n')

# to write a non-string object, first convert it to string
tpl = (‘my_string', 40)
f.write(str(tpl))

# flush data to the file and close it, unlock file, free resources
f.close()
```

Don’t forget to close the file!

```
f = open('test_file_3.txt', 'r')
print(f.read())

This is a test
(‘my_string', 40)
```

Example: Writing a list of numbers to a text file

```
my_list = [i**2 for i in range(1,11)]
#  List comprehension !
# Generates a list of squares of the numbers 1 – 10

f = open("output.txt", "w")

for item in my_list:
    f.write(str(item) + "\n")

f.close()
```

Don't forget to close the file!

Parsing a text file – **split()**

- **Parsing:** Reading a text file and breaking every line to fields based on some predefined format
- Reminder: The **split** command splits a string to ***tokens*** using a ***delimiter***, returns a ***list*** of strings.

```
s = "topeka, kansas city,wichita,,olathe"

cities = s.split(',')

for city in cities:
     print(city)
```

```
topeka
 kansas city
wichita

olathe
```

If a delimiter is not specified, the string is split to words separated by a sequence of whitespaces.

Parsing a text file – strip(), rstrip(), lstrip()

- Use **strip()** to remove unwanted characters from both sides of the string.

```
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('comwz.')
'example'
```

- The string method **rstrip()** can be used to get rid of trailing newlines (right-strip):

```
lines = ['this is line 1\n', 'this is line 2\n', 'the end']
for i in range(len(lines)):
    lines[i] = lines[i].rstrip()
print(lines)
```

```
['this is line 1', 'this is line 2', 'the end']
```

What does **lstrip()** do?

Exercise 1: Printing word frequencies (Once again)

- Print the words appearing in a file sorted by decreasing frequencies (print most frequent word first)

```
f = open("input.txt", "r")
d = {}
for line in f:
    for word in line.split():
        d[word] = d.get(word, 0) + 1
f.close()

for w in sorted(d.keys(), key = d.get, reverse = True):
    print(w + ":", d[w])
```

Exercise 2: Copy a text file omitting comment lines

Write a function that copies every line from a source file to a target file, excluding lines that start with a '#'

```
def copy_file_excluding_comments(source, target):
    infile  = open(source, 'r')
    outfile = open(target, 'w')
    for line in infile:
        if line[0] == '#':
            continue
        outfile.write(line)
    infile.close()
    outfile.close()
    # What is returned?
```

Reading **a CSV file**

- A CSV file contains data in a tabular format using comma to separate between values
- Each row holds the same number of columns

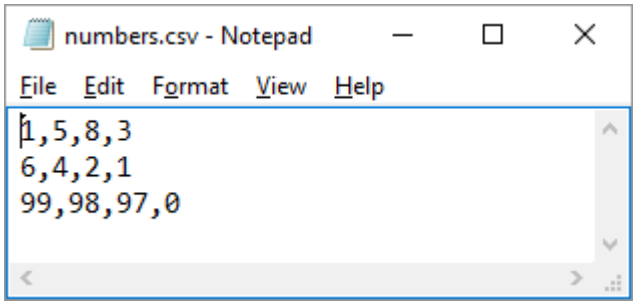
```
0,0,1,3,1,2,4,7,8,3,3,3,10,5,7,4,7,7,12,18,6,13,11,11,7,7,4,6,8,8,4,4,5,7,3,4,2,3,0,0
0,1,2,1,2,1,3,2,2,6,10,11,5,9,4,4,7,16,8,6,18,4,12,5,12,7,11,5,11,3,3,5,4,4,5,5,1,1,0,1
0,1,1,3,3,2,6,2,5,9,5,7,4,5,4,15,5,11,9,10,19,14,12,17,7,12,11,7,4,2,10,5,4,2,2,3,2,2,1,1
0,0,2,0,4,2,2,1,6,7,10,7,9,13,8,8,15,10,10,7,17,4,4,7,6,15,6,4,9,11,3,5,6,3,3,4,2,3,2,1
0,1,1,3,3,1,3,5,2,4,4,7,6,5,3,10,8,10,6,17,9,14,9,7,13,9,12,6,7,7,9,6,3,2,2,4,2,0,1,1
```

Exercise 3: Sum row of numbers read from a CSV file

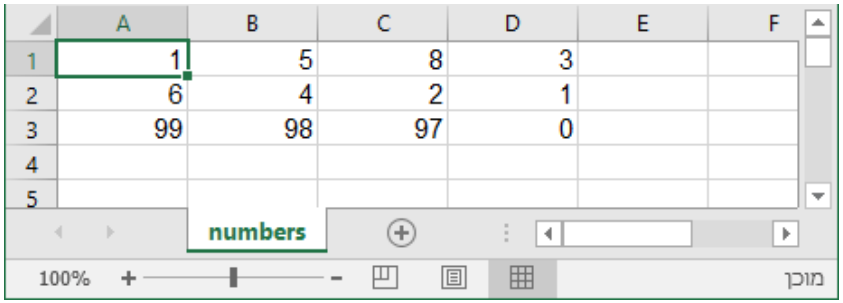
Write a function that sums the numbers in each row for a given CSV file.

Input: CSV file name

Output: A list containing the line sums



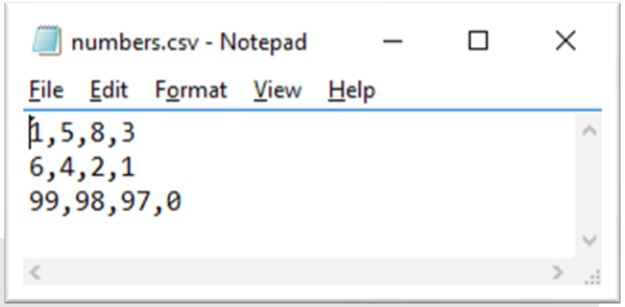
As seen in Notepad



As seen in Excel

```
>>> sum_lines_in_csv_file('numbers.csv')
[17, 13, 294]
```

Exercise 3: Sum row of numbers read from a CSV file




```
def sum_lines_in_csv_file(filename):
    f = open(filename,'r')
    sums = []
    for line in f:
        tokens = line.rstrip().split(',')
        line_sum = 0
        for token in tokens:
            line_sum += int(token)
        sums.append(line_sum)

    f.close()
    return sums
```

```
#Can you compute the sum
#using list comprehension?
```

```
>>> sum_lines_in_csv_file('numbers.csv')
[17, 13, 294]
```

Summary

```
>>> fname = 'myfile.txt'

>>> f = open(fname, 'r')
>>> text = f.read()
>>> f.close()
```

content holds a string of the entire content of ***fname***.

```
>>> f = open(fname, 'r')
>>> for l in f:
...     print(l.split()[1])
>>> f.close()
```

The second word from each line in ***fname*** is printed.

```
>>> outfname = 'myoutput.txt'
>>> outf = open(outfname, 'w')
>>> outf.write('My very own file\n')
>>> outf.close()
```

Creates a new txt file (or overwrites, if a file with same name and path exists) with one line in it:
My very own file