CSCI 141

Scott Wehrwein

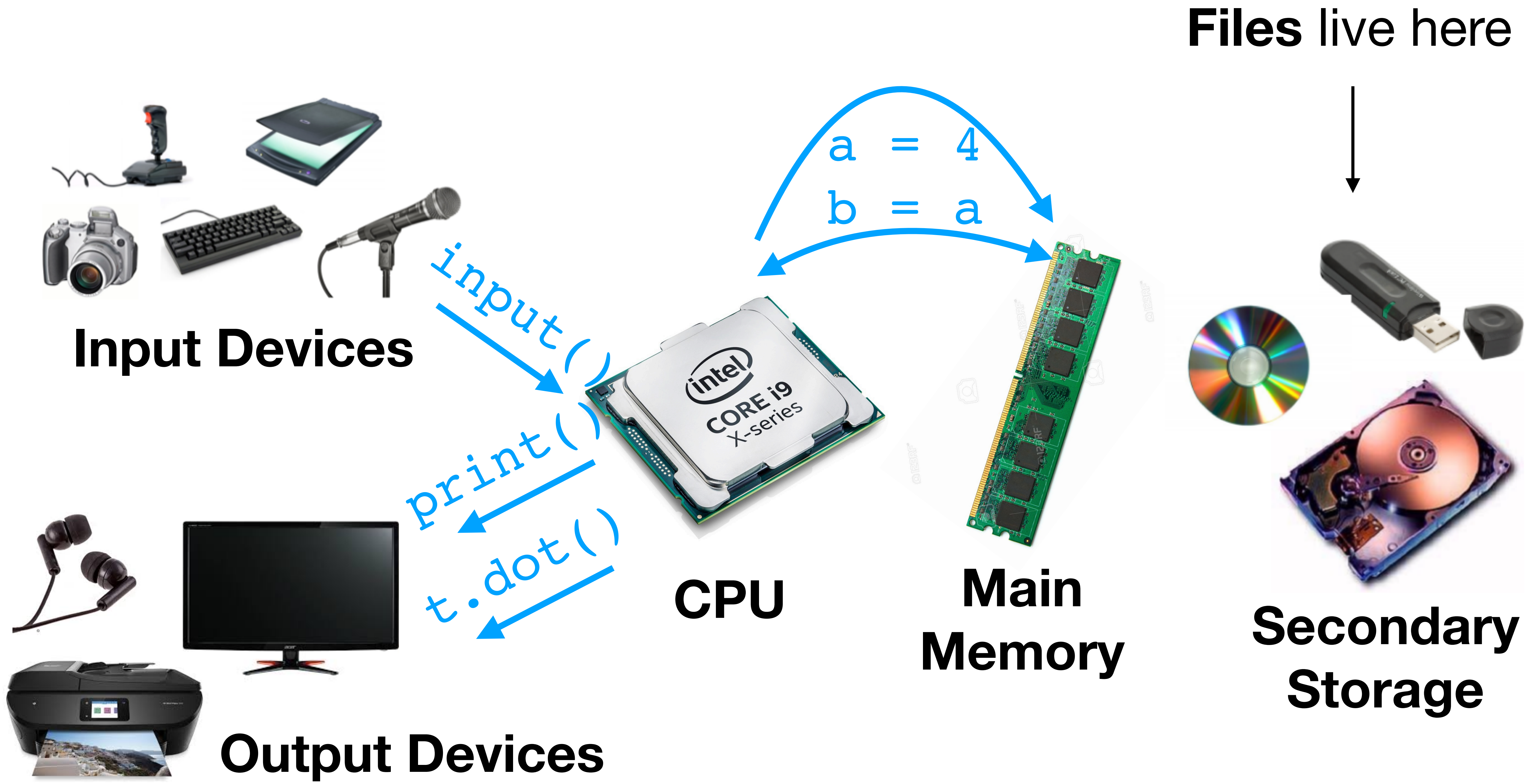
Reading and Writing Files

Goals

- Know the basics of file input/output:
  - Reading and seeking - iterating over lines, `read`, `readlines`, `seek`
  - Writing - `write` method

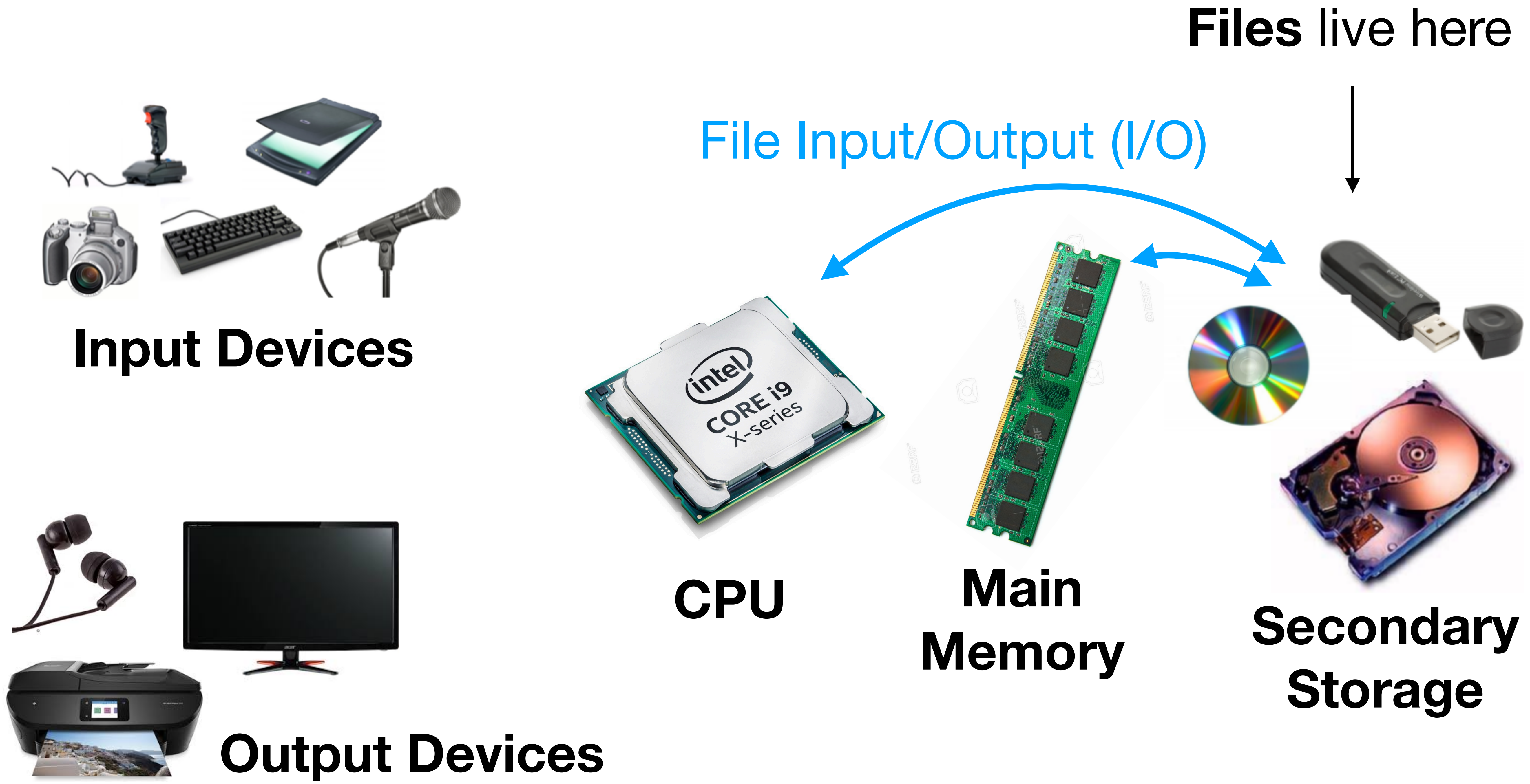
A blast from the past:

A simple model of a computer:



A blast from the past:

A simple model of a computer:



Files - Opening, Reading

"Comma-Separated Values"

Here's some code to read lines from a CSV file like the ones you'll use for the final project:

path to the file (str)

File object

open file for reading (vs. writing)

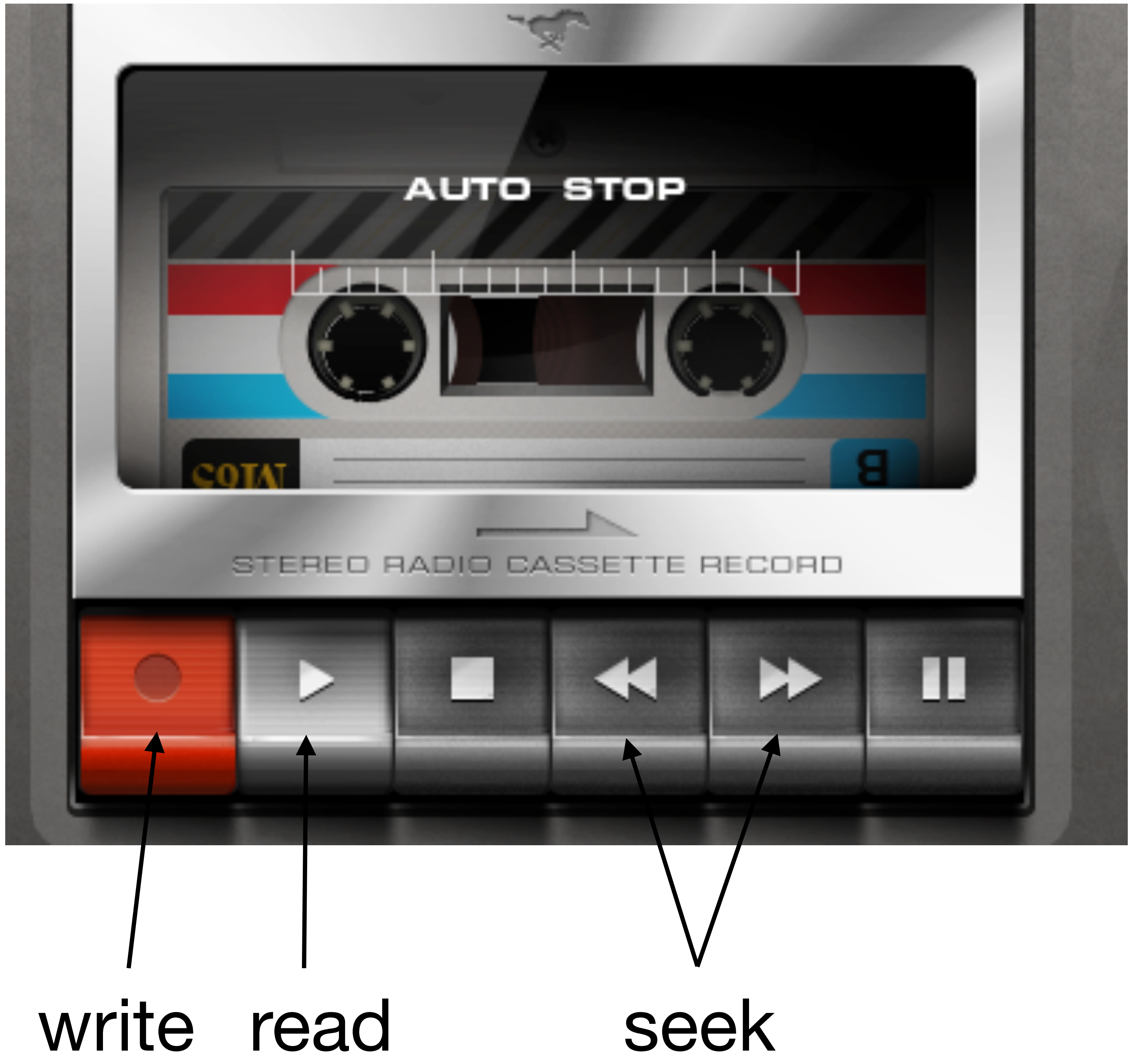
```
input_file = open(filename, "r")
for line in input_file:
    line = line.strip("\n")
    line_list = line.split(",")
```

you can **iterate** over a File object (!)

split line on commas

familiar string stuff

File objects



Reading files: demo

Reading files: demo

```
fobj = open("test.txt", "r")

fobj.read(1)

fobj.read(5)

fobj.tell()

fobj.seek(0)

fobj.read(10)

fobj.read(10)

fobj.readline()

fobj.readline()

fobj.close()
```

```
fobj = open("test.txt", "r")

for line in fobj:

  print(line)


for line in fobj:

  print(line.strip("\n")
```

A More Practical Example

Let's the data from the About You Survey!

csv_demo.py

Writing files

opens the file for writing
deletes any existing contents!

```
output_file = open(filename, "w")

output_file.write("a string\n")
```

`write` doesn't behave like `print`: it writes exactly the string you give it, with no implicit newlines or spacing

Writing files: Demo

opens the file for writing
deletes any existing contents!

file_read_write.py

```
output_file = open(filename, "w")

output_file.write("a string\n")
```

`write` doesn't behave like `print`: it writes exactly the string you give it, with no implicit newlines or spacing