

# Why Program?

## Chapter 1

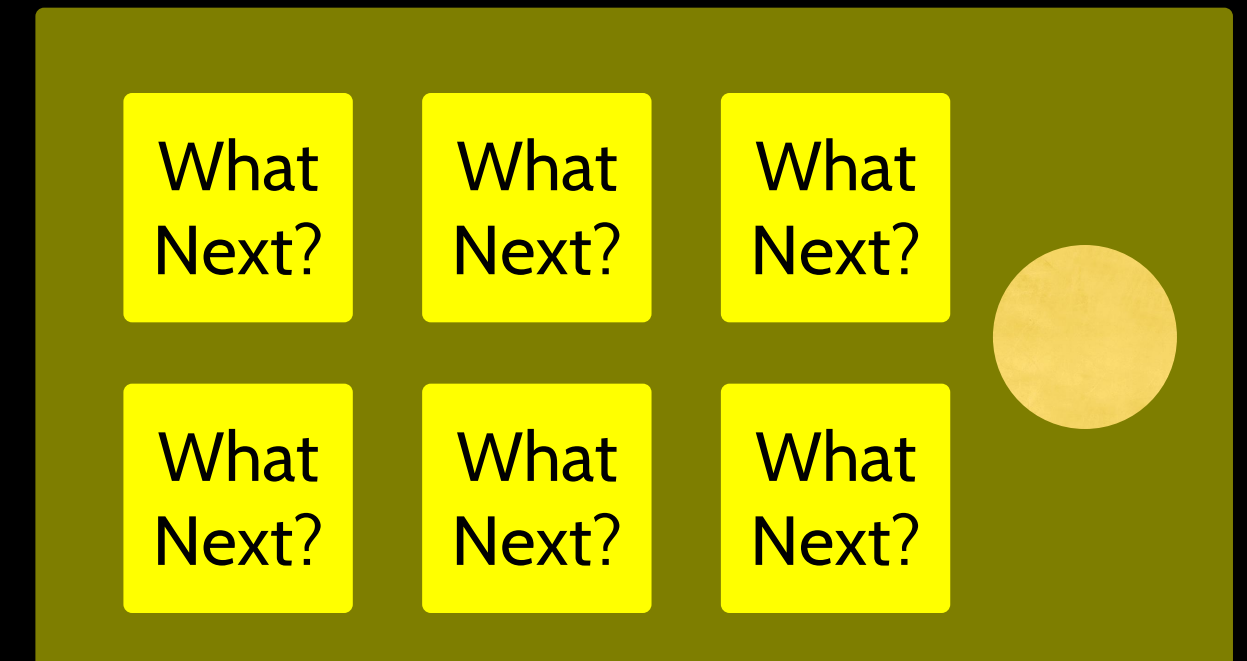
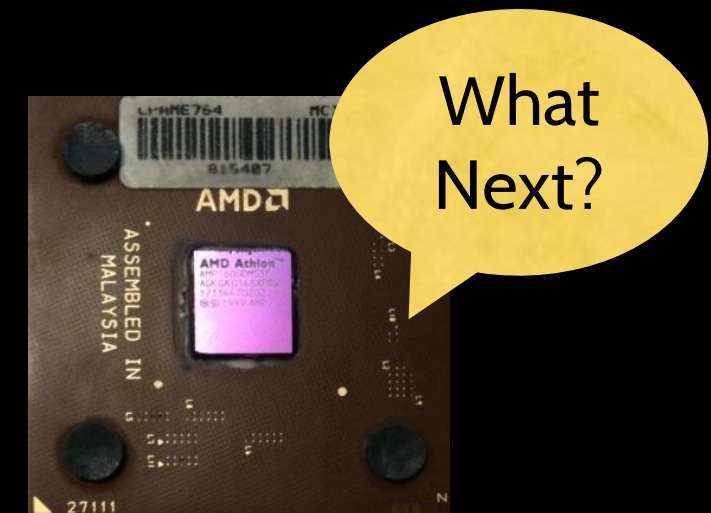


Python for Informatics: Exploring Information  
[www.pythonlearn.com](http://www.pythonlearn.com)



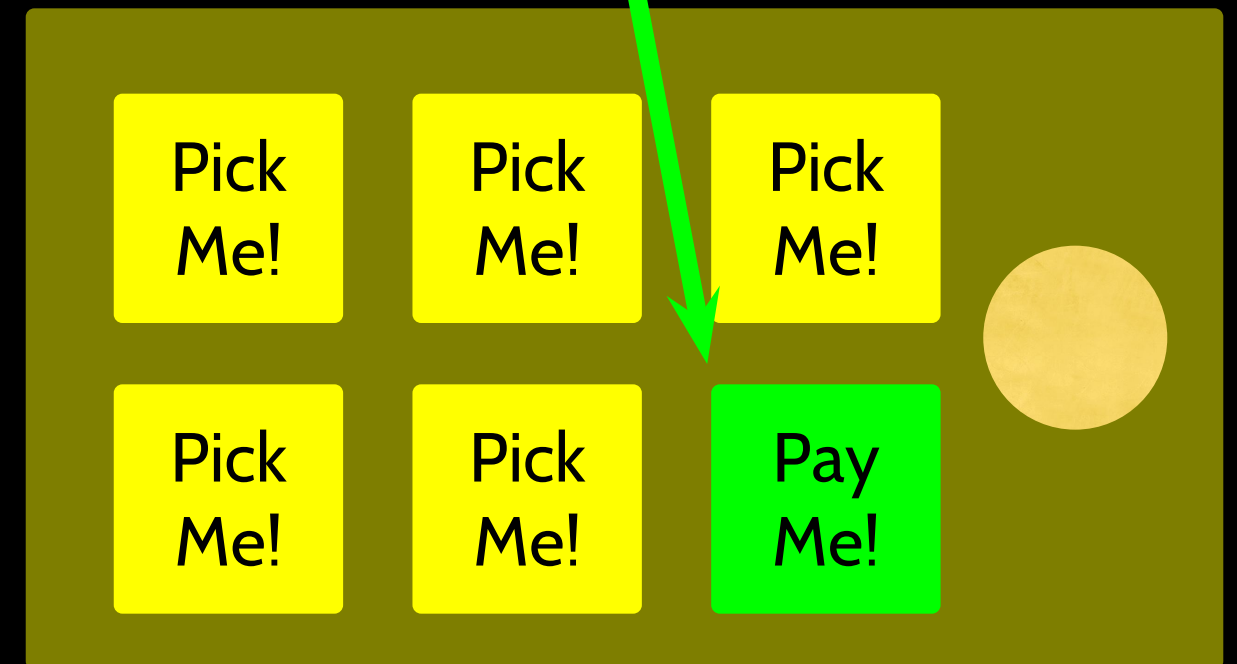
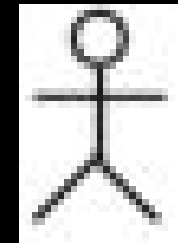
# Computers want to be helpful...

- Computers are built for one purpose - to do things for us
- But we need to speak their language to describe what we want done
- Users have it easy - someone already put many different programs (instructions) into the computer and users just pick the ones we want to use



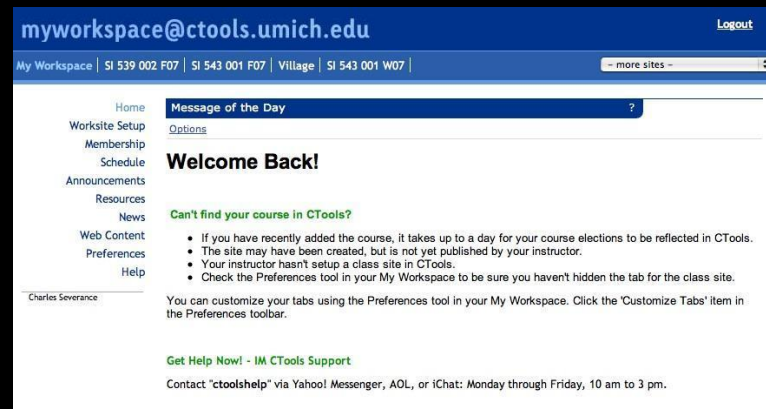
# Programmers Anticipate Needs

- iPhone Applications are a market
- iPhone Applications have over 3 Billion downloads
- Programmers have left their jobs to be full-time iPhone developers
- Programmers know the **ways of the program**

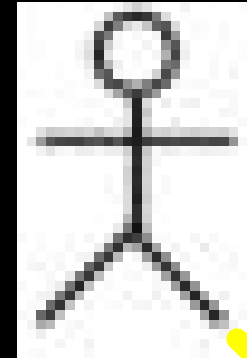


# Users vs. Programmers

- Users see computers as a set of tools – word processor, spreadsheet, map, todo list, etc.
- Programmers learn the computer “ways” and the computer language
- Programmers have some tools that allow them to build new tools
- Programmers sometimes write tools for lots of users and sometimes programmers write little “helpers” for themselves to automate a task



User



Programmer



....



From a software creator's point of view, we build the software. The end users (stakeholders/actors) are our masters – who we want to please – often they pay us money when they are pleased. But the data, information, and networks are our problem to solve on their behalf. The hardware and software are our friends and allies in this quest.

# Why be a programmer?

- To get some task done - we are the user and programmer
  - › Clean up survey data
- To produce something for others to use - a programming job
  - › Fix a performance problem in the Sakai software
  - › Add guestbook to a web site

# What is Code? Software? A Program?

- A sequence of stored instructions
  - › It is a little piece of our intelligence in the computer
  - › It is a little piece of our intelligence we can give to others - we figure something out and then we encode it and then give it to someone else to save them the time and energy of figuring it out
- A piece of creative art - particularly when we do a good job on user experience

# Programs for Humans...



<http://www.youtube.com/watch?v=vlzwuFkn88U>



# Programs for Humans...

while music is playing:

Left hand out and up

Right hand out and up

Flip Left hand

Flip Right hand

Left hand to right shoulder

Right hand to left shoulder

Left hand to back of head

Right hand to back of head

Left hand to right hip

Right hand to left hip

Left hand on left bottom

Right hand on right bottom

Wiggle

Wiggle

Jump



<http://www.youtube.com/watch?v=sN62PAKoBfE>

# Programs for Humans...

while music is playing:

Left hand out and up

Right hand out and up

Flip Left hand

Flip Right hand

Left hand to right shoulder

Right hand to left shoulder

Left hand to back of head

Right **ham** to back of head

Left hand to right **hit**

Right hand to left **hit**

Left hand on left bottom

Right hand on right bottom

Wiggle

Wiggle

Jump



<http://www.youtube.com/watch?v=vlzwuFkn88U>

# Programs for Humans...

while music is playing:

Left hand out and up

Right hand out and up

Flip Left hand

Flip Right hand

Left hand to right shoulder

Right hand to left shoulder

Left hand to back of head

Right **hand** to back of head

Left hand to right **hip**

Right hand to left **hip**

Left hand on left bottom

Right hand on right bottom

Wiggle

Wiggle

Jump



<http://www.youtube.com/watch?v=vlzwuFkn88U>





the clown ran after the car and the car ran into the tent and the  
tent fell down on the clown and the car

# Programs for Python...



# Programs for Python...

```
name = raw_input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1
bigcount = None
bigword = None

for word,count in counts.items():
    if bigcount is None or count >
bigcount:
        bigword = word
        bigcount = count
print bigword, bigcount
```

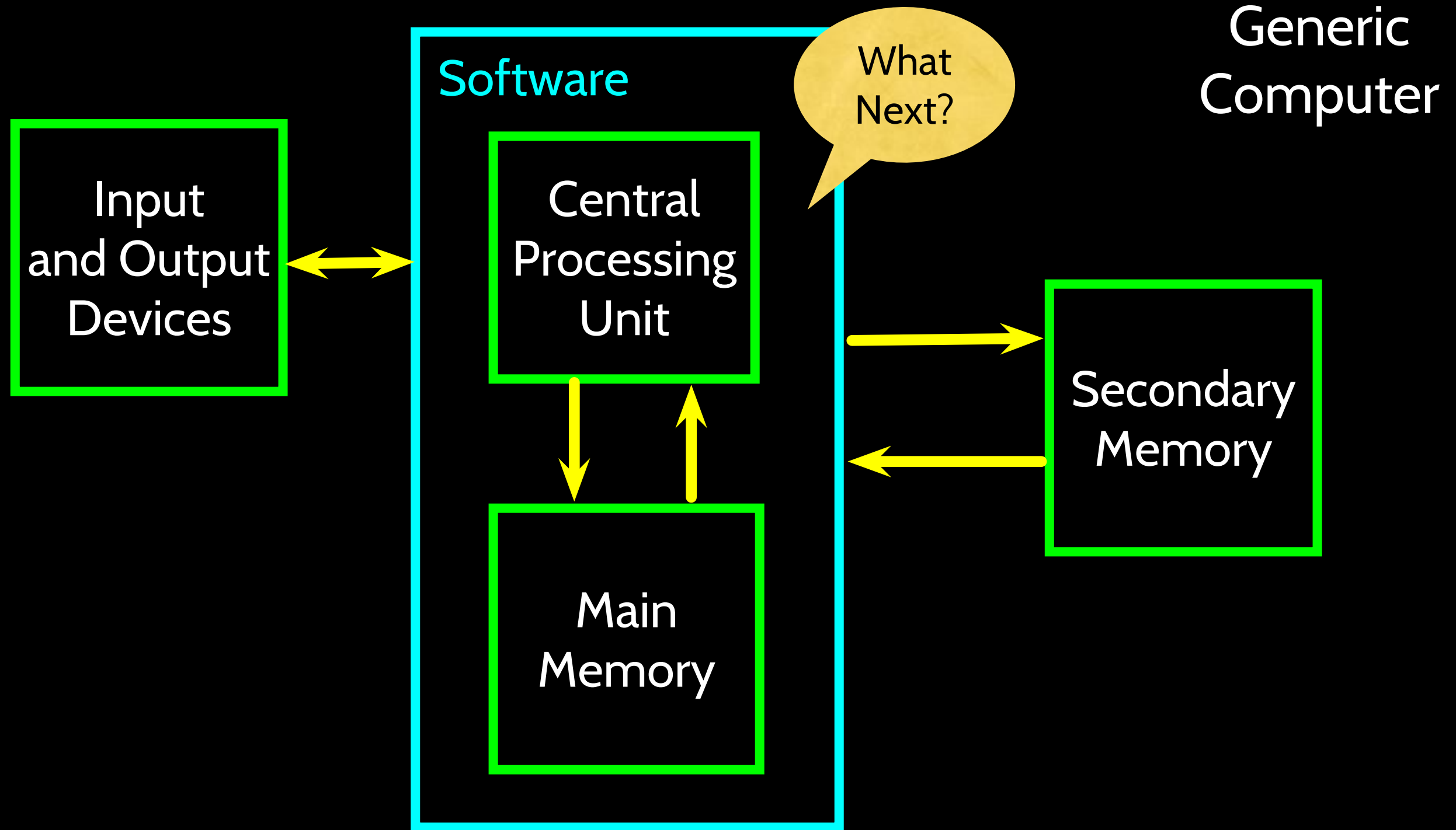
**python words.py**  
**Enter file:** words.txt  
**to 16**

**python words.py**  
**Enter file:** clown.txt  
**the 7**

# Hardware Architecture

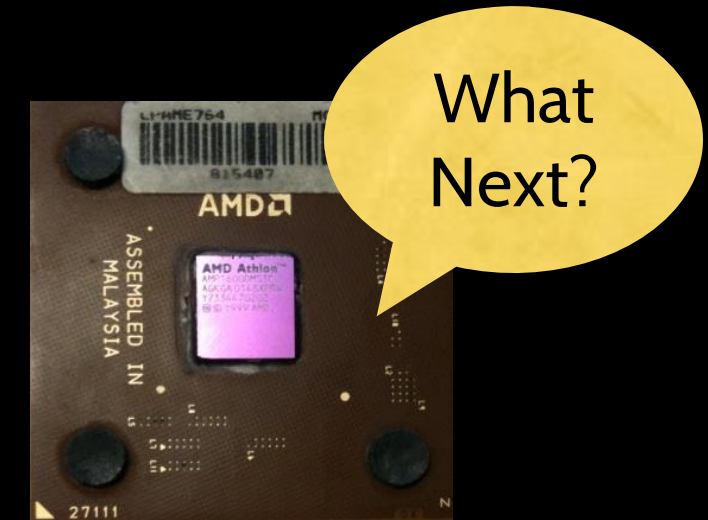


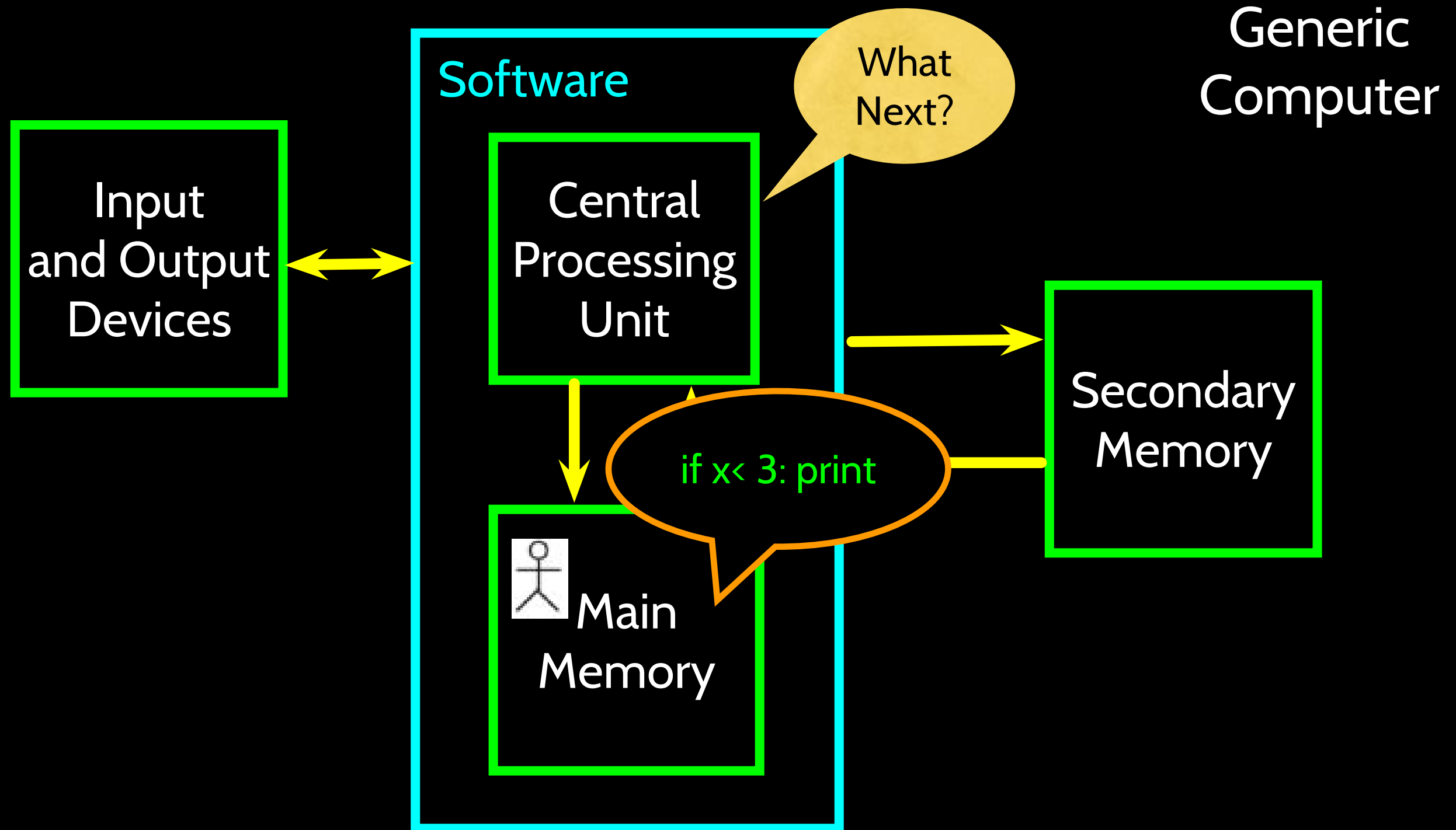


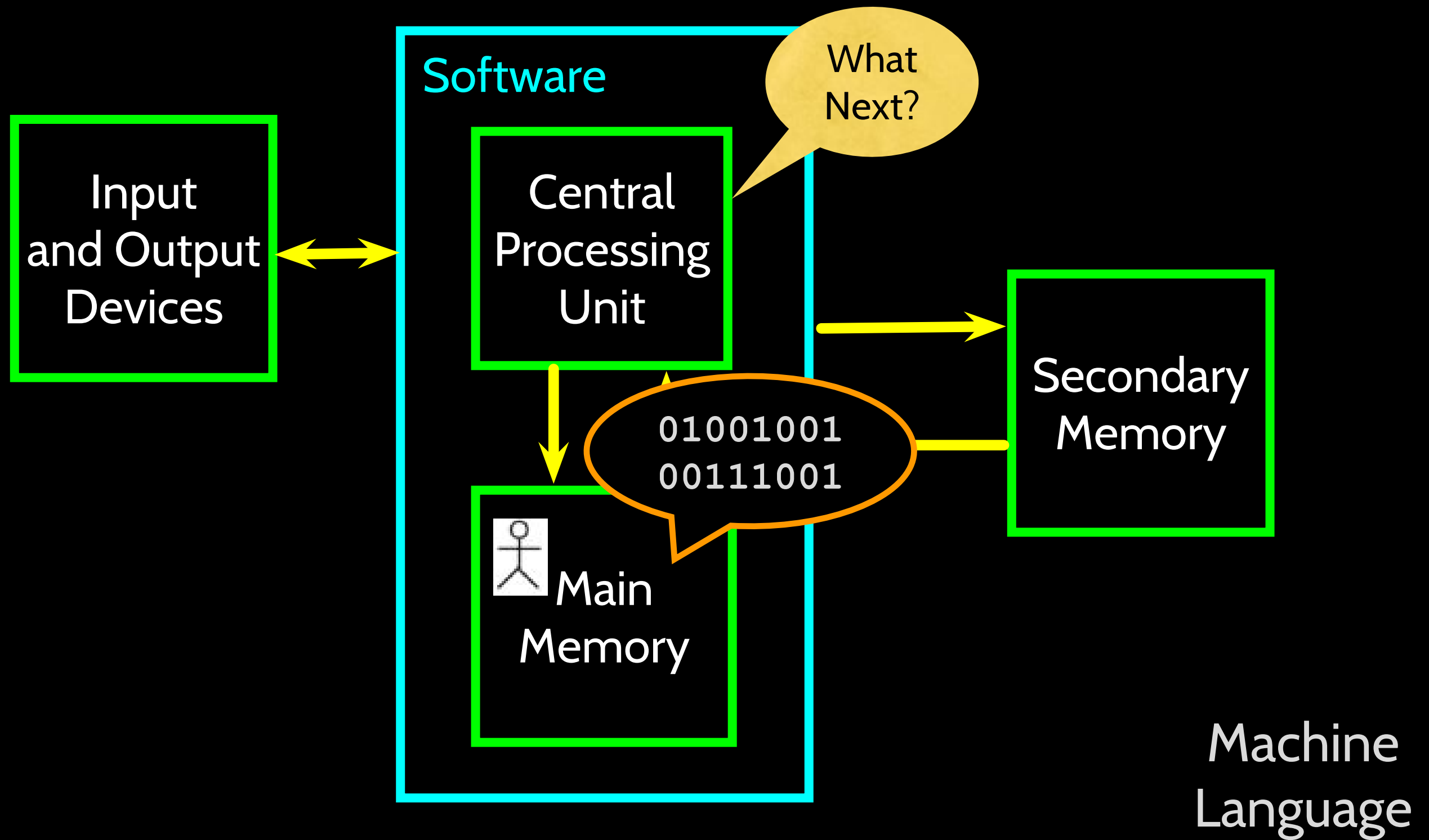


# Definitions

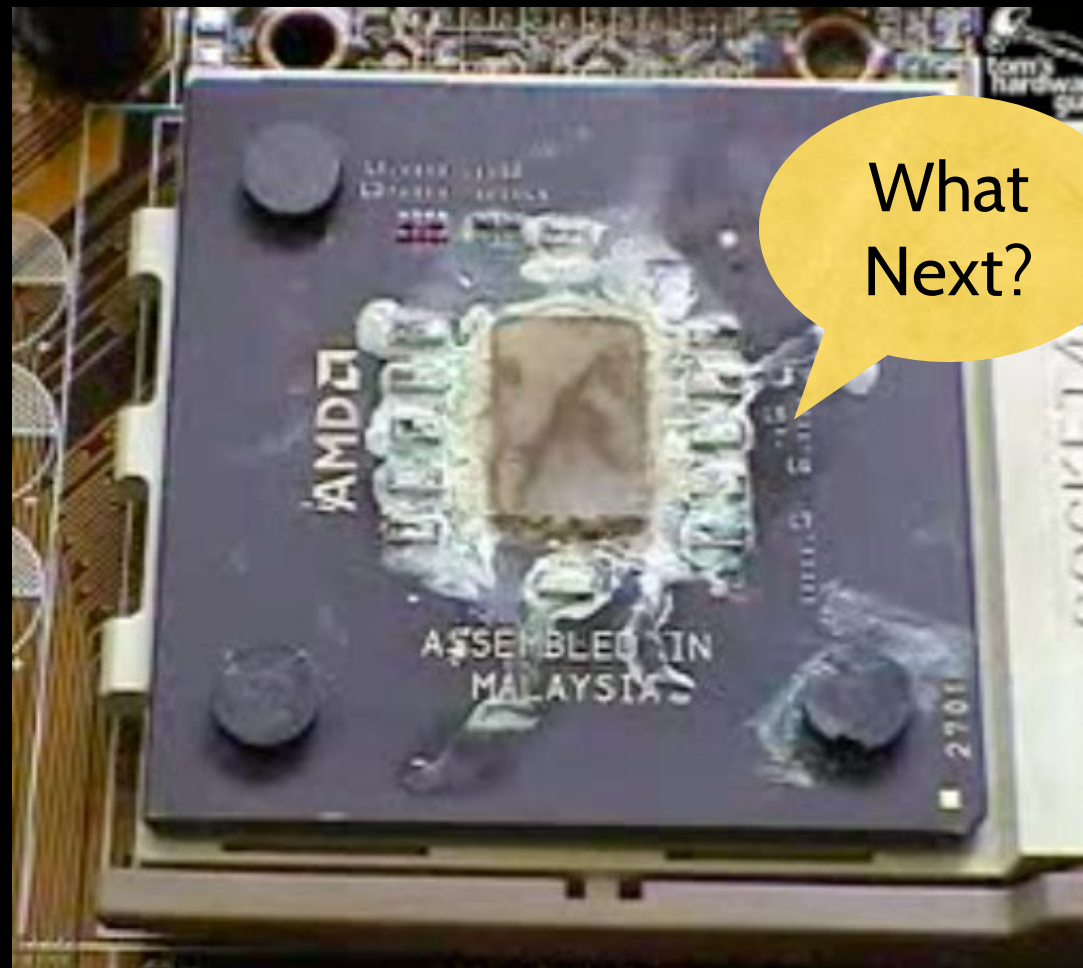
- **Central Processing Unit:** Runs the Program - The CPU is always wondering “what to do next”? Not the brains exactly - very dumb but very very fast
- **Input Devices:** Keyboard, Mouse, Touch Screen
- **Output Devices:** Screen, Speakers, Printer, DVD Burner
- **Main Memory:** Fast small temporary storage - lost on reboot - aka RAM
- **Secondary Memory:** Slower large permanent storage - lasts until deleted - disk drive / memory stick







# Totally Hot CPU



<http://www.youtube.com/watch?v=y39D4529FM4>

# Hard Disk in Action



<http://www.youtube.com/watch?v=9eMWG3fwiEU>

# Python as a Language

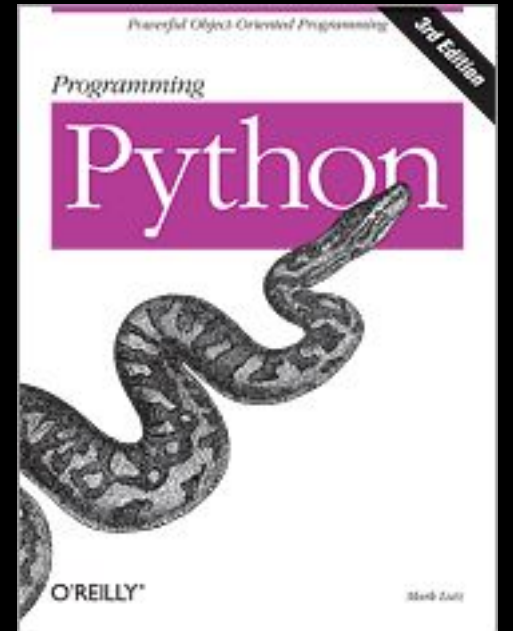
**Parseltongue** is the language of serpents and those who can converse with them. An individual who can speak **Parseltongue** is known as a **Parselmouth**. It is a very uncommon skill, and may be hereditary. Nearly all known **Parselmouths** are descended from Salazar Slytherin.



<http://harrypotter.wikia.com/wiki/Parseltongue>



**Python** is the language of the Python Interpreter and those who can converse with it. An individual who can speak **Python** is known as a **Pythonista**. It is a very uncommon skill, and may be hereditary. Nearly all known **Pythonistas** use software initially developed by **Guido van Rossum**.



# Early Learner: Syntax Errors

- We need to learn the **Python language** so we can communicate our instructions to Python. In the beginning we will make lots of mistakes and speak gibberish like small children.
- When you make a mistake, the computer does not think you are “cute”. It says “**syntax error**” – given that it *knows* the language and you are just learning it. It seems like Python is cruel and unfeeling.
- You must remember that *you* are intelligent and *can* learn. The computer is simple and very fast, but cannot learn. So **it is easier for you to learn Python than for the computer to learn English...**

# Talking to Python

```
csev$ python
```

```
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
```

```
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more  
information.
```

```
>>>
```



What next?

```
csev$ python
```

```
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
```

```
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more  
information.
```

```
>>> x = 1
```

```
>>> print x
```

```
1
```

```
>>> x = x + 1
```

```
>>> print x
```

```
2
```

```
>>> exit()
```

This is a good test to make sure that you have Python correctly installed. Note that `quit()` also works to end the interactive session.

# Let's Talk to Python...

```
dr-chuck2:~ csev$ python
Python 2.6.1 (r261:67515, Jun 24 2010, 21:47:49)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello world"
hello world
>>> 
```

```
Administrator: C:\Windows\system32\cmd.exe - C:\Python27\python.exe
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>C:\Python27\python.exe
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello world"
hello world
>>> _
```

What Do We Say?

# Elements of Python

- **Vocabulary / Words** - Variables and Reserved words (Chapter 2)
- **Sentence structure** - valid syntax patterns (Chapters 3-5)
- **Story structure** - constructing a program for a purpose



```
name = raw_input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1
bigcount = None
bigword = None

for word,count in counts.items():
    if bigcount is None or count >
bigcount:
        bigword = word
        bigcount = count
print bigword, bigcount
```

A short “story” about  
how to count words  
in a file in Python

python words.py  
Enter file: words.txt  
to 16

# Reserved Words

- You cannot use **reserved words** as variable names / identifiers

and del for is raise assert elif from  
lambda return break else global  
not try class except if or while  
continue exec import pass yield  
def finally in print as with

# Sentences or Lines

x	=	2	←	Assignment statement		
x	=	x	+	2	←	Assignment with expression
print	x	←	Print statement			

Variable

Operator

Constant

Reserved  
Word

# Programming Paragraphs

# Python Scripts

- Interactive Python is good for experiments and programs of 3-4 lines long.
- Most programs are much longer, so we type them into a file and tell Python to run the commands in the file.
- In a sense, we are “giving Python a script”.
- As a convention, we add “.py” as the suffix on the end of these files to indicate they contain Python.

# Writing a Simple Program

# Interactive versus Script

- Interactive

- You type directly to Python one line at a time and it responds

- Script

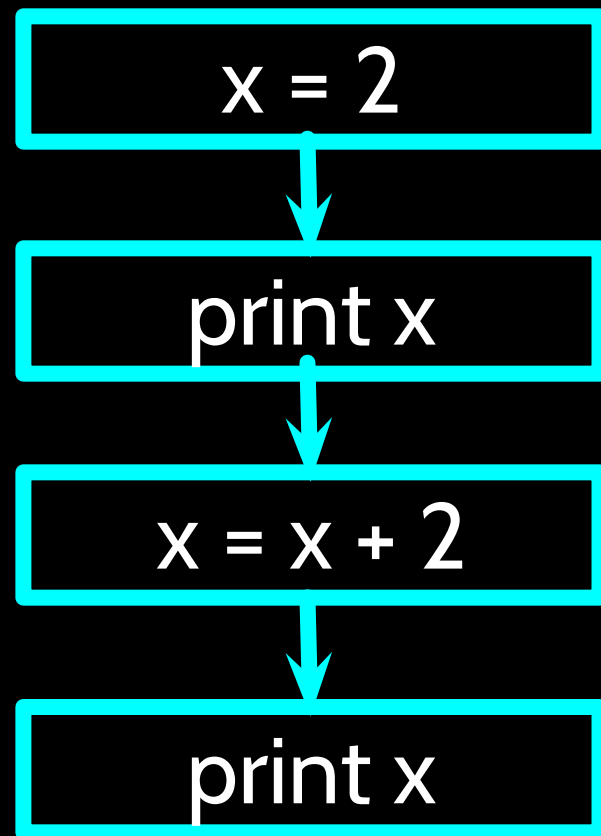
- You enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the statements in the file

# Program Steps or Program Flow

- Like a recipe or installation instructions, a program is a **sequence** of steps to be done in order.
- Some steps are **conditional** – they may be skipped.
- Sometimes a step or group of steps are to be **repeated**.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program (Chapter 4).



# Sequential Steps



Program:

```
x = 2  
print x  
x = x + 2  
print x
```

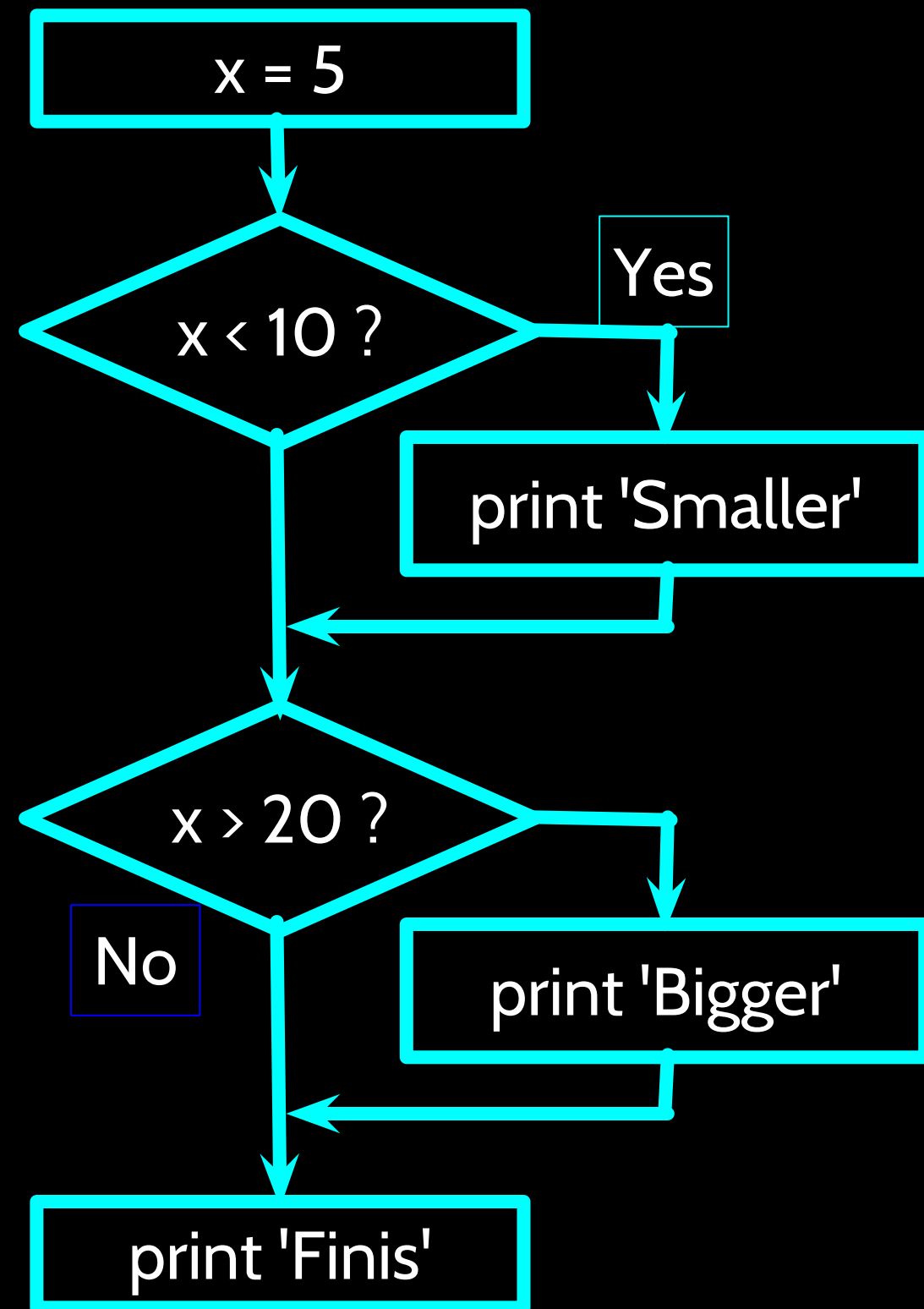
Output:

2

4

When a program is running, it flows from one step to the next.  
As programmers, we set up “paths” for the program to follow.

# Conditional Steps

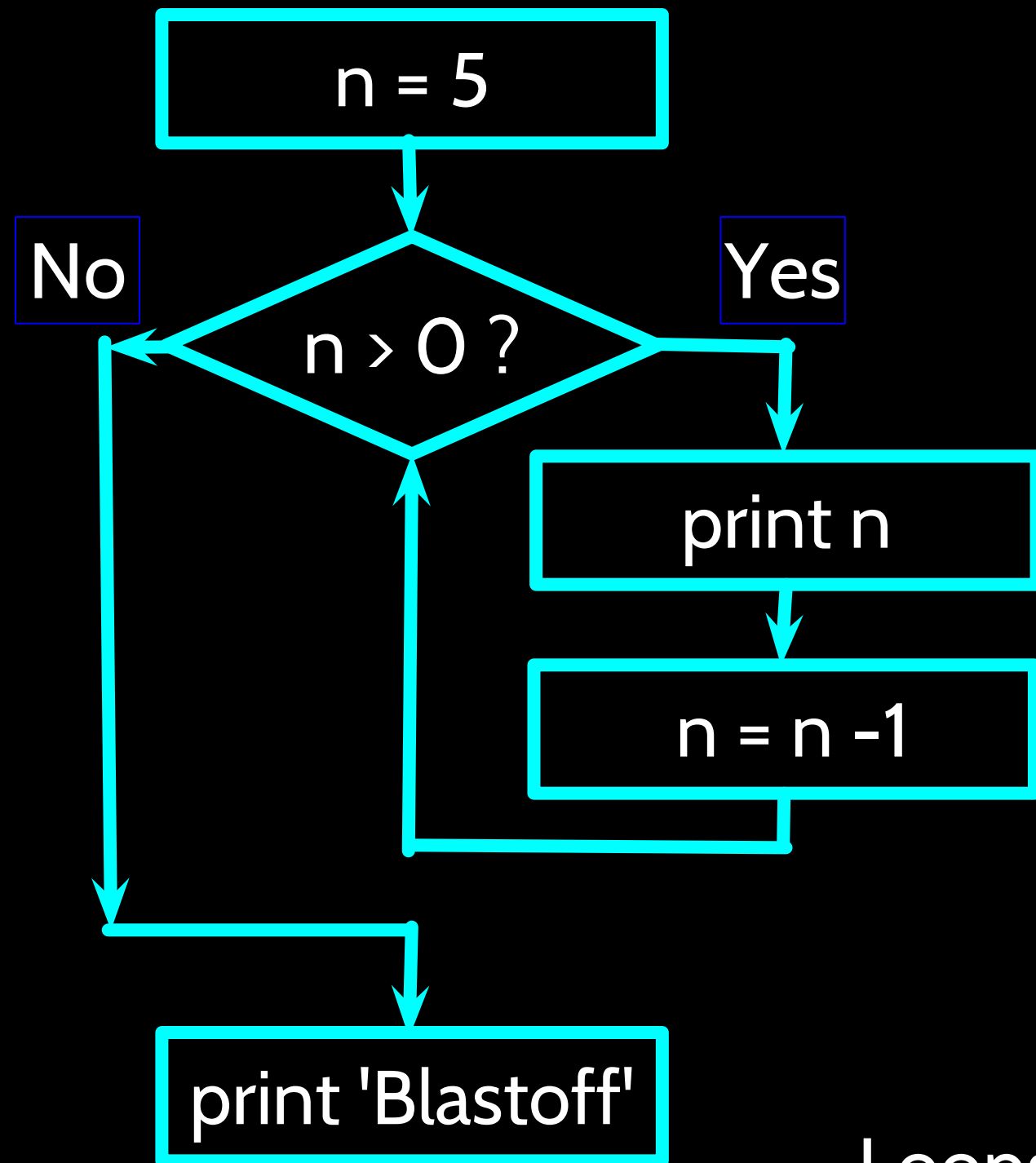


Program:

```
x = 5
if x < 10:
    print 'Smaller'
if x > 20:
    print 'Bigger'
print 'Finis'
```

Output:

Smaller  
Finis



# Repeated Steps

Program:

```
n = 5
while n > 0:
    print n
    n = n - 1
print 'Blastoff!'
```

Output:

5  
4  
3  
2  
1  
Blastoff!

Loops (repeated steps) have **iteration variables** that change each time through a loop. Often these **iteration variables** go through a sequence of numbers.

```
name = raw_input('Enter file:')  
handle = open(name, 'r')  
text = handle.read()  
words = text.split()
```

Sequential

Repeated

Conditional

```
counts = dict()  
for word in words:  
    counts[word] = counts.get(word,0) + 1  
bigcount = None  
bigword = None
```

```
for word,count in counts.items():  
    if bigcount is None or count >  
bigcount:  
        bigword = word  
        bigcount = count
```

```
print bigword, bigcount
```

```
name = raw_input('Enter file:')  
handle = open(name, 'r')  
text = handle.read()  
words = text.split()  
counts = dict()  
for word in words:  
    counts[word] = counts.get(word, 0) + 1
```

```
bigcount = None  
bigword = None  
for word, count in counts.items():  
    if bigcount is None or count >  
bigcount:  
        bigword = word  
        bigcount = count
```

```
print bigword, bigcount
```

A short Python “Story”  
about how to count  
words in a file

A word used to read  
data from a user

A sentence about  
updating one of the  
many counts

A paragraph about how  
to find the largest item  
in a list

# Summary

- This is a quick overview of Chapter 1
- We will revisit these concepts throughout the course
- Focus on the big picture



# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and [open.umich.edu](http://open.umich.edu) and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

... Insert new Contributors and Translators here

...