CSCI 127: Joy and Beauty of Data

Lecture 2: Computers, Computer Science, and Ratatouille

Note: You will **not** be tested on this

Reese Pearsall

Snowmester 2020

https://reesep.github.io/classes/127/main.html

What is a computer?

What is a computer?

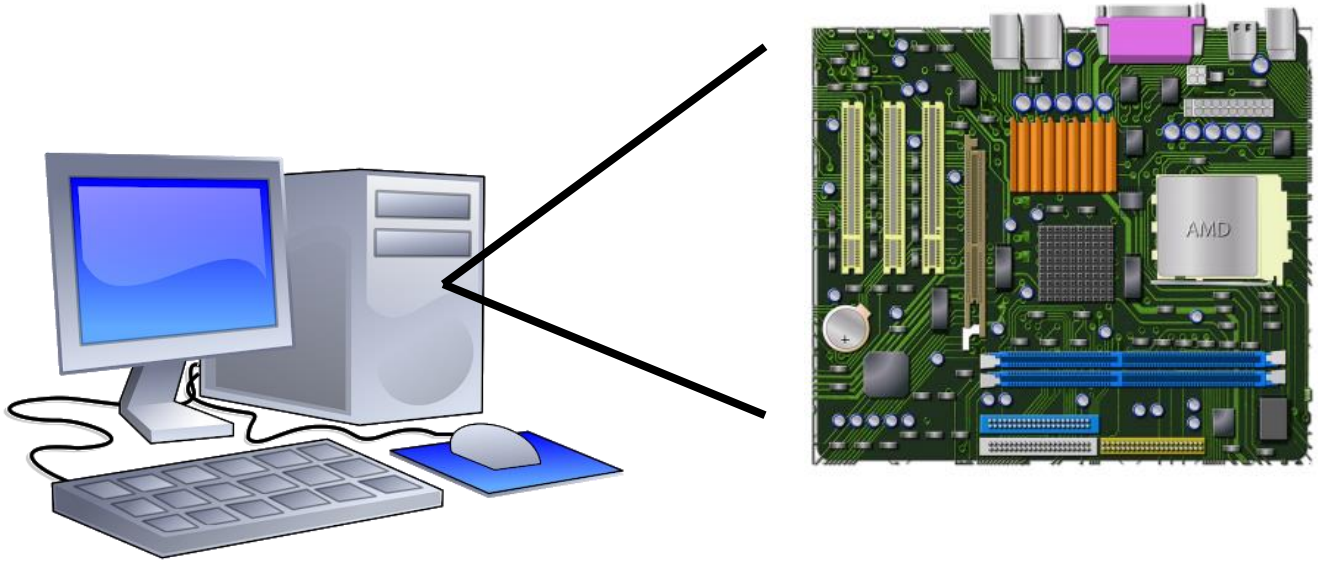
A magical box that gets stuff dun

What is a computer?

Better answer:

semi- executes instructions

A magical box that ~~gets stuff dun~~

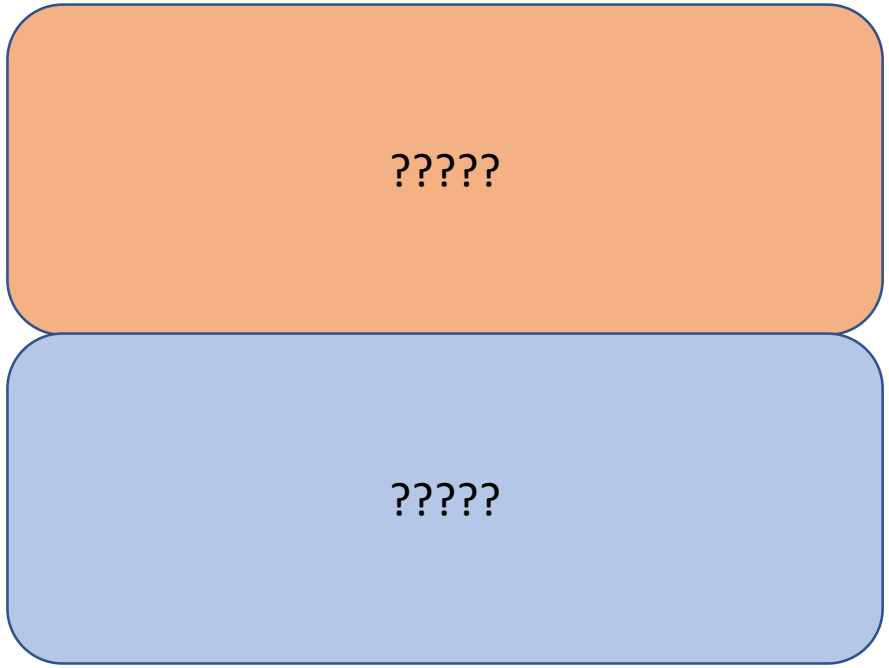


What is magical about it?

What is magical about it?

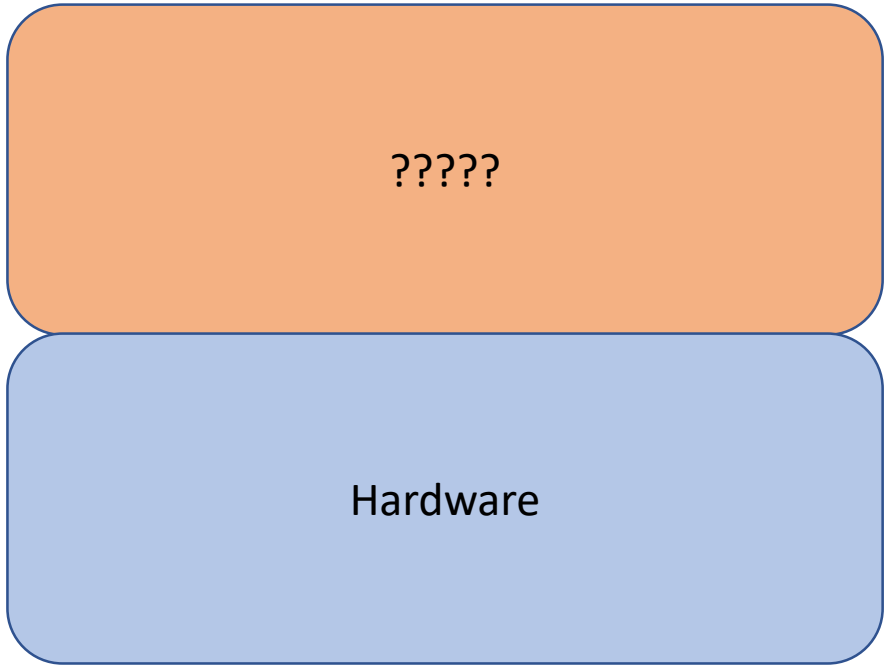
Big Idea: Computers can only understand instructions in the form of 0s and 1s (binary)

How does this happen?



From a high level, we will divide a computer system into two parts

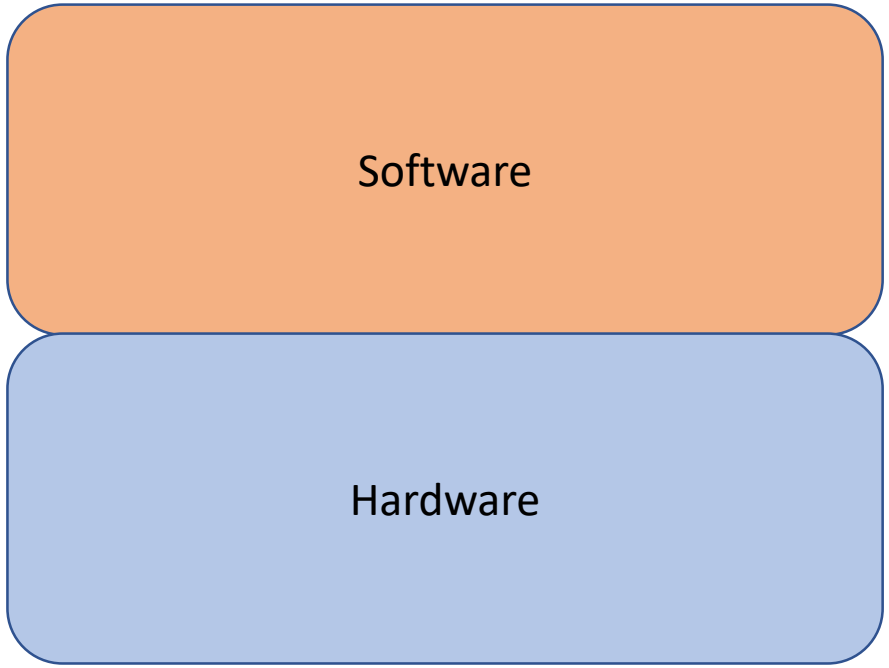
How does this happen?



From a high level, we will divide a computer system into two parts

I. Hardware

How does this happen?

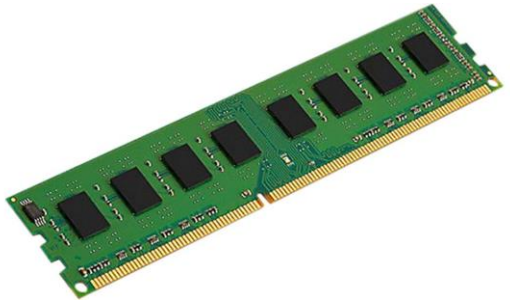


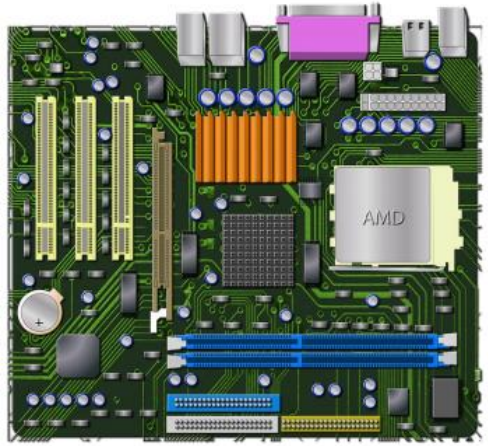
From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

I. Hardware

The **physical** parts of a computer

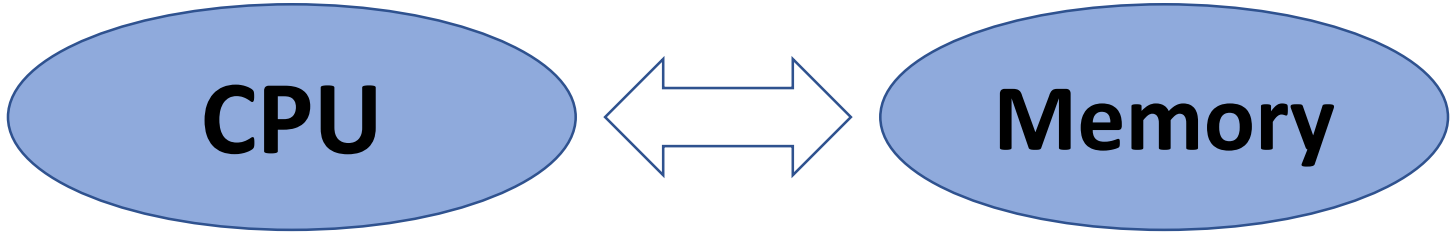


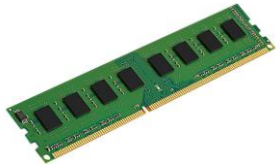


I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



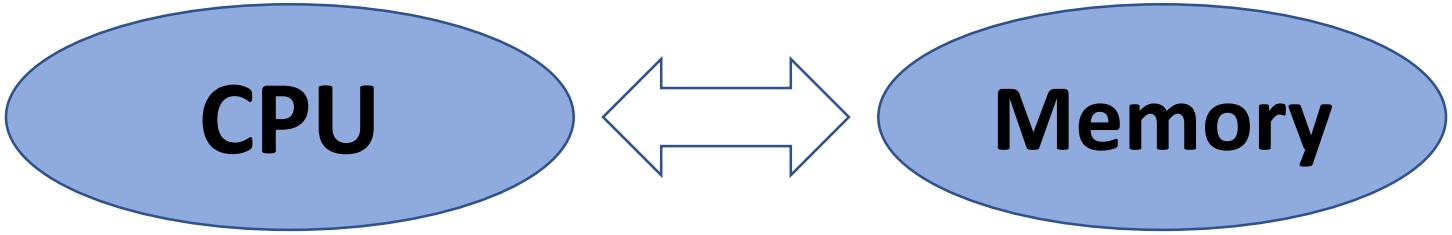


I. Hardware

(The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



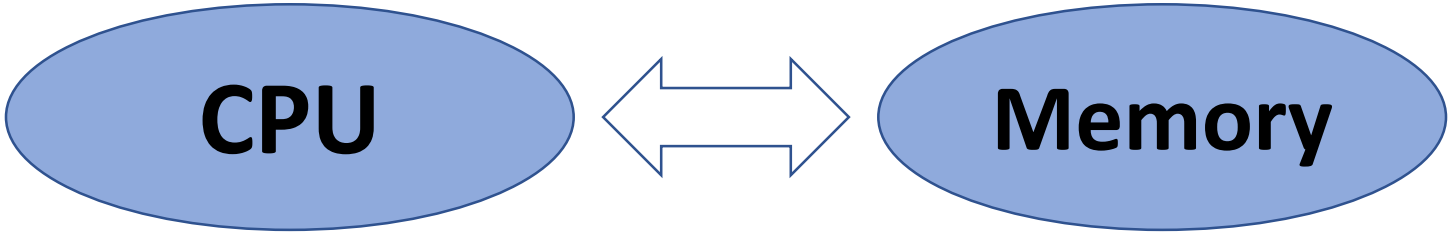
Brain with no short-term memory

Scratch Pad

I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



Brain with no short-term memory

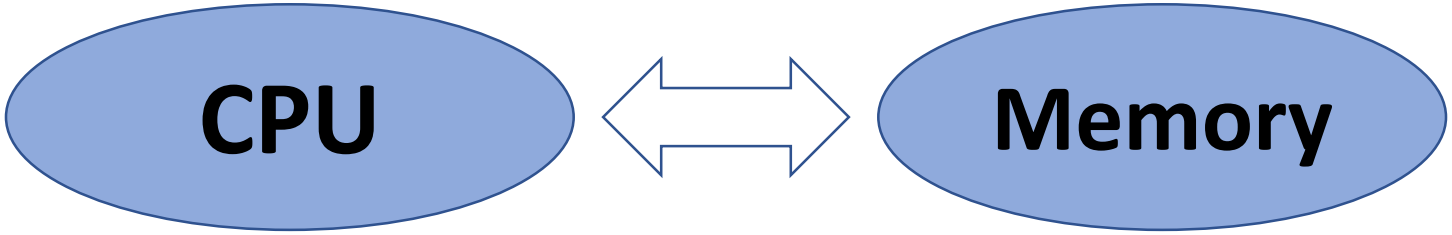
Scratch Pad

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

Scratch Pad

- List of instructions
- Data

I. Hardware

(The **physical** parts of a compute

that

How does it "execute" instructions?

Simplistic View:

CPU

Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

List

- Data

I. Hardware

(The **physical** parts of a compute

that

How does it "execute" instructions?

Remember that instructions are strands of zeros and ones (0011011010110101001)

Simplistic View:

CPU

Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

List

- Data

I. Hardware

(The **physical** parts of a compute

that

Simplistic View:

CPU

Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

List

- Data

How does it "execute" instructions?

Remember that instructions are strands of zeros and ones (0011011010110101001)

CPU deciphers these instructions by using

Electricity™

I. Hardware

(The **physical** parts of a compute

that

Simplistic View:

CPU

How does it "execute" instructions?

Remember that instructions are strands of zeros and ones (0011011010110101001)

CPU deciphers these instructions by using

Electricity™

The CPU then executes the appropriate operation based on the instruction

Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

List

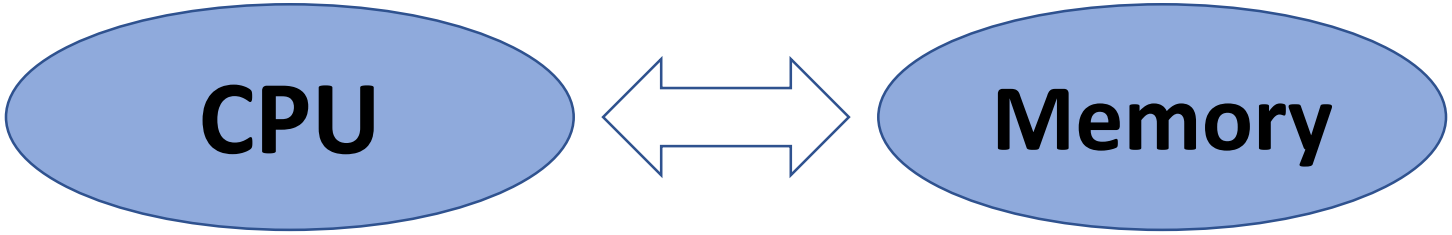
- Data

Arithmetic, Move Stuff Around in Memory, …

I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

Scratch Pad

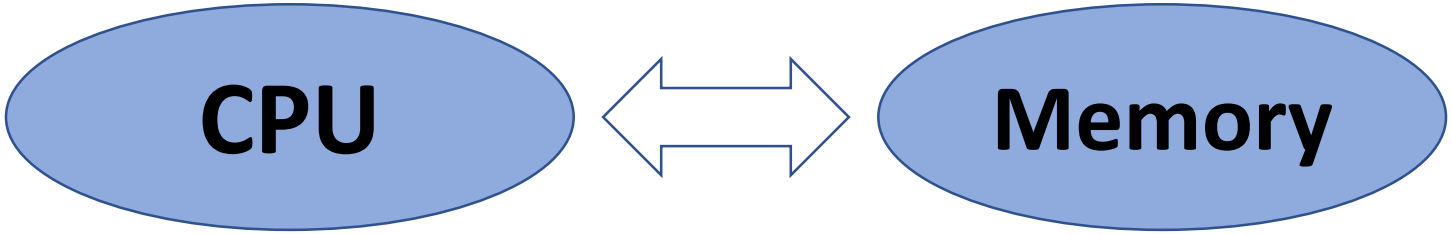
- List of instructions
- Data

Instructions will always be in the form of 0s and 1s, but will vary by hardware

I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



This happens very fast

… like REALLY fast

Brain with no short-term memory

- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

Scratch Pad

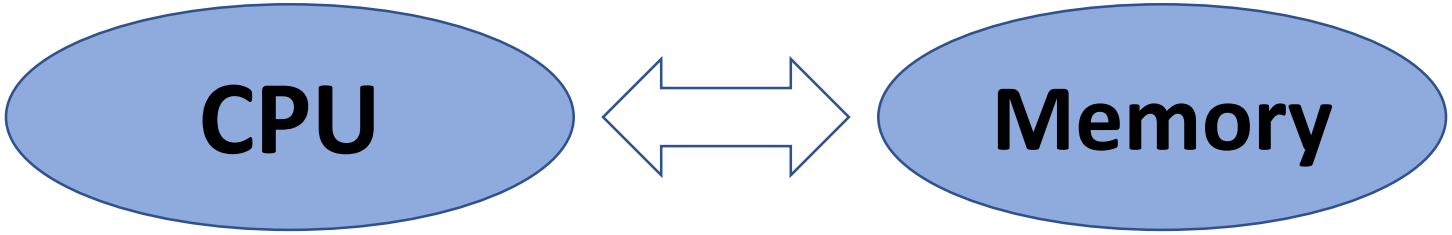
- List of instructions
- Data

Instructions will always be in the form of 0s and 1s, but will vary by hardware

I. Hardware (The **physical** parts of a computer)

Computer: semi-magical box that executes instructions

Simplistic View:



This happens very fast

… like REALLY fast

Intel i7 = **3 BILLION** instructions per second

Brain with no short-term memory

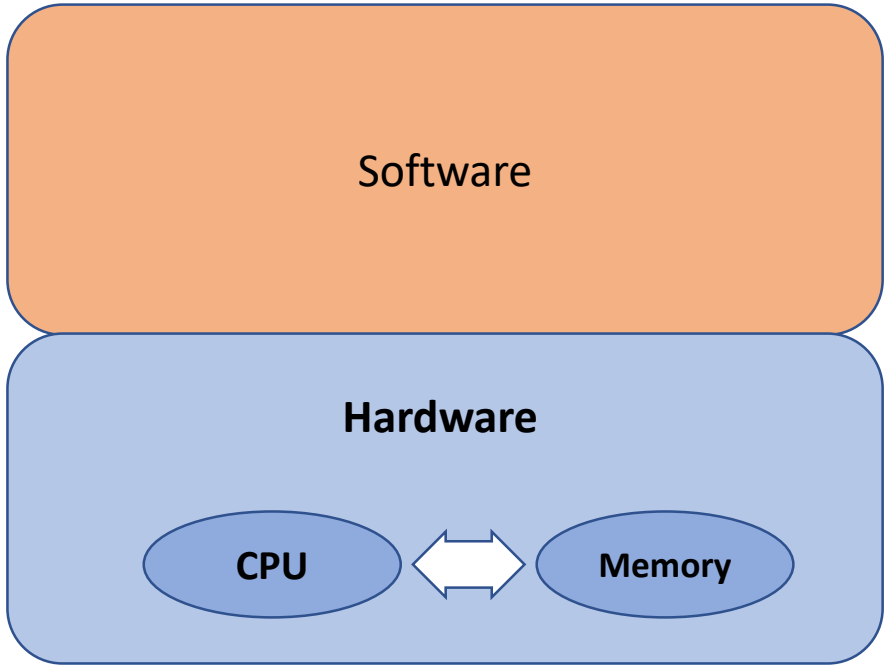
- Executes instructions
- Manipulates memory (changes it, moves things around, ..)

Scratch Pad

- List of instructions
- Data

Instructions will always be in the form of 0s and 1s, but will vary by hardware

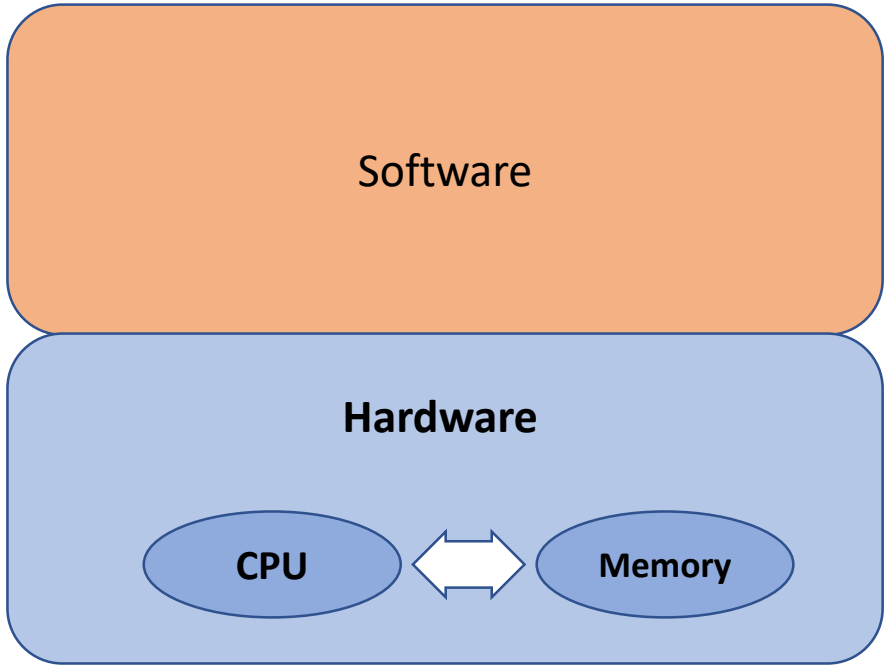
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

II. Software



Where do these instructions come from??

II. Software

A program (a sequence of computer instructions) that tells the computer how to work

II. Software

A program (a sequence of computer instructions) that tells the computer how to work

II. Software

Computer: semi-magical box that executes instructions

A program (a sequence of computer instructions) that tells the computer how to work

Humans (computer programmers) writes software

II. Software

Computer: semi-magical box that executes instructions

A program (a sequence of computer instructions) that tells the computer how to work

Humans (computer programmers) writes software

Remember that computers only understand 0s and 1s

… So, do we have to write programs in 0s and 1s?????

II. Software

Computer: semi-magical box that executes instructions

A program (a sequence of computer instructions) that tells the computer how to work

Humans (computer programmers) writes software

Remember that computers only understand 0s and 1s

… So, do we have to write programs in 0s and 1s?????

NO!!!

(thank goodness!!)

II. Software

Computer: semi-magical box that executes instructions

We write programs in a **high-level** programming language

These are languages that are very easy for humans to read

II. Software

Computer: semi-magical box that executes instructions

We write programs in a **high-level** programming language

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

II. Software

Computer: semi-magical box that executes instructions

We write programs in a **high-level** programming language

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

A computer doesn’t understand what this means…

II. Software

Computer: semi-magical box that executes instructions

We write programs in a **high-level** programming language

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

A computer doesn't understand what this means…

We need to translate it to 0s and 1s

II. Software

Computer: semi-magical box that executes instructions

Translating out code into binary***

Source Code

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

```
section     .text
global      _start

_start:

    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80

    mov     eax,1
    int     0x80

section     .data

msg     db  'Hello, world!',0xa
len     equ $ - msg
```

Source code gets translated into **assembly language**

*** This process varies by language (this process is not entirely true for Python)

II. Software

Computer: semi-magical box that executes instructions

Translating out code into binary***

Source Code

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

Source code gets translated into **assembly language**

10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000

Then translated into **machine code** (0s and 1s)

*** This process varies by language (this process is not entirely true for Python)

II. Software

Computer: semi-magical box that executes instructions

Translating out code into binary***

Source Code

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

```
section     .text
global      _start

_start:

    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80

    mov     eax,1
    int     0x80

section     .data

msg     db  'Hello, world!',0xa
len     equ $ - msg
```

Source code gets translated into **assembly language**

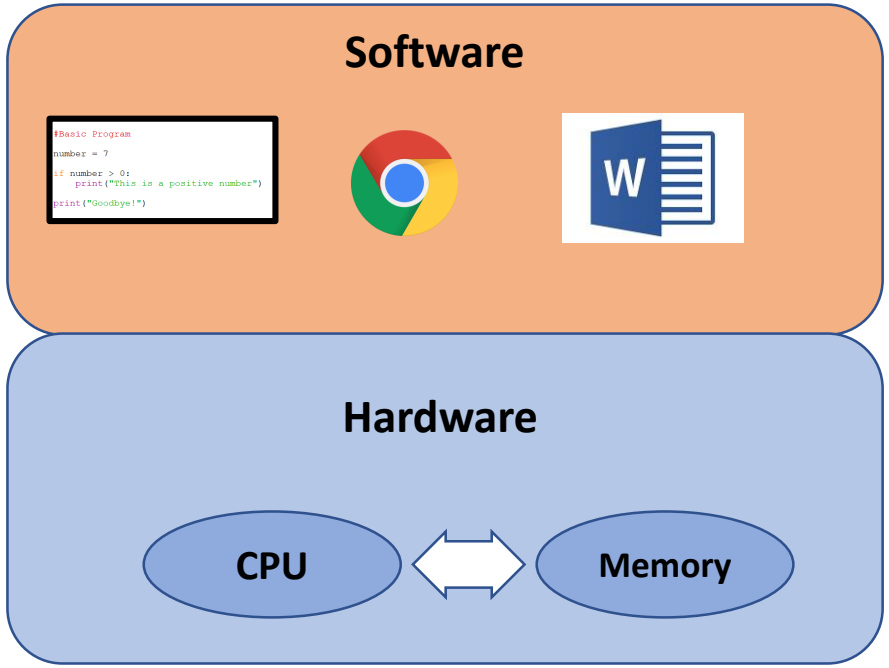
10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000

Then translated into **machine code** (0s and 1s)

The program that does this translation from source code to machine code is known as the **compiler**

*** This process varies by language (this process is not entirely true for Python)

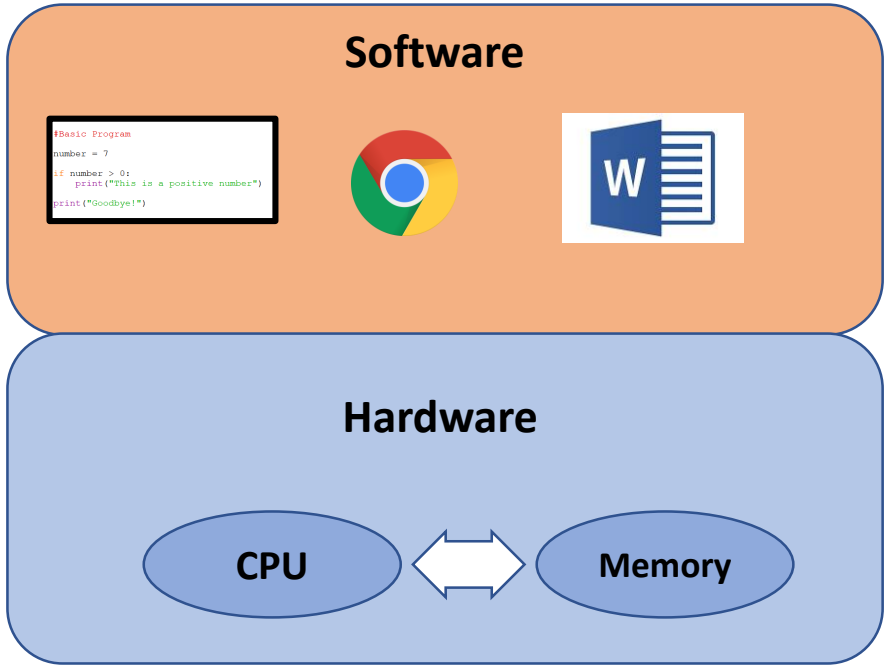
How does this happen?



From a high level, we will divide a computer system into two parts

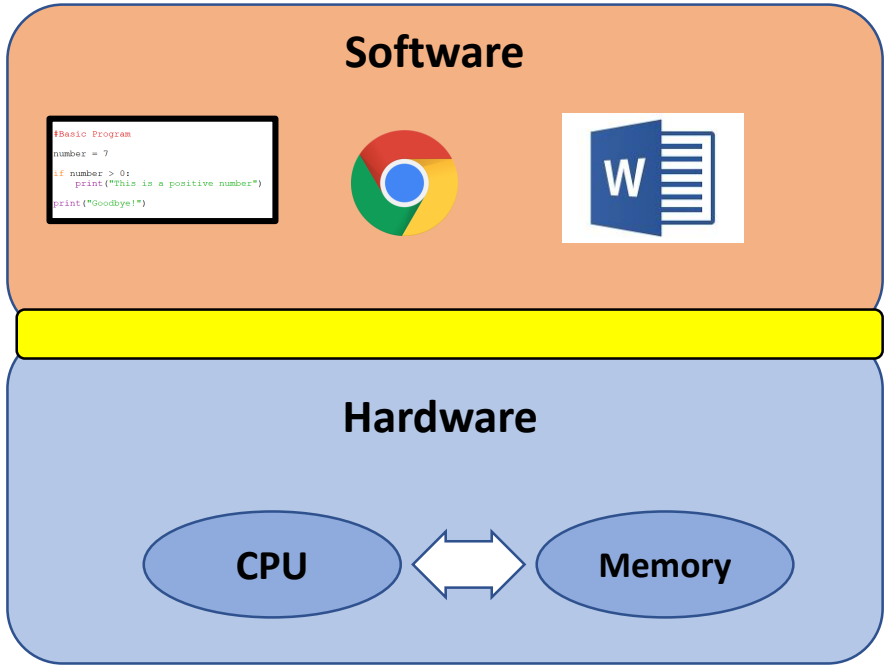
I. **Hardware**
II. **Software**

How does this happen?



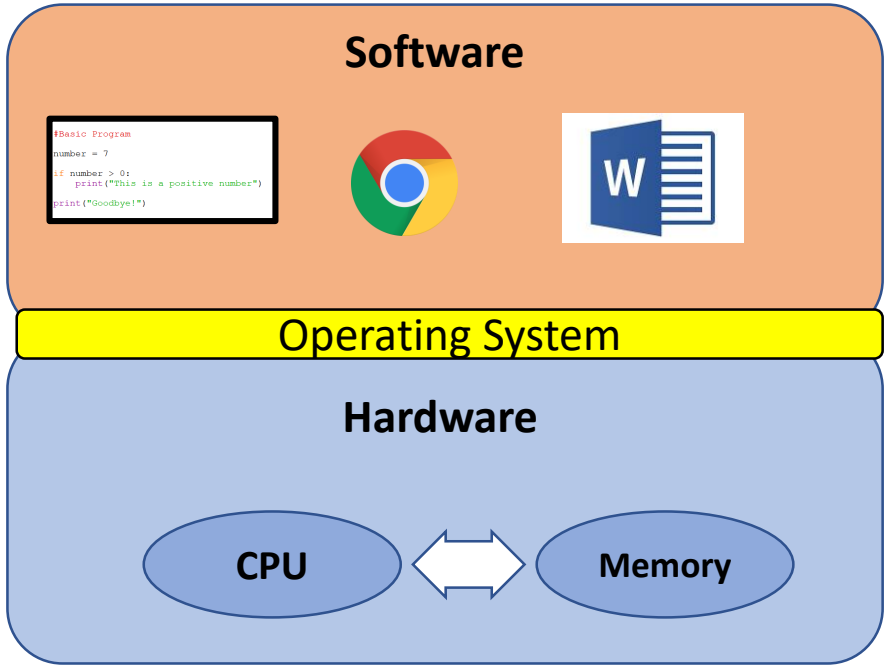
We are missing an important piece here….

How does this happen?



We are missing an important piece here….

How does this happen?

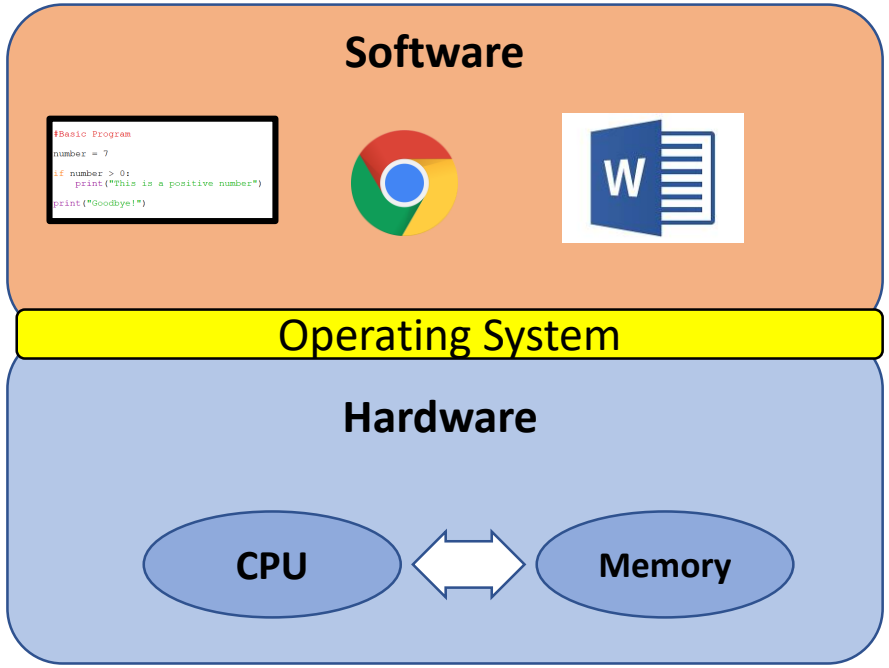


We are missing an important piece here….

The **operating system** !!

Operating System is a piece of software that acts as a middleman between hardware and software

How does this happen?



We are missing an important piece here….

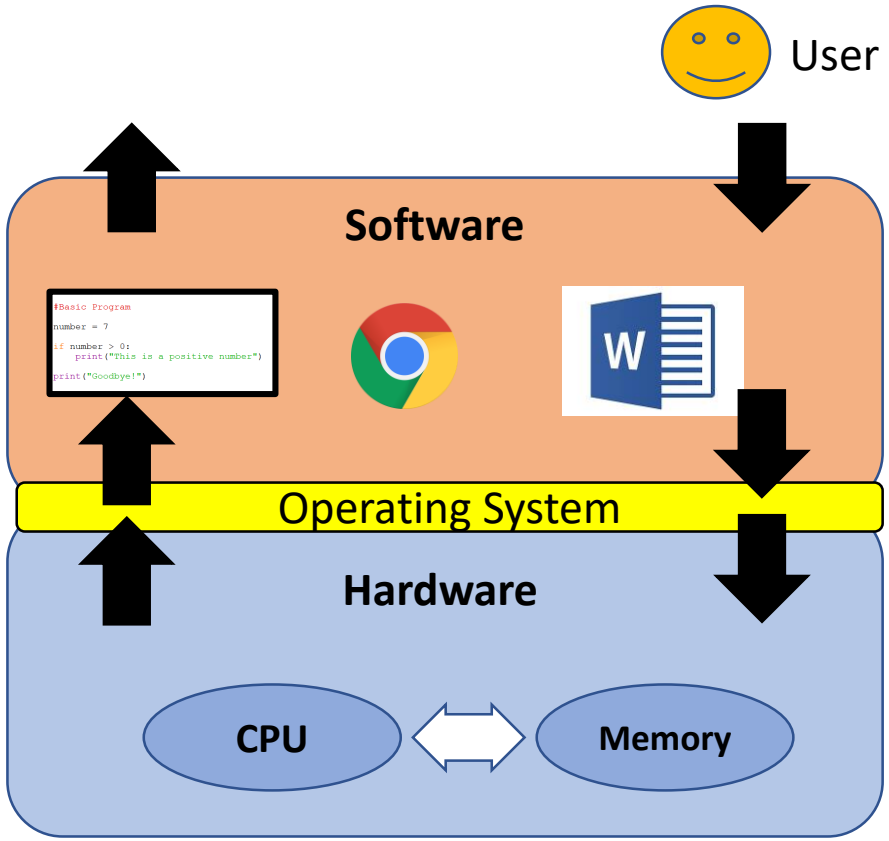
The **operating system** !!

Operating System is a piece of software that acts as a middleman between hardware and software

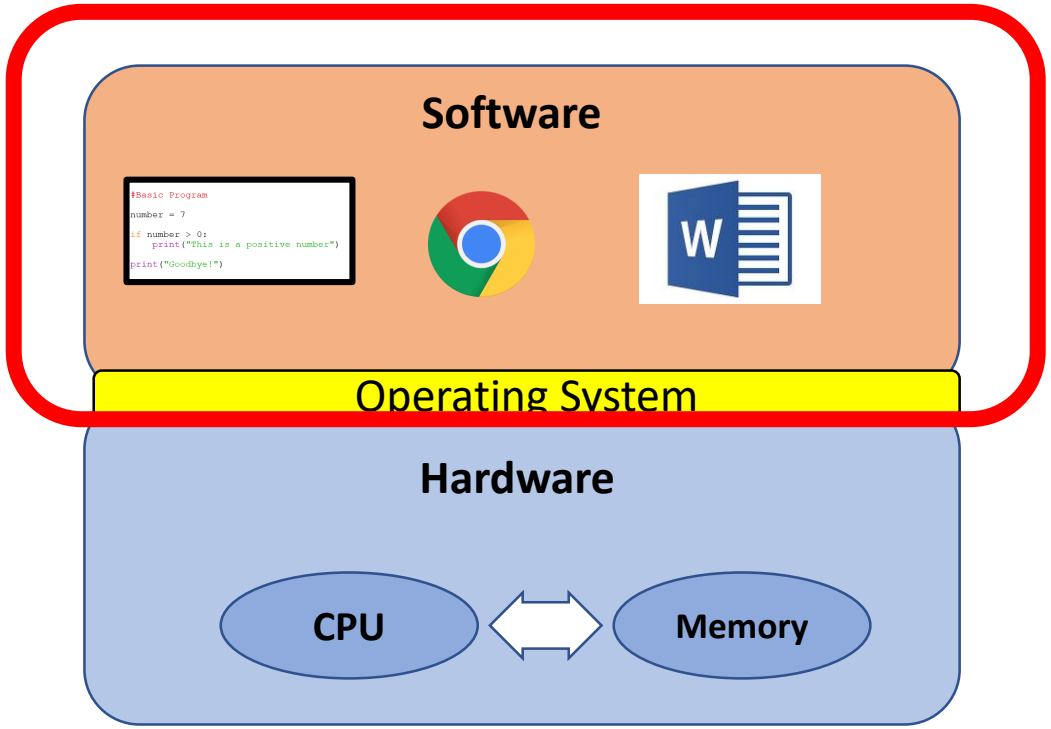
It is also serves as the computer's fundamental user interface

Software writers must write programs that are compatible with the OS

How does this happen?

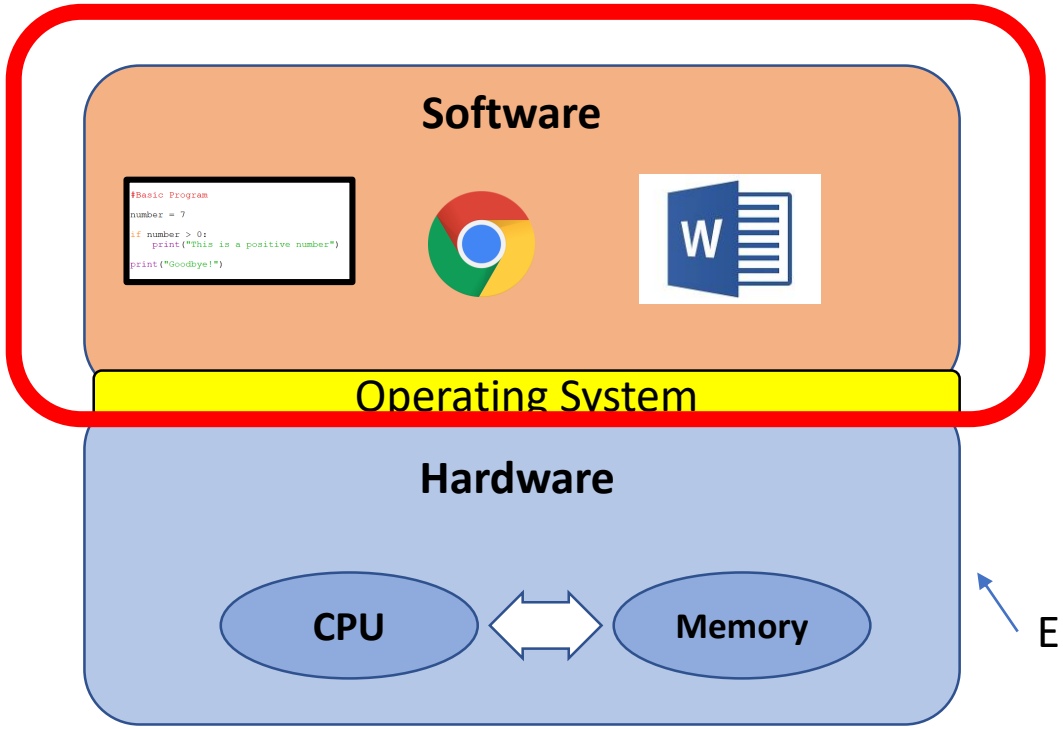


How does this happen?



Computer scientist focus more specifically on software

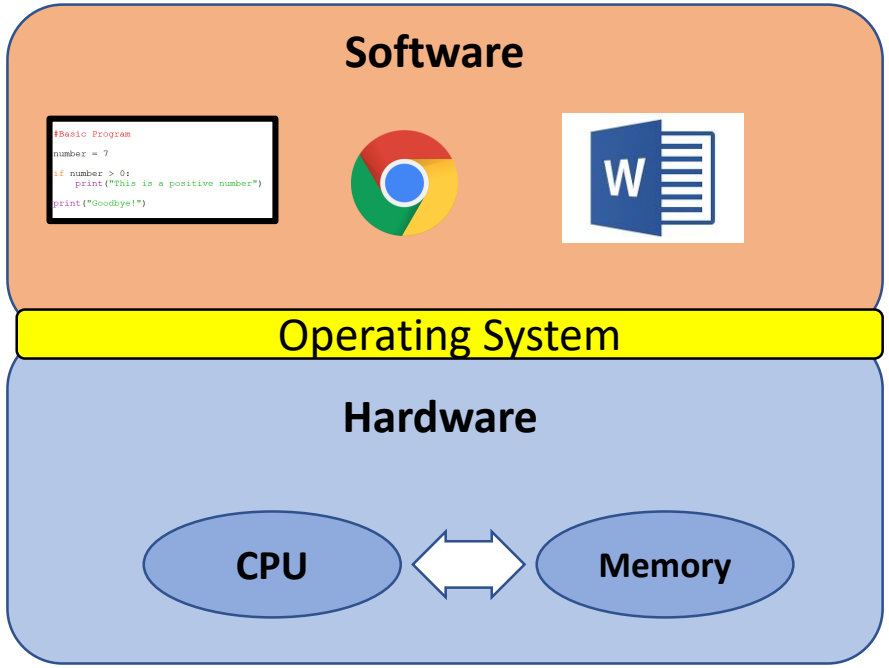
How does this happen?



Computer scientist focus more specifically on software

Electrical and Computer Engineers handle all the magic

How does this happen?



All done!

I hope you have a greater appreciate for computers ☺

CSCI 127: What do we do?

We will be writing computer programs in Python!

CSCI 127: What do we do?

We will be writing computer programs in Python!

Worldwide, Nov 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	30.8 %	+1.8 %
2		Java	16.79 %	-2.3 %
3		JavaScript	8.37 %	+0.3 %
4		C#	6.42 %	-0.9 %
5		PHP	5.92 %	-0.2 %
6		C/C++	5.78 %	-0.2 %
7		R	4.16 %	+0.4 %
8		Objective-C	3.57 %	+1.0 %
9		Swift	2.29 %	-0.2 %
10		TypeScript	1.84 %	-0.0 %
11		Matlab	1.65 %	-0.1 %

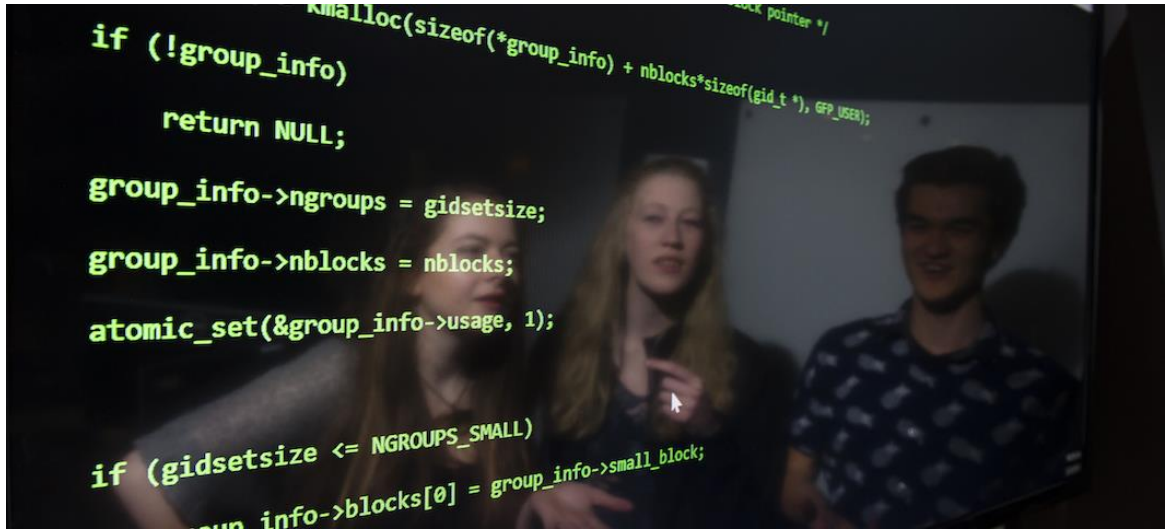
Areas of Computer Science

- Software Engineering
- Data Science
- Web Development
- Operating Systems
- Cybersecurity
- Machine Learning
- System Administration
- Programming Language Design
- Mobile Computing
- Artificial Intelligence
- Human Computer Interaction
- Data Mining
- Theoretical CS
- Cloud Computing
- Algorithms
- Computer Graphics
- Computer Vision / Robotics
- Computer Networks
- Computational Biology

Why computer science?

- Job market is very desirable
- Very rewarding
- Job flexibility
- Creating computer programs can improve efficiency + help solve problems in ANY field

This world needs more problem solvers



Some closing thoughts….

Programming is science…. but also art

It requires skill, time, and sometimes creativity

Some closing thoughts….

You might struggle at first and things might not make sense, and that is **ok**

That does not mean that you can't become a great computer programmer

Some closing thoughts….

Some closing thoughts….

“Not everyone can become a great artist, but a great artist can come from anywhere”

-Anton Ego, *Ratatouille*

Some closing thoughts….

“Not everyone can become a great programmer, but a great programmer can come from anywhere”

~~-Anton Ego, *Ratatouille*~~

-Reese Pearsall

The End

Next time: Python Installation, Python Introduction