

Why Program?

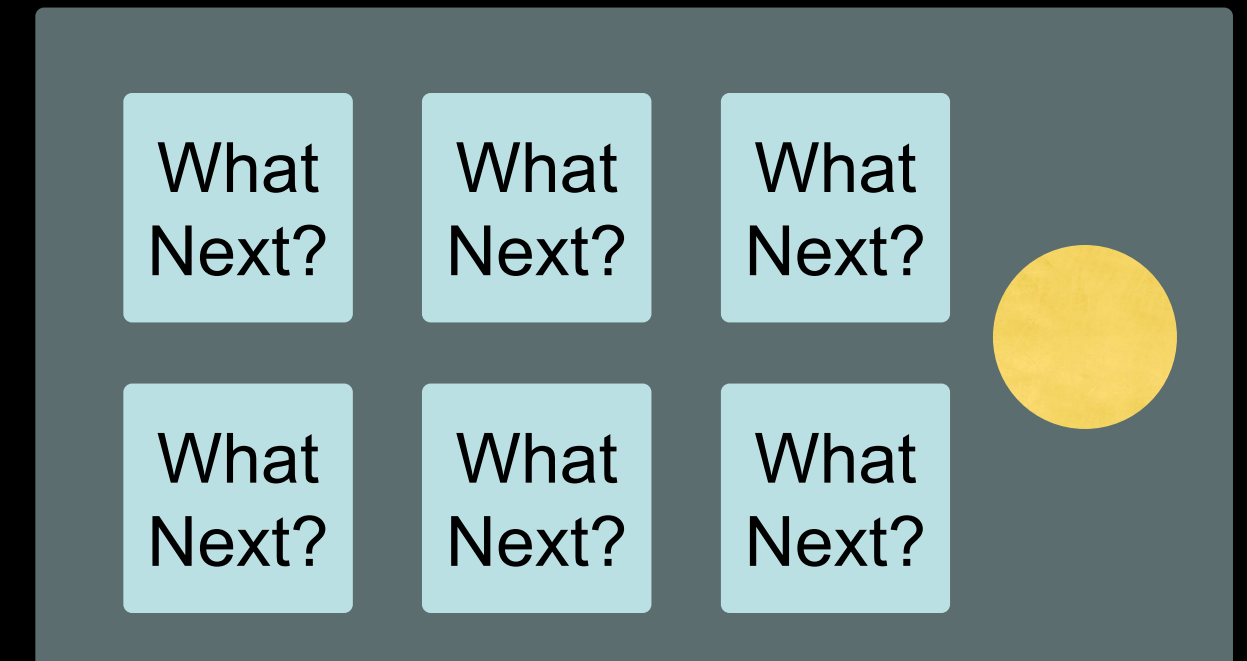
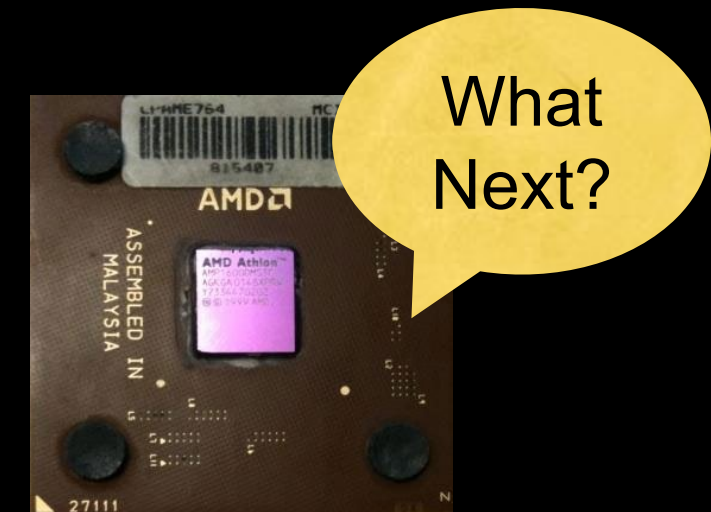
Chapter 1

Python for Everybody

www.py4e.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 they want to use



Users vs. Programmers

- Users see computers as a set of tools - word processor, spreadsheet, map, to-do 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

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 company software
 - Add a 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
 - 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

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

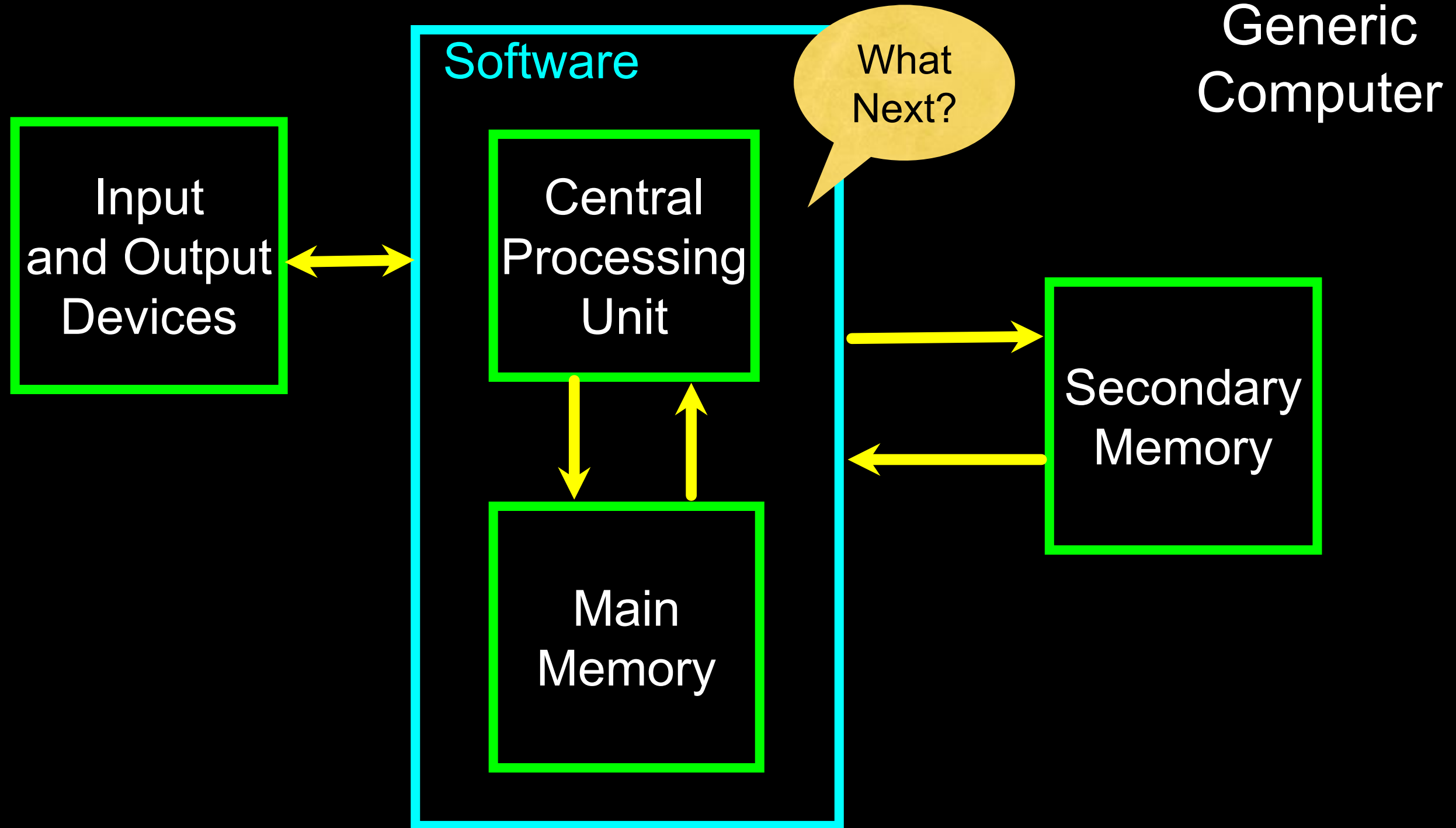
bigcount = None
bigword = None
for word, count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

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

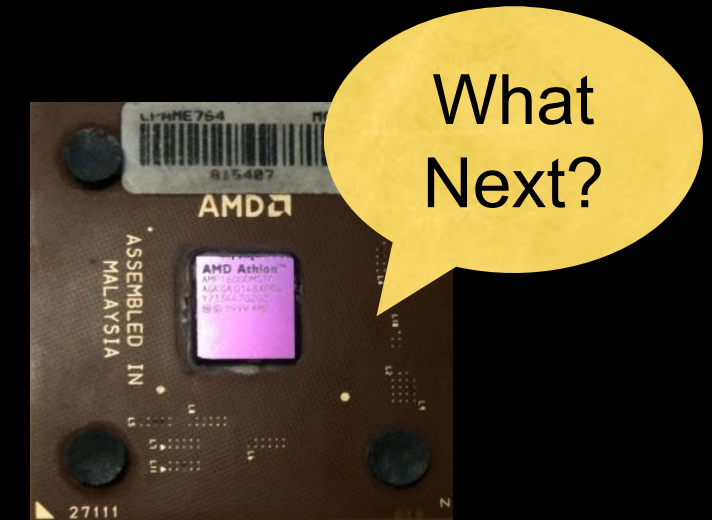
python words.py
Enter file: clown.txt
the 7

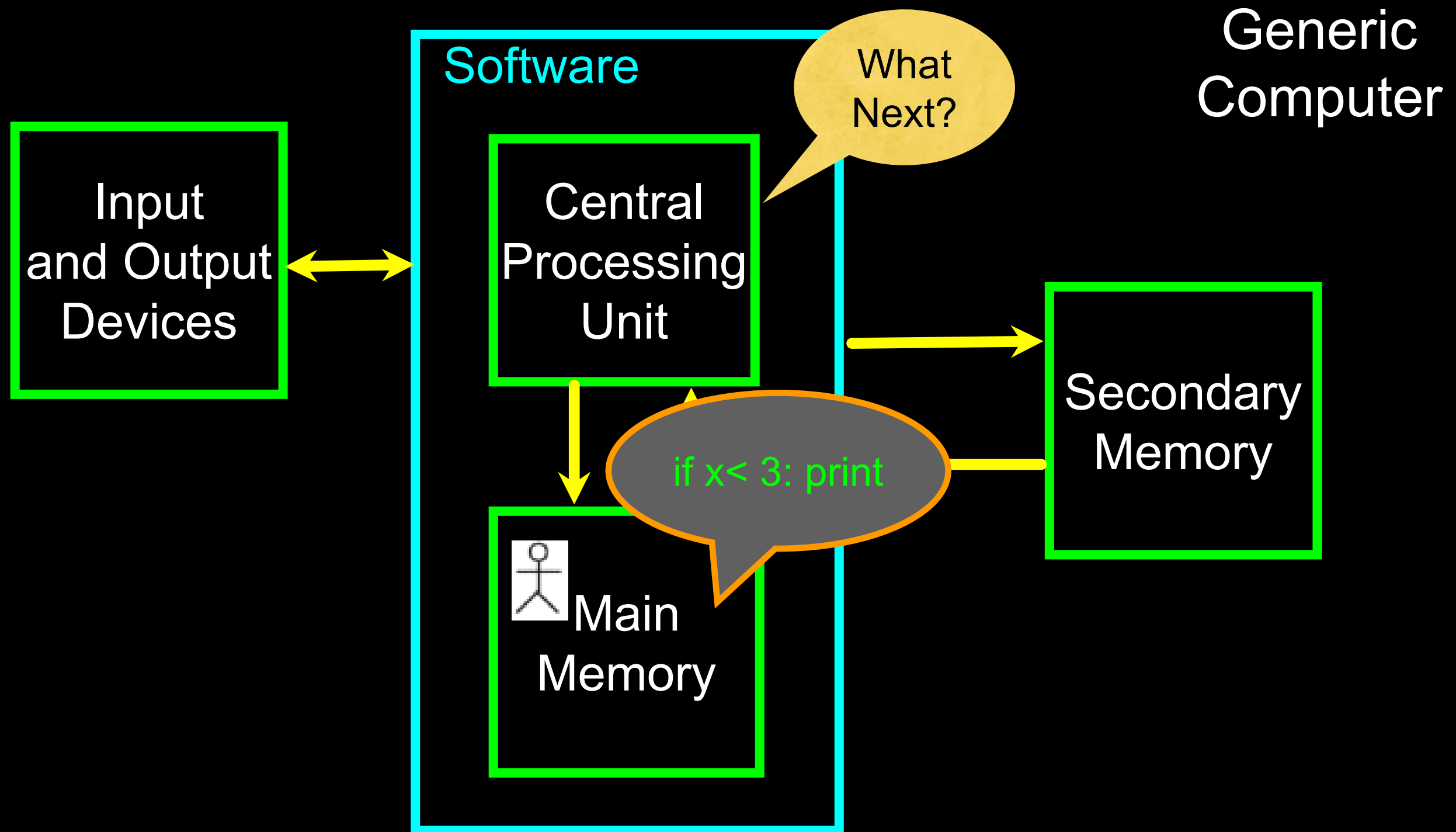
Hardware Architecture

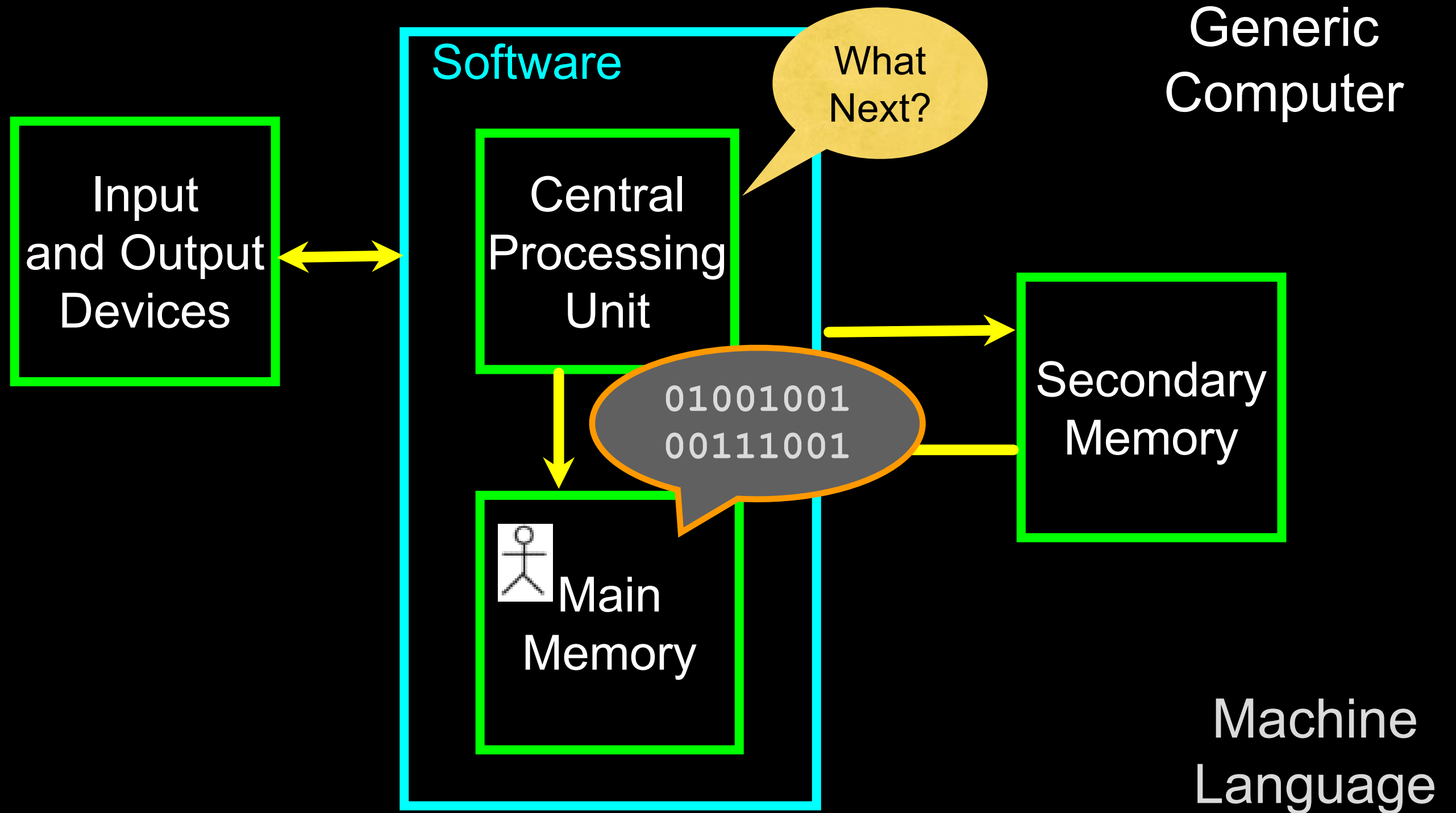


Definitions

- **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







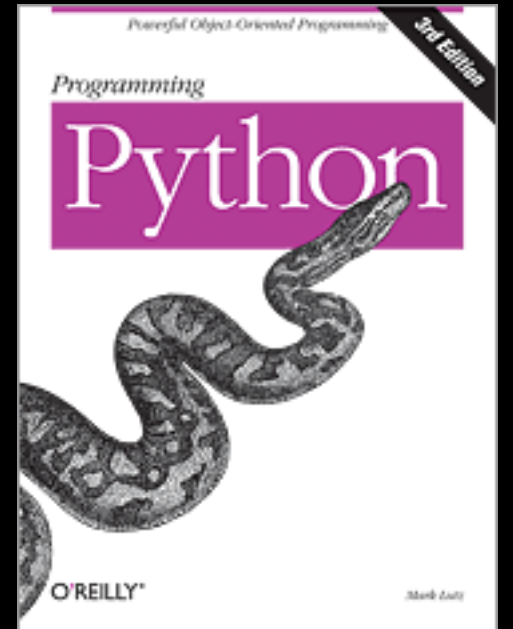
Python as a Language

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$ python3
```

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 5 2015, 21:12:44) [GCC 4.2.1 (Apple  
Inc. build 5666) (dot 3)] on darwinType "help", "copyright", "credits" or "license"  
for more information.
```

```
>>>
```



```
Microsoft Windows [Version 10.0.17763.1158]
```

```
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\cl_on>python
```

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit  
(AMD64)] on win32
```

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

```
>>>
```



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

```
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

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 = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    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

| | | | | |
|--------|--------|--------|--------|----------|
| False | class | return | is | finally |
| None | if | for | lambda | continue |
| True | def | from | while | nonlocal |
| and | del | global | not | with |
| as | elif | try | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

Sentences or Lines

| | | | | | | |
|--------------------|----------------|----------------|----------------|----------------------|-----------------|----------------------------|
| <code>x</code> | <code>=</code> | <code>2</code> | ← | Assignment statement | | |
| <code>x</code> | <code>=</code> | <code>x</code> | <code>+</code> | <code>2</code> | ← | Assignment with expression |
| <code>print</code> | <code>(</code> | <code>x</code> | <code>)</code> | ← | Print statement | |

Variable

Operator

Constant

Function

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.

Interactive versus Script

- Interactive

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

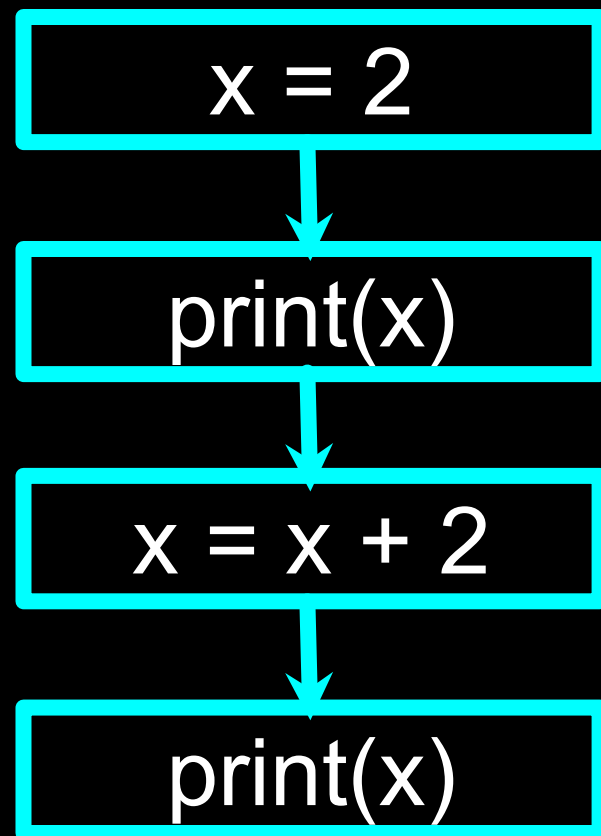
- Script

- 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 is 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)
```

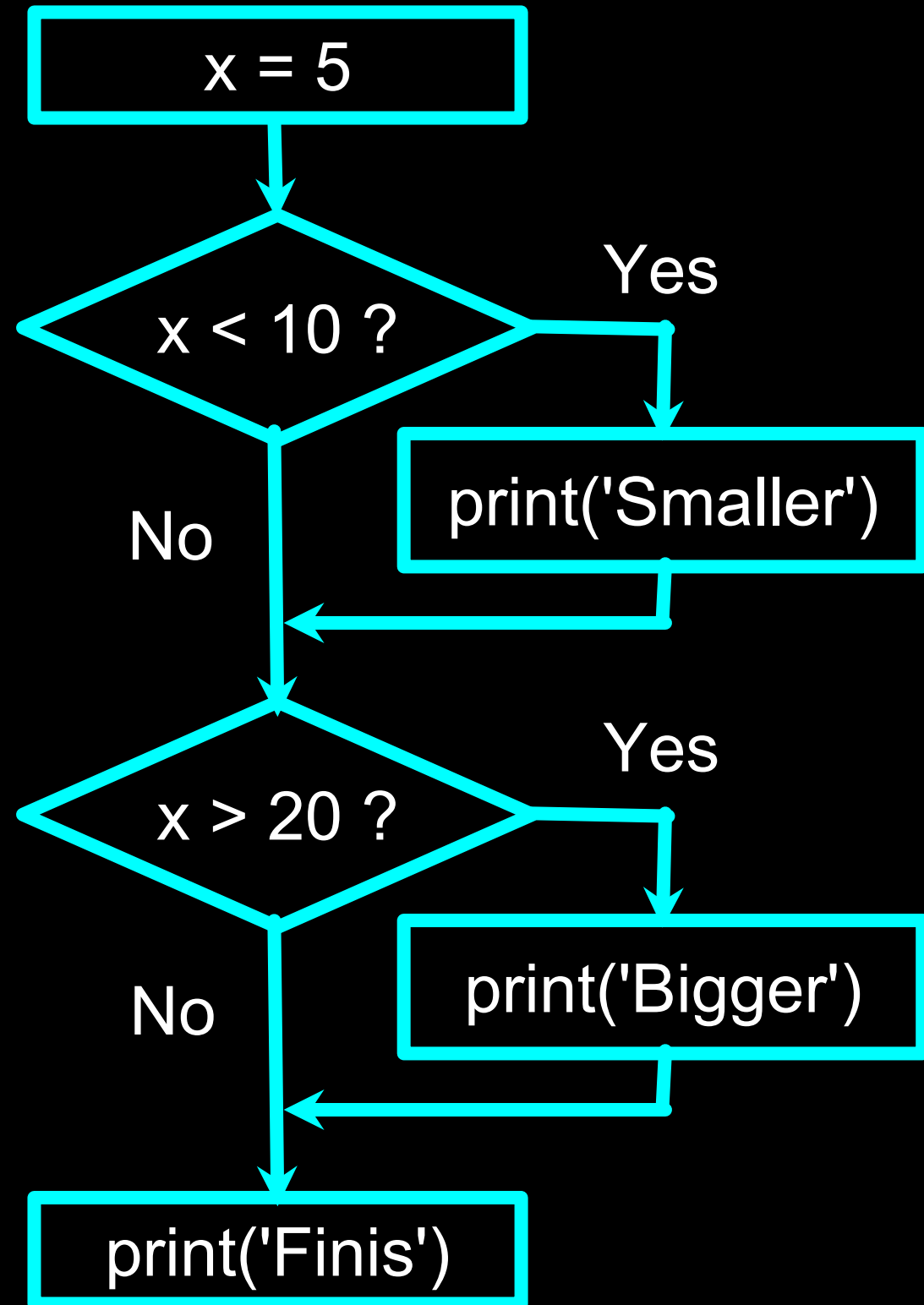
Output:

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:

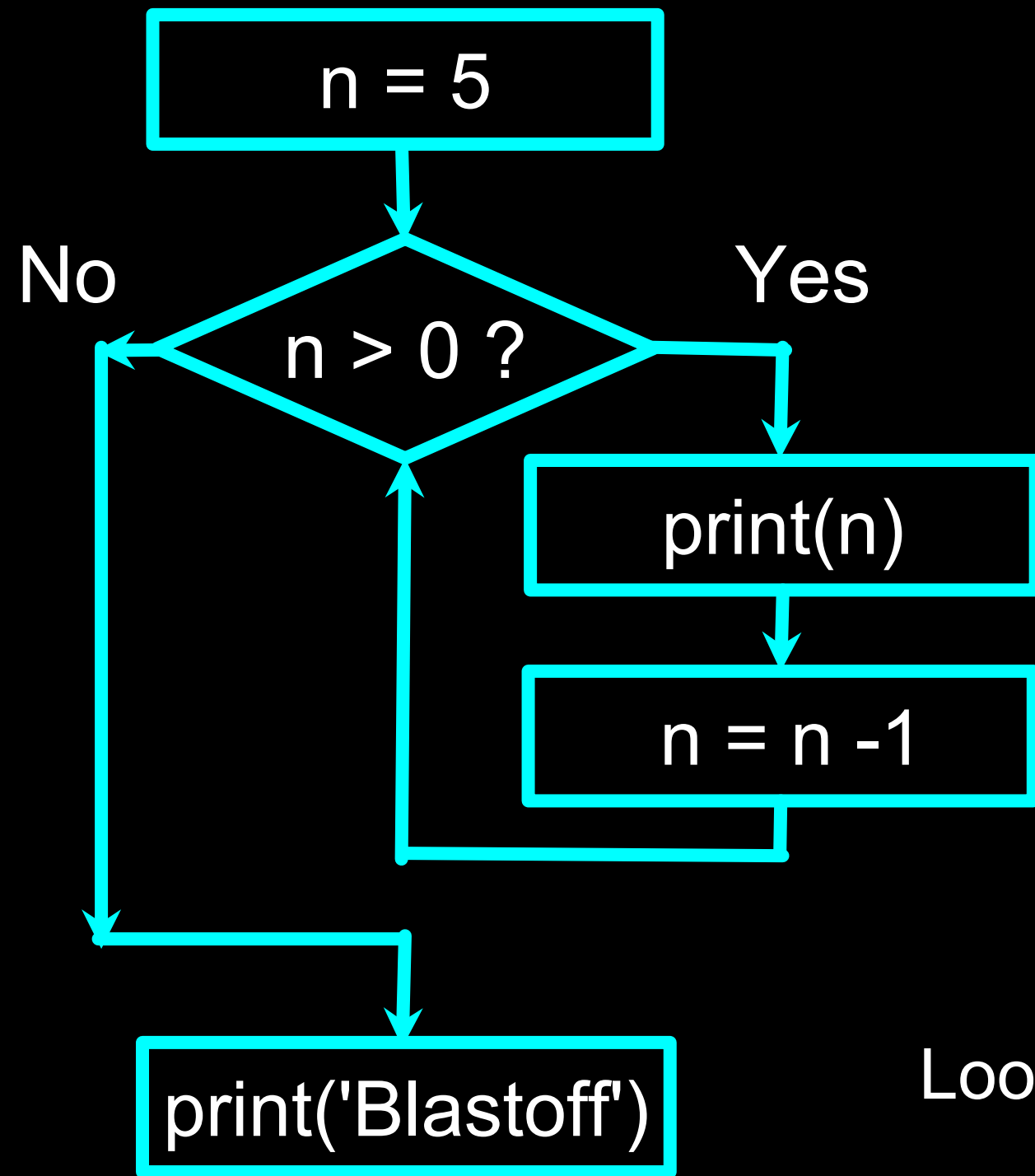
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

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.

```
name = input('Enter file:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    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)
```

Sequential

Repeated

Conditional

```
name = input('Enter file:')  
handle = open(name, 'r')
```

```
counts = dict()  
for line in handle:  
    words = line.split()  
    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) of the University of Michigan School of Information 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

Continue...