

# CSCI 141



Lecture 24  
Reading and Writing Files

# CSCI 141



## Lecture 24 Reading and Writing Files

# Announcements

# Announcements

- No labs next week.  
Extra TA office hours instead:

# Announcements

- No labs next week.  
Extra TA office hours instead:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)

# Announcements

- No labs next week.  
Extra TA office hours instead:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)
  - Rory 12-2 Monday and Tuesday (CF 477)

# Goals

- Know the basics of file input/output:
  - Reading and seeking - iterating over lines, `read`, `readlines`, `seek`
  - Writing - `write` method
- Know how to use the convenient string methods `split` and `join`

# QOTD

```
def z4(d1, d2):  
    a = d1  
    d1 = {}  
    d1 = d2  
    d1["A"] = 2  
    return a  
  
a = {"A": 4, "B": 6}  
b = {"A": 6, "B": 11}  
f = z4(a, b)  
print(a["A"], b["A"], f["A"])
```

4 2 4



# A blast from the past:

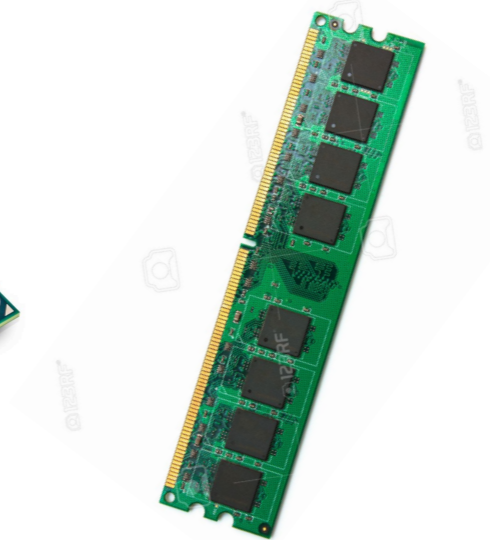
A simple model of a computer:



**Input Devices**



**CPU**



**Main  
Memory**



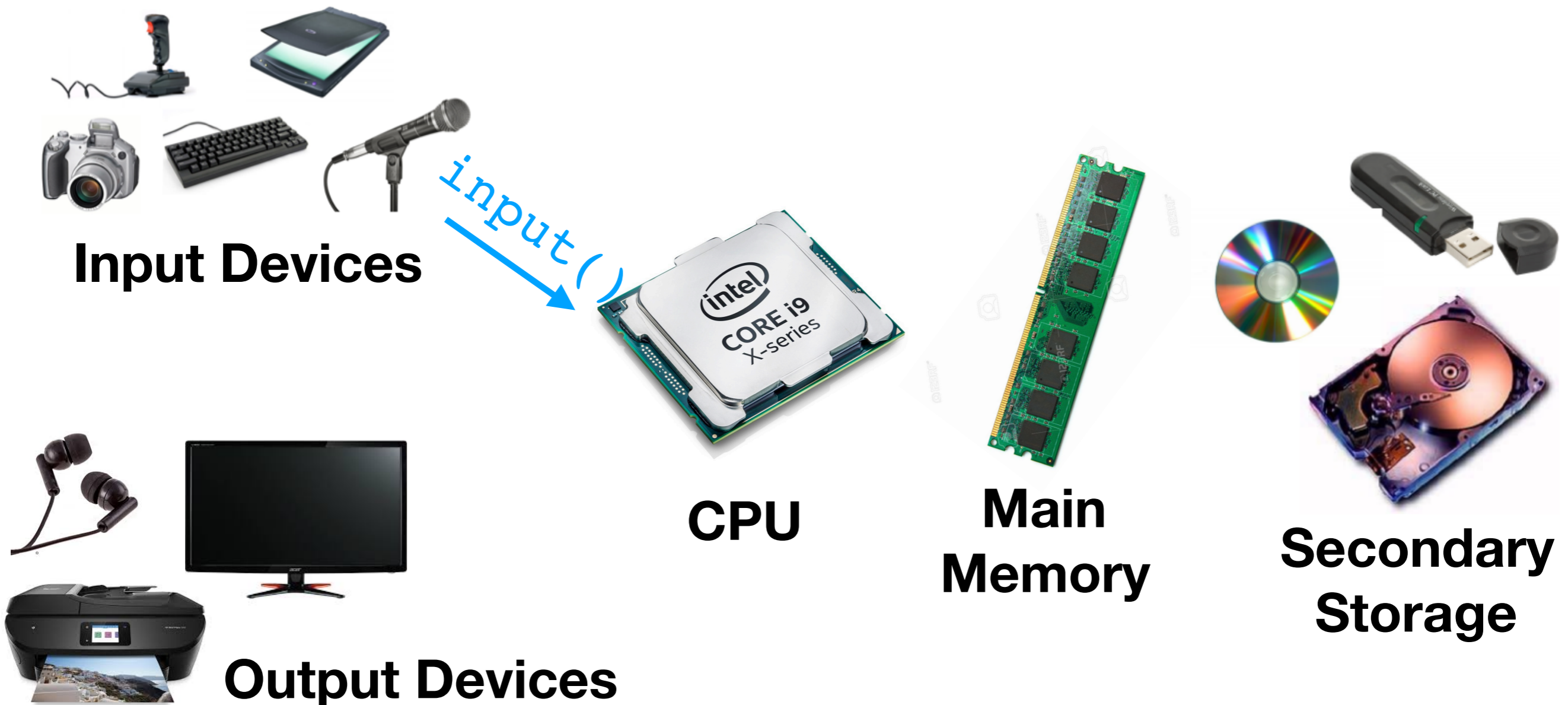
**Secondary  
Storage**



**Output Devices**

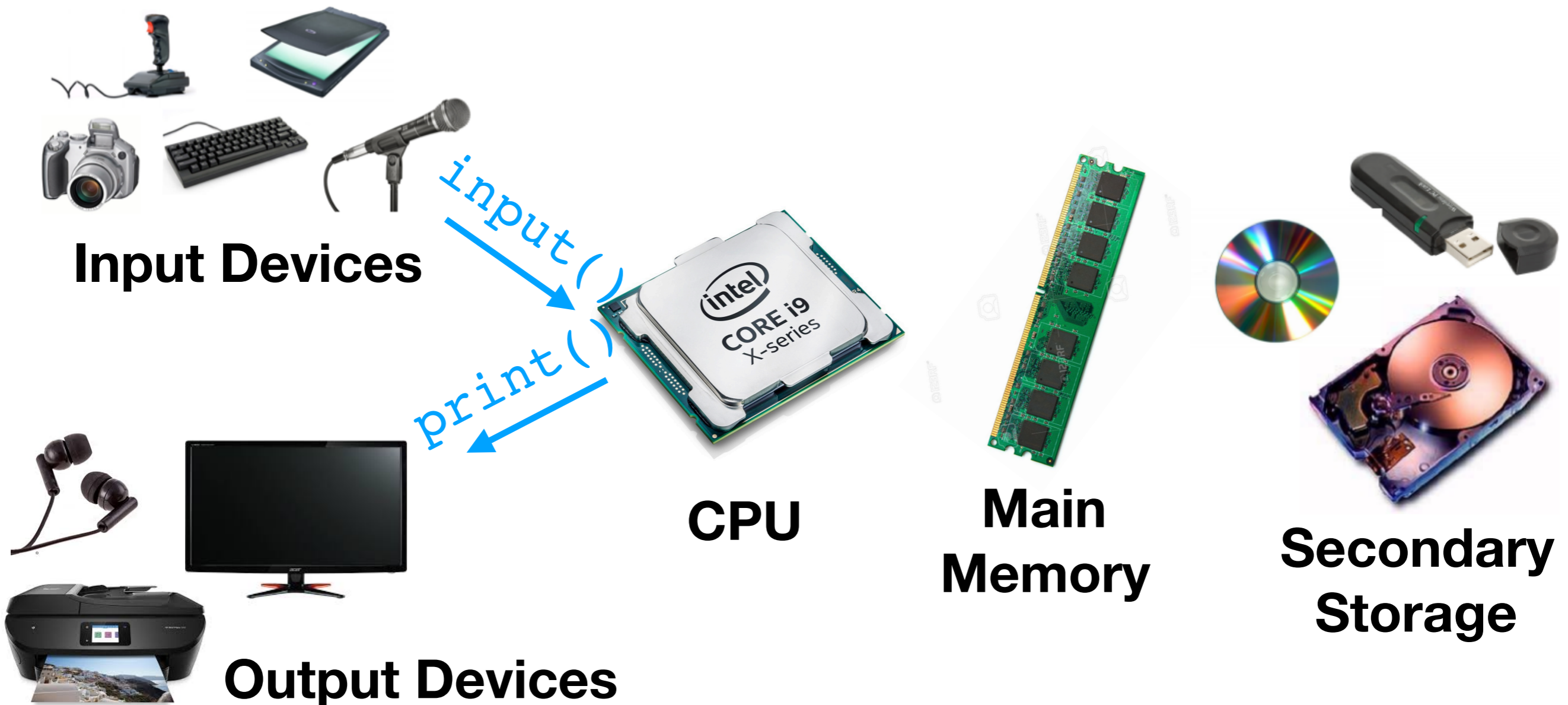
# A blast from the past:

A simple model of a computer:



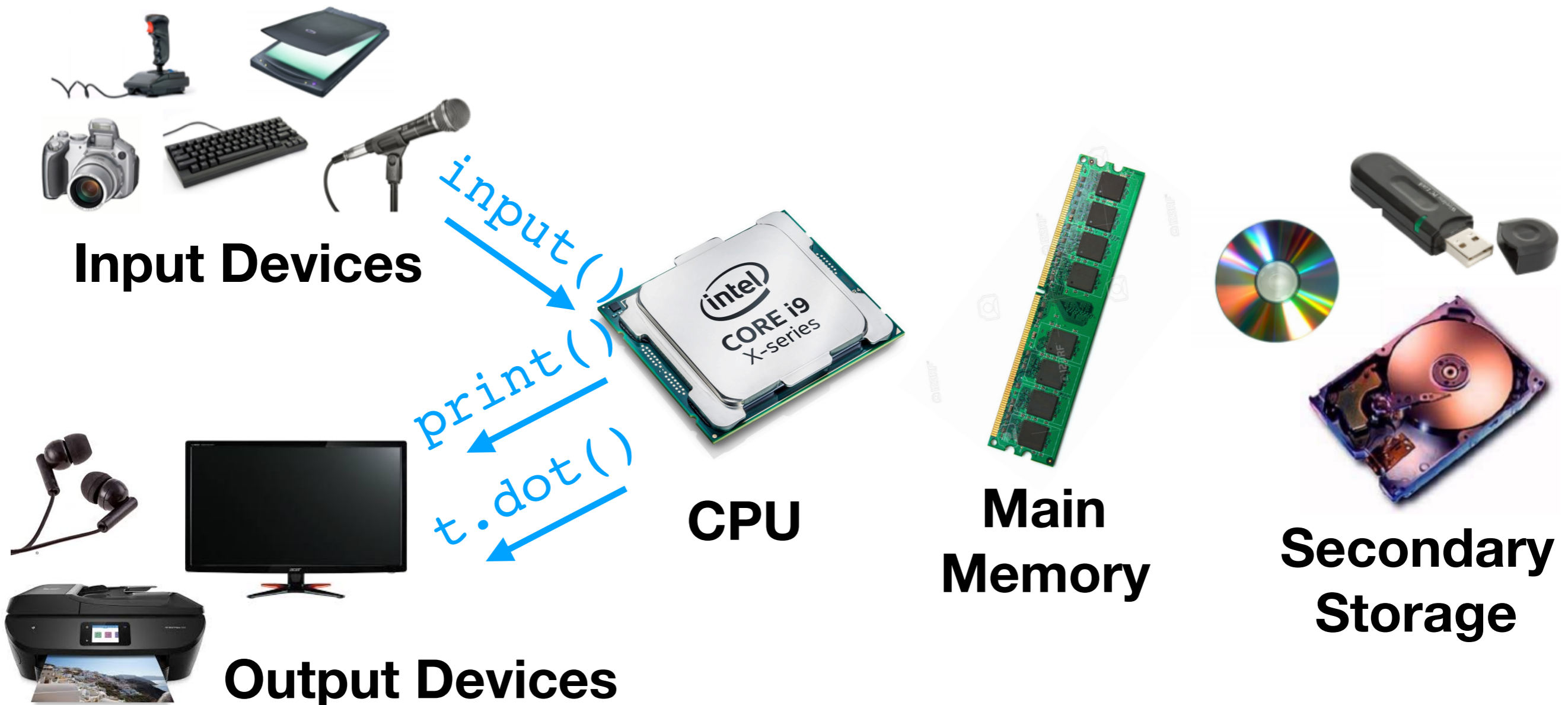
# A blast from the past:

A simple model of a computer:



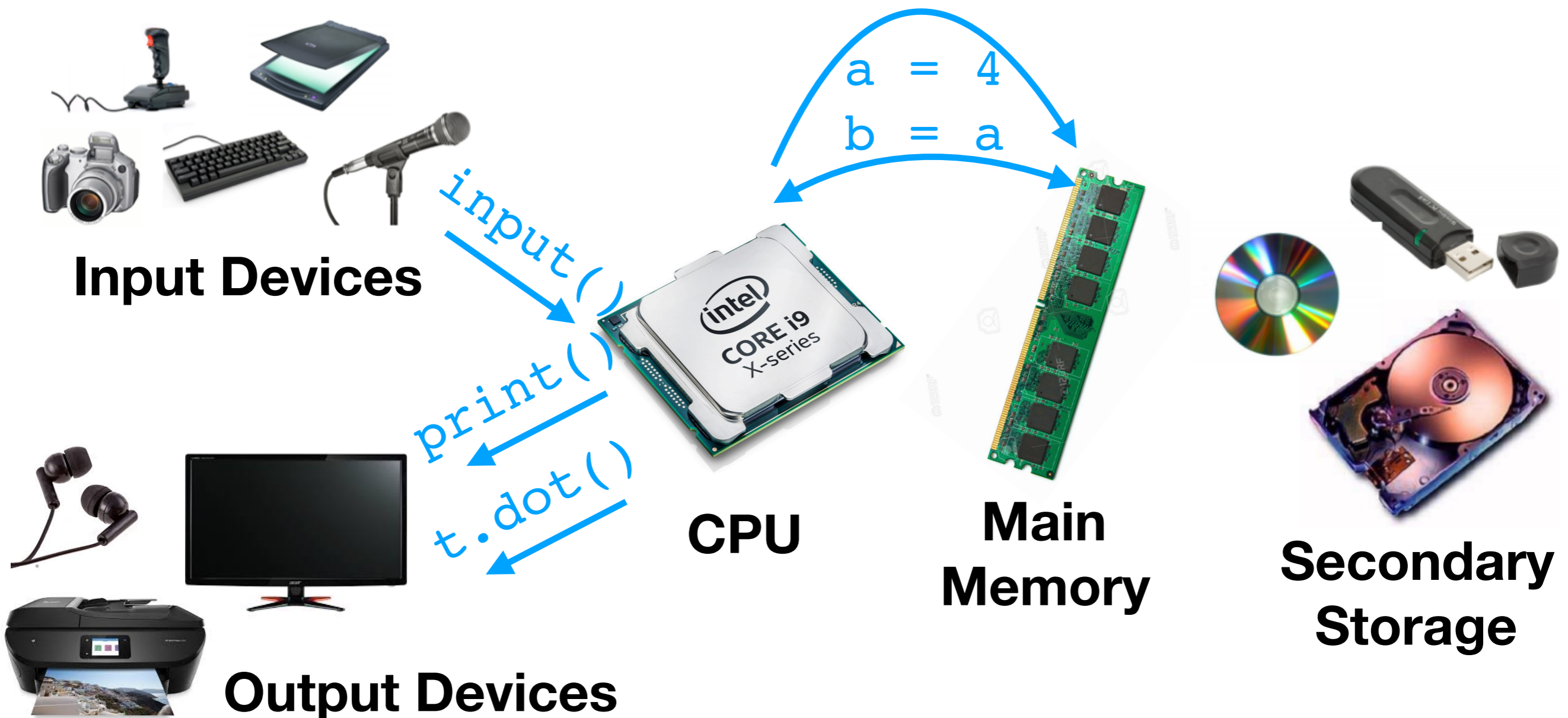
# A blast from the past:

A simple model of a computer:



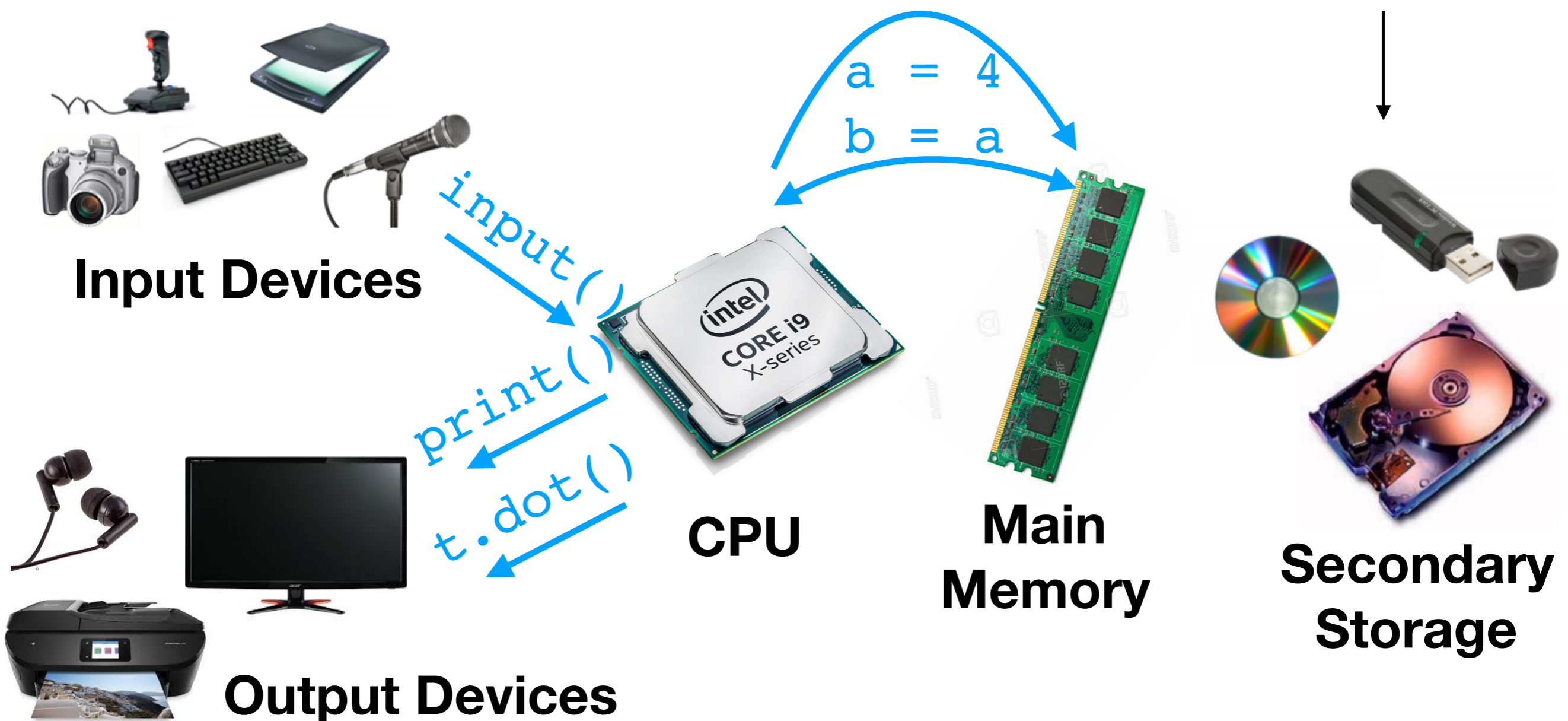
# A blast from the past:

A simple model of a computer:



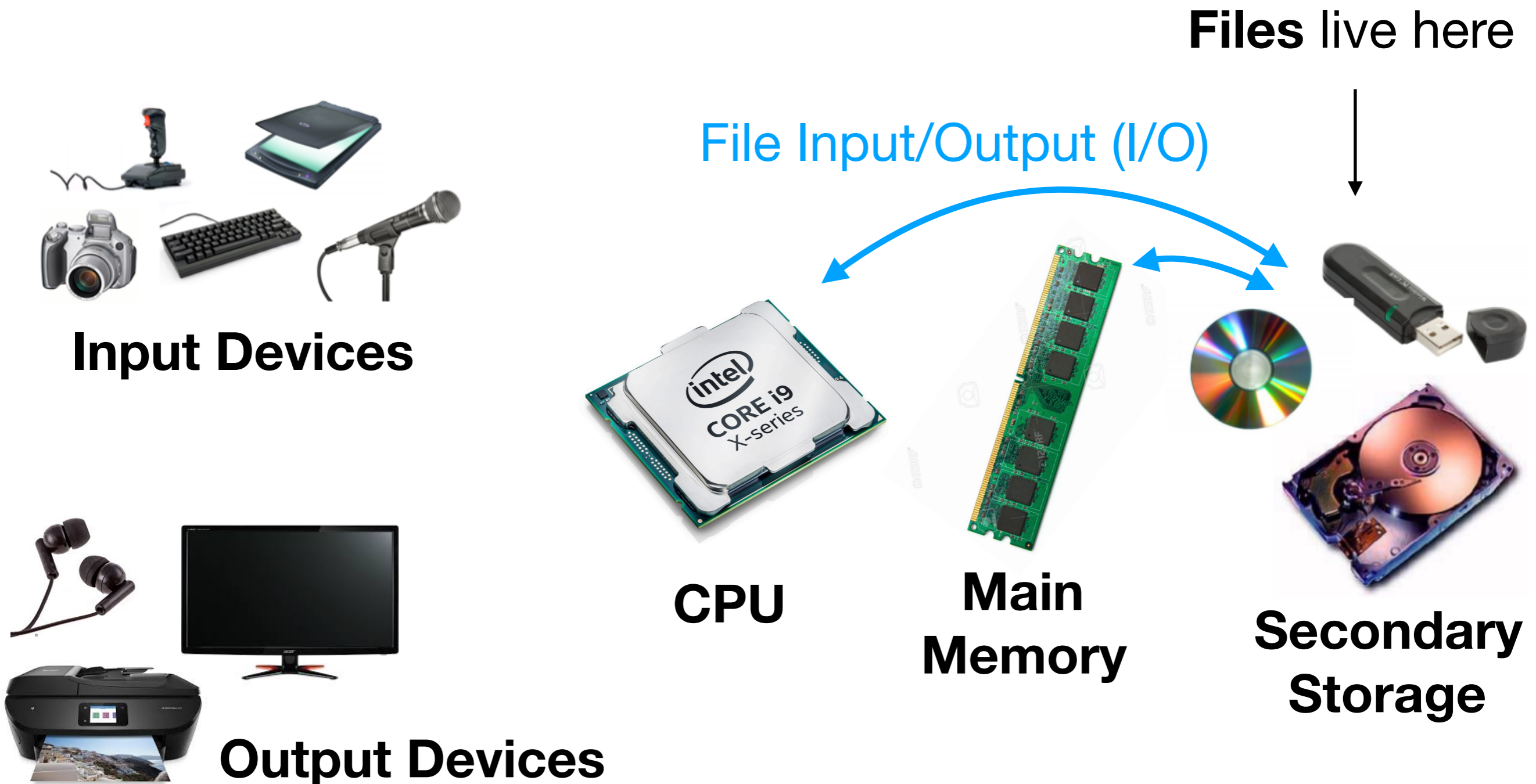
# A blast from the past:

A simple model of a computer:



# A blast from the past:

A simple model of a computer:



# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

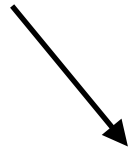
```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```



# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

File object



```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

File object

path to the file (str)

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

File object

path to the file (str)

open file for reading (vs. writing)

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

File object

path to the file (str)

open file for reading (vs. writing)

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

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

File object

path to the file (str)

open file for reading (vs. writing)

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

familiar string stuff

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

File object

path to the file (str)

open file for reading (vs. writing)

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

what does split do?

familiar string stuff

# String's `split` method

```
string.split(separator_string)
```

Splits the string into a list on a given separator, or all whitespace by default. It "eats" the separators.

# String's `split` method

```
string.split(separator_string)
```

Splits the string into a list on a given separator, or all whitespace by default. It "eats" the separators.

```
a = "This is a sentence."  
b = "4.5, 6.8, 82.3"  
c = """This is a string with \t weird
```

```
whitespace"""
```

```
a.split() # on all whitespace
```

```
c.split() # on all whitespace
```

```
c.split(" ") # on spaces only
```

```
b.split() # commas remain
```

```
b.split(",") # spaces remain
```

```
b.split(", ") # just the values
```



# String's `join` method

```
string.join(list_of_strings)
```

Joins its argument's elements into a single string, separated by the string that `join` was called on.

```
a = [1, 2, 3, 4]
" ".join(a) # error - not a list of strings

# enumerate gives you pairs of (index, value):
for i, v in enumerate(a):           (enumerate - useful, but not on the exam)
    a[i] = str(v)

" ".join(a) # => "1 2 3 4"
" one thousand ".join(a)
# => "1 one thousand 2 one thousand 3 one thousand 4"
```

# String's `join` method

```
string.join(list_of_strings)
```

Joins its argument's elements into a single string, separated by the string that `join` was called on.

```
a = [1, 2, 3, 4]
" ".join(a) # error - not a list of strings

# enumerate gives you pairs of (index, value):
for i, v in enumerate(a):           (enumerate - useful, but not on the exam)
    a[i] = str(v)

" ".join(a) # => "1 2 3 4"
" one thousand ".join(a)
# => "1 one thousand 2 one thousand 3 one thousand 4"
```

# File objects



write

read

seek

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

File object

path to the file (str)

open file for reading (vs. writing)

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

what does split do?

familiar string stuff

# Files - Opening, Reading

In A5, I provided the code to read the training and test data from a file (lightly edited):

File object

path to the file (str)

open file for reading  
(vs. writing)

```
input_file = open(filename, "r")
for line in input_file:
    if "#" not in line:
        line = line.strip("\n")
        line_list = line.split(",")
```

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

what does split do?

familiar string stuff

# Reading files: demo

# Reading files: demo

- `file_read.py`

# Writing files

```
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

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

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

# Writing files: Demo

file\_read\_write.py

opens the file for writing  
deletes any existing contents!

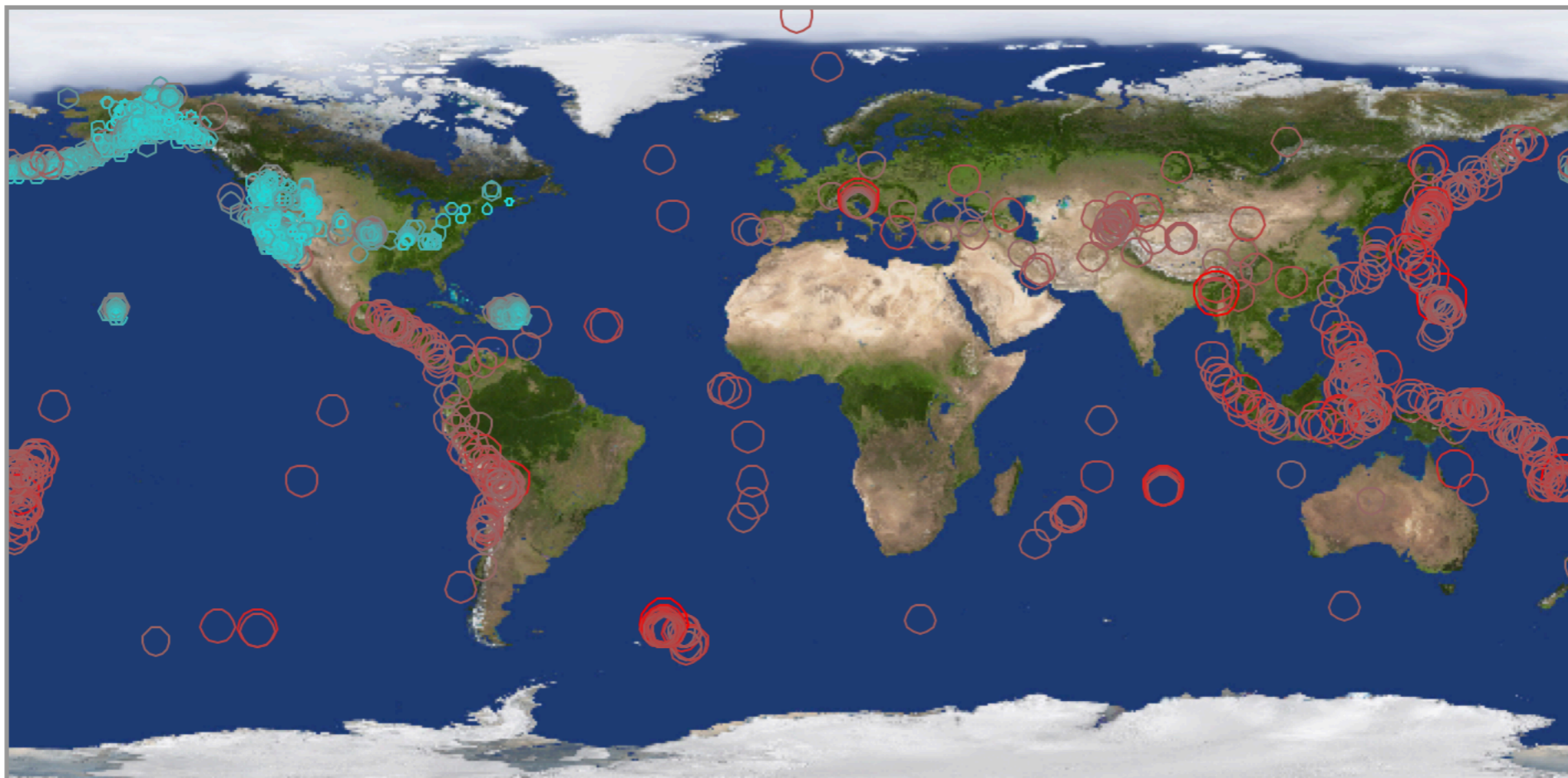
```
output_file = open(filename, "w")
```

```
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

# Reading files: why is this cool?

- You can now play with some big data:
  - A5, for example.
  - Another example - Lab 8:  
Make this map plotting locations and magnitudes of earthquakes



Suppose the file rick.txt contains:

```
Never gonna give you up
```



What is the output of the following code?

```
print(open("rick.txt", "r").read(5).split("e"))
```

A. Nvr

B. Never

C. ["N", "e", "v", "e", "r"]

D. ["N", "v", "r"]

**What can we do with this?**

# grep

```
def grep(string, filename):  
    """ Print all lines of the file filename  
    that contain the given string.  
    Precondition: the file exists. """
```

# split an address

```
def split_address(addr_line):  
    """ Split the postal address in address_line into its  
    component pieces. Return a tuple of strings containing:  
        (number, street, city, state, zip).  
    Precondition: the address matches the following format:  
        "<number> <street>, <city> <state> <zip>"  
    Example: split_address("516 High St, Bellingham WA 98225")  
    => ("516", "High St", "Bellingham", "WA", "98225")  
    """
```



# write a spellchecker

```
def spellcheck(in_filename, out_filename, wordlist):  
    """ Write a spellchecked version of in_filename to  
    out_filename. For each word in the input file, write  
    it as-is to the output file if it is in the wordlist;  
    otherwise, write it to the output file in ALLCAPS to  
    indicate that it's not in the wordlist. """
```