Why Program?

Chapter 1

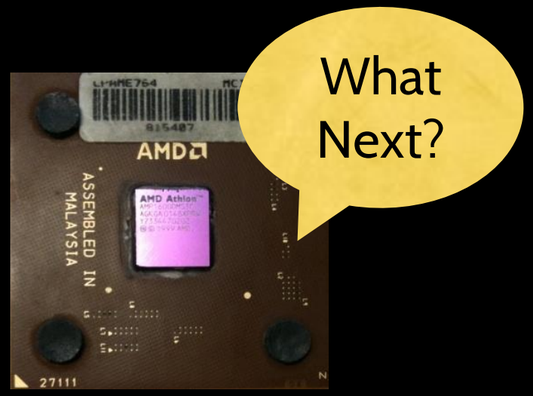
Python for Informatics: Exploring Information
www.pythonlearn.com

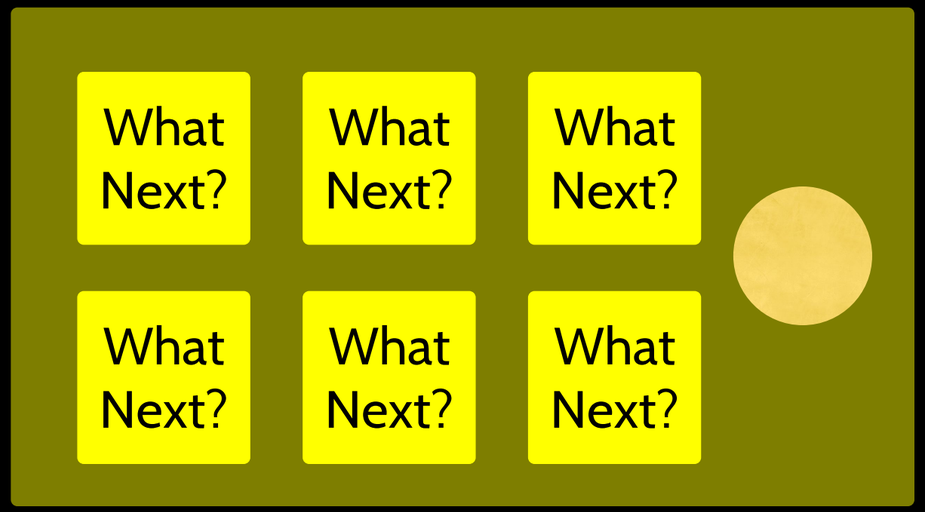
UMSI

CC BY

Computers want to be helpful...

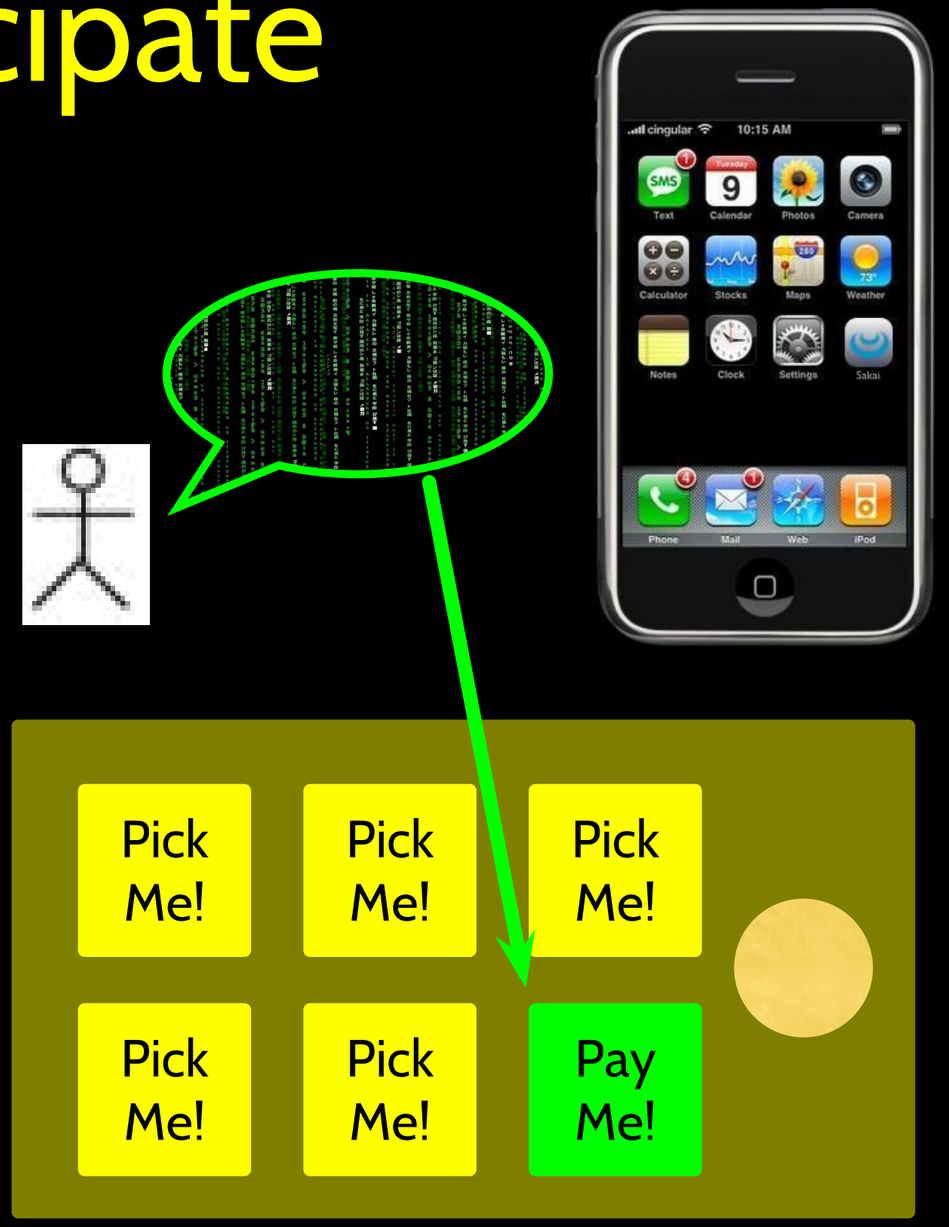
- Computers are built for one purpose - to do things for us
- But we need to speak their language to describe what we want done
- Users have it easy - someone already put many different programs (instructions) into the computer and users just pick the ones we want to use





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

What Programmers Use

- Programmers use the following:
  - Text Editor:
    - Sublime Text
  - Interface:
    - Mac OS or PC
  - Repository:
    - GitHub

What Programmers Use

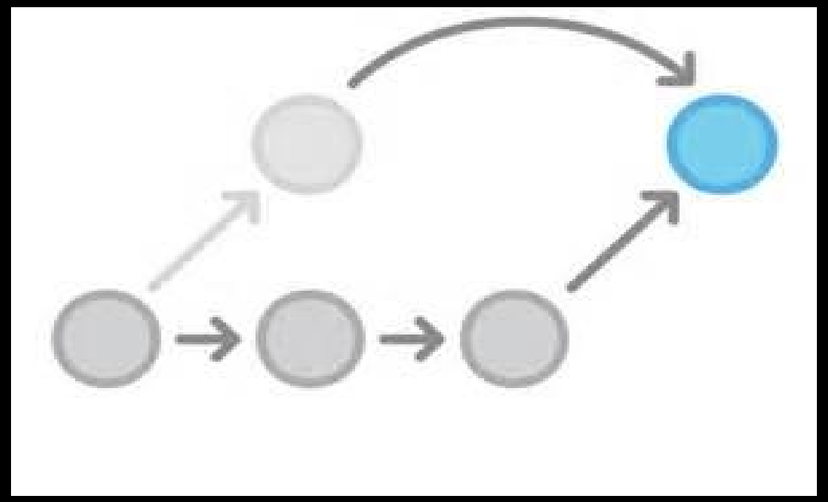
- Programmers use the following:
  - Text Editor:
    - Sublime Text
  - Interface:
    - Mac OS or PC
  - Repository:
    - GitHub

KNOWLEDGEHOUSE

Git

- Git is a source control management tool.

- Git allows you to store and update your code in a structured way.



- Git includes history of changes you make, so you can create "checkpoints" and track your work better over time.

KNOWLEDGEHOUSE

Installing GIT

For windows git can be installed for the cmd line by downloading github for windows.

http://windows.github.com

For Mac

http://mac.github.com

For Linux:

sudo apt-get install git

KNOWLEDGEHOUSE

Installing GIT via Terminal

Type: git config --global user.name “YOUR NAME”

- Assures that your commits are under your name

Type: git config --global user.email “YOUR EMAIL ADDRESS”

- Make sure it’s the same email address that you used to create GitHub

KNOWLEDGEHOUSE

GitHub

GitHub is a service that lets you host Git repositories in the cloud.

KNOWLEDGEHOUSE

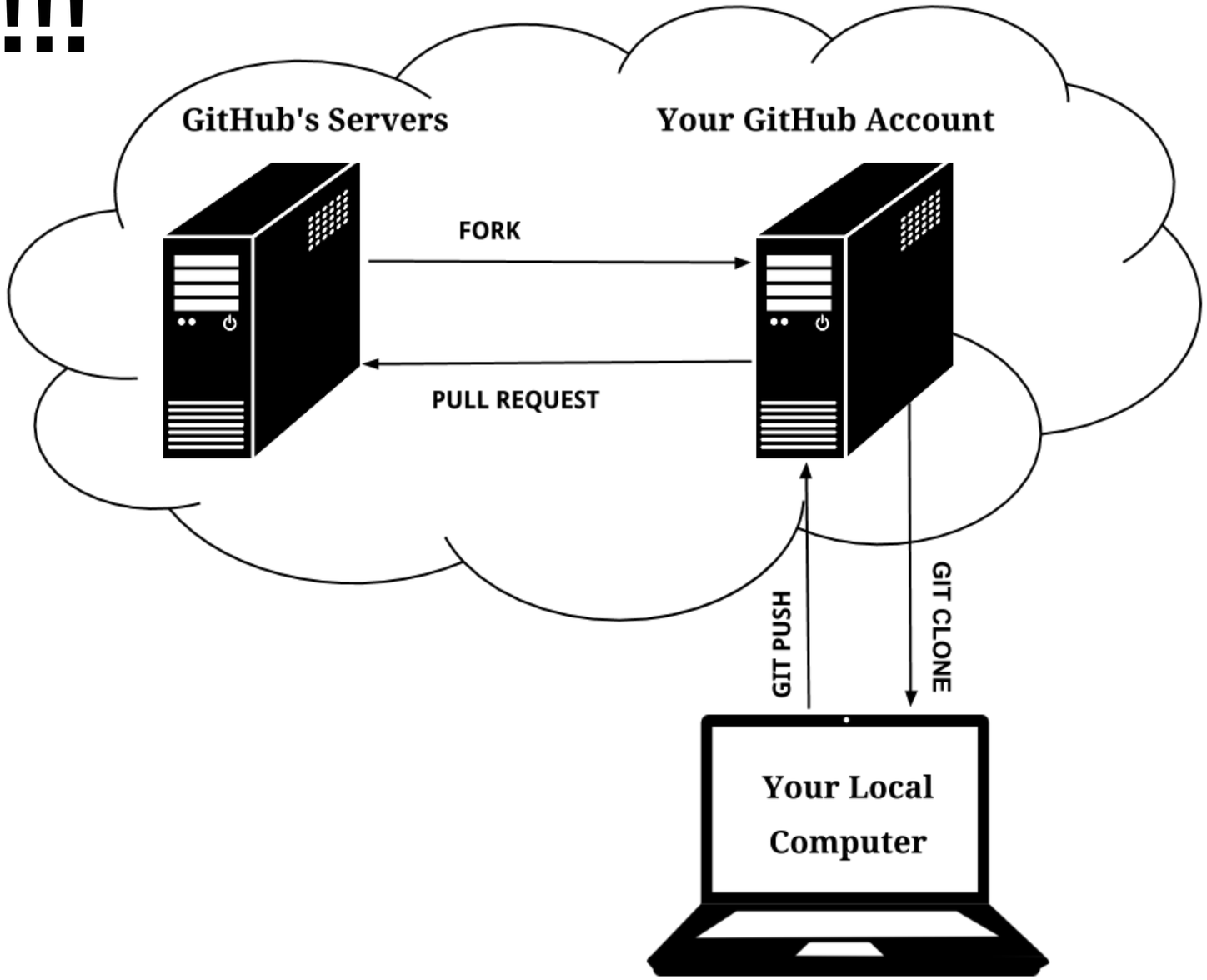
GitHub: Benefits

Distribute code to others by sharing your repository

View your code online easily with a web interface

Free Public Use!!!

Forking!!!


GitHub's Servers
Your GitHub Account
FORK
PULL REQUEST
GIT PUSH
GIT CLONE
Your Local Computer

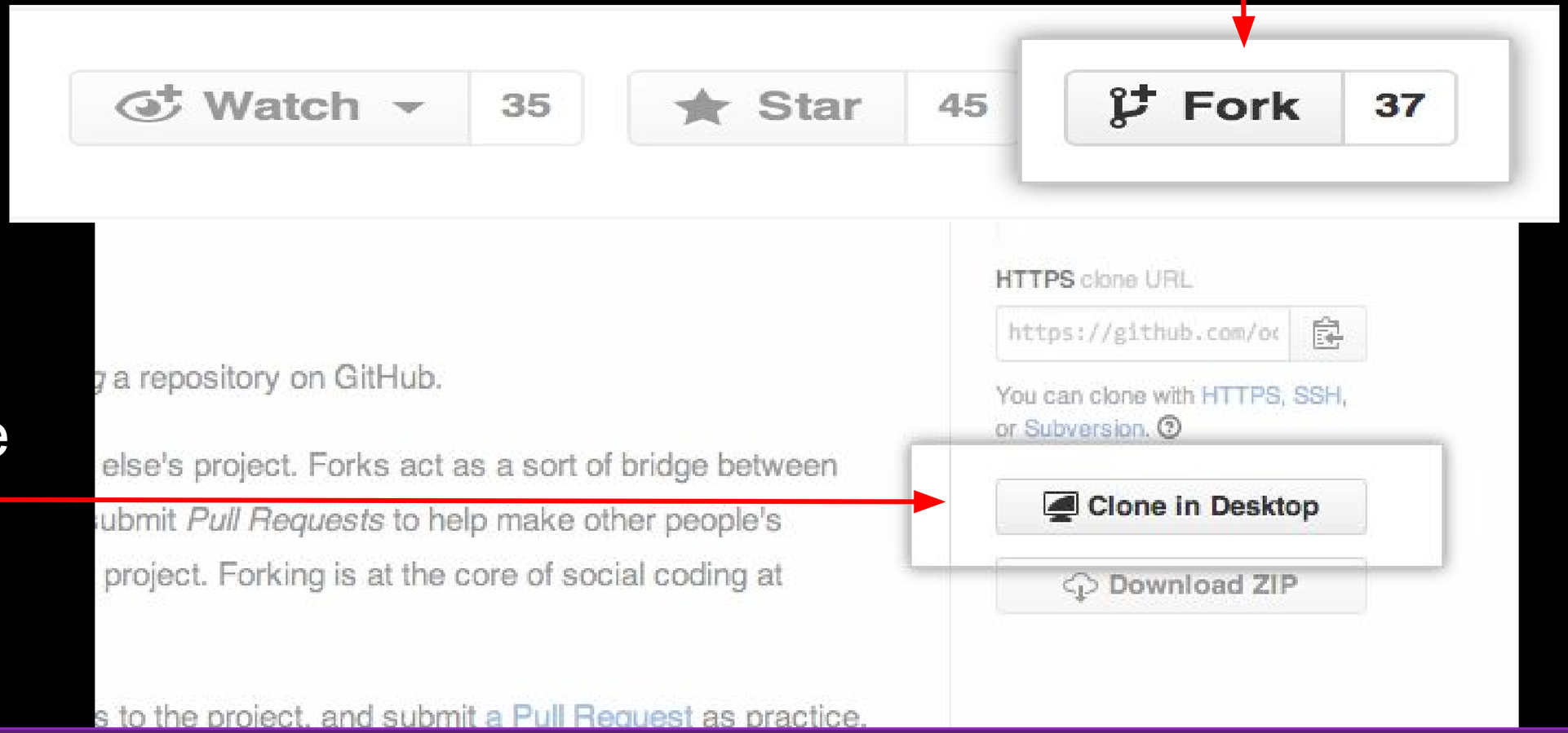
KNOWLEDGEHOUSE

Let's Get Connected

1. Type: the link of the project repository into the URL Search Bar

Let's Fork!

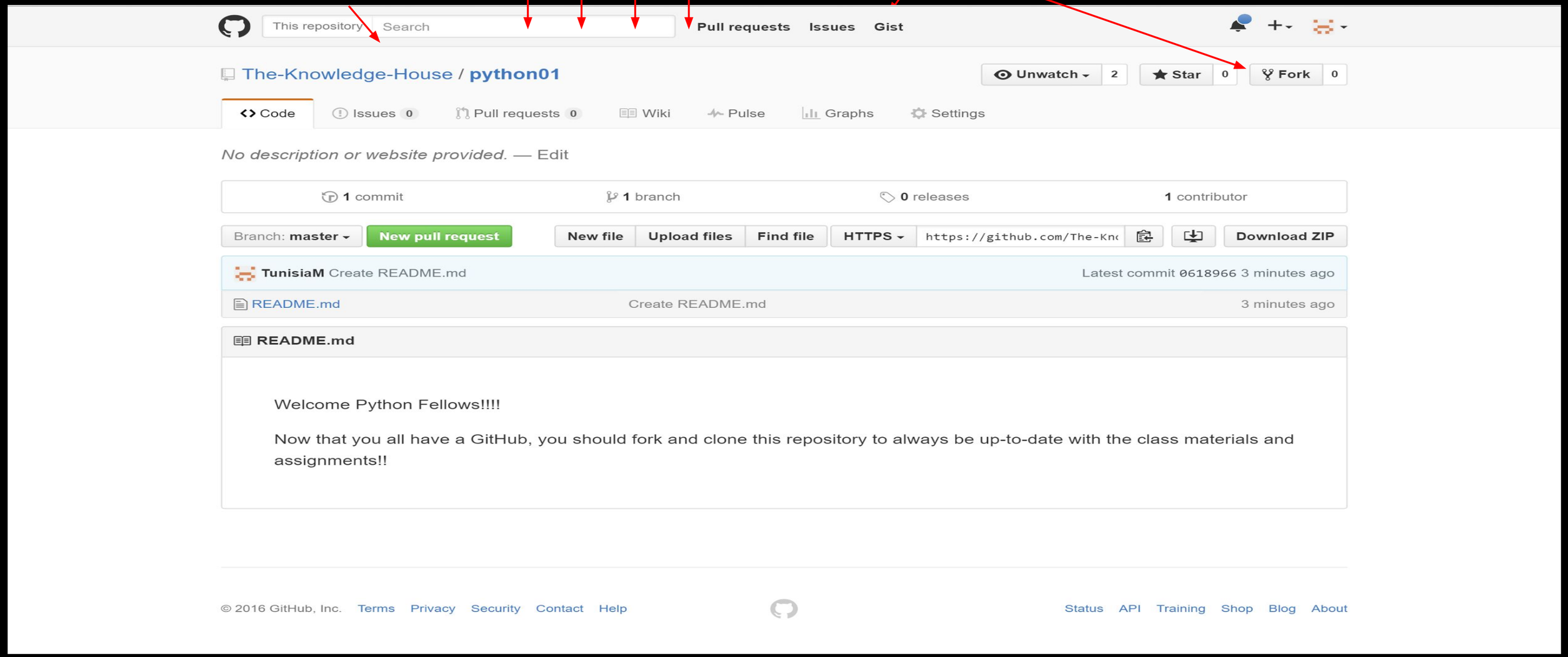
2. Click Fork



3. Let's Clone the Fork!

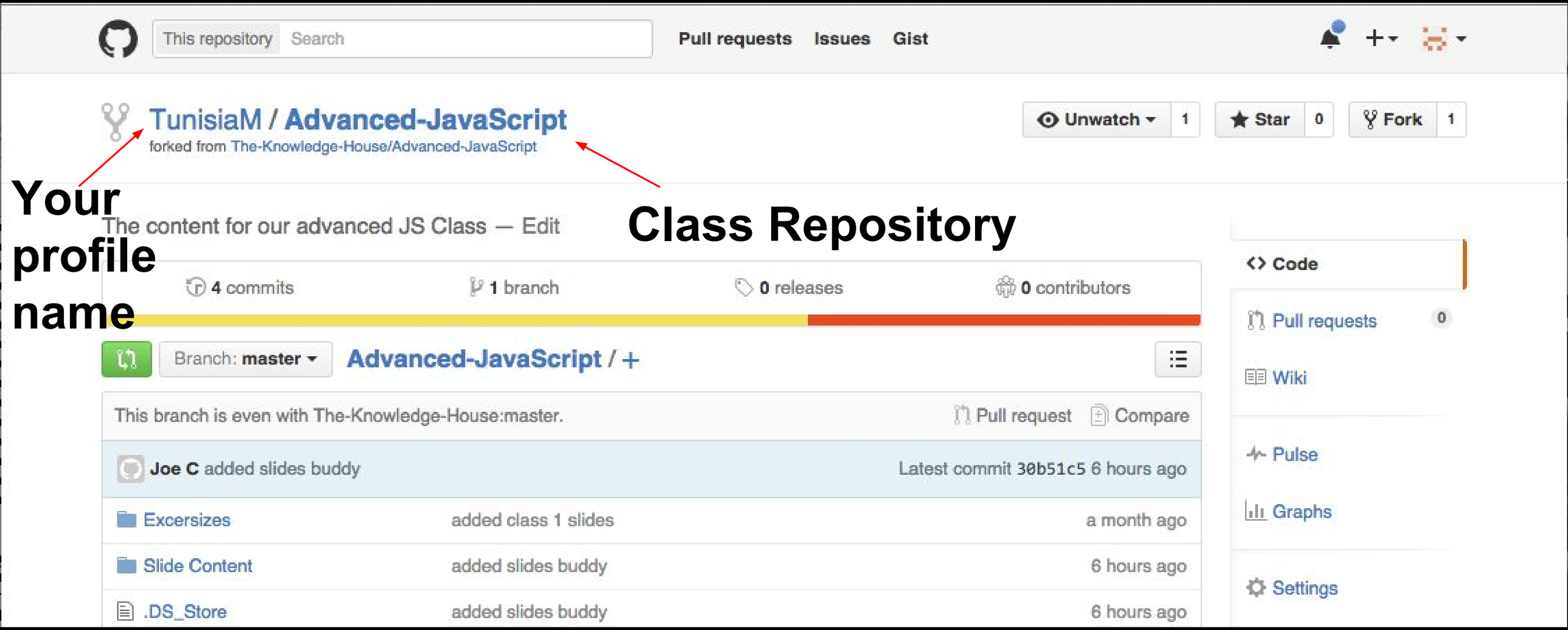
KNOWLEDGEHOUSE

Step 1 and 2: Type Link and Fork



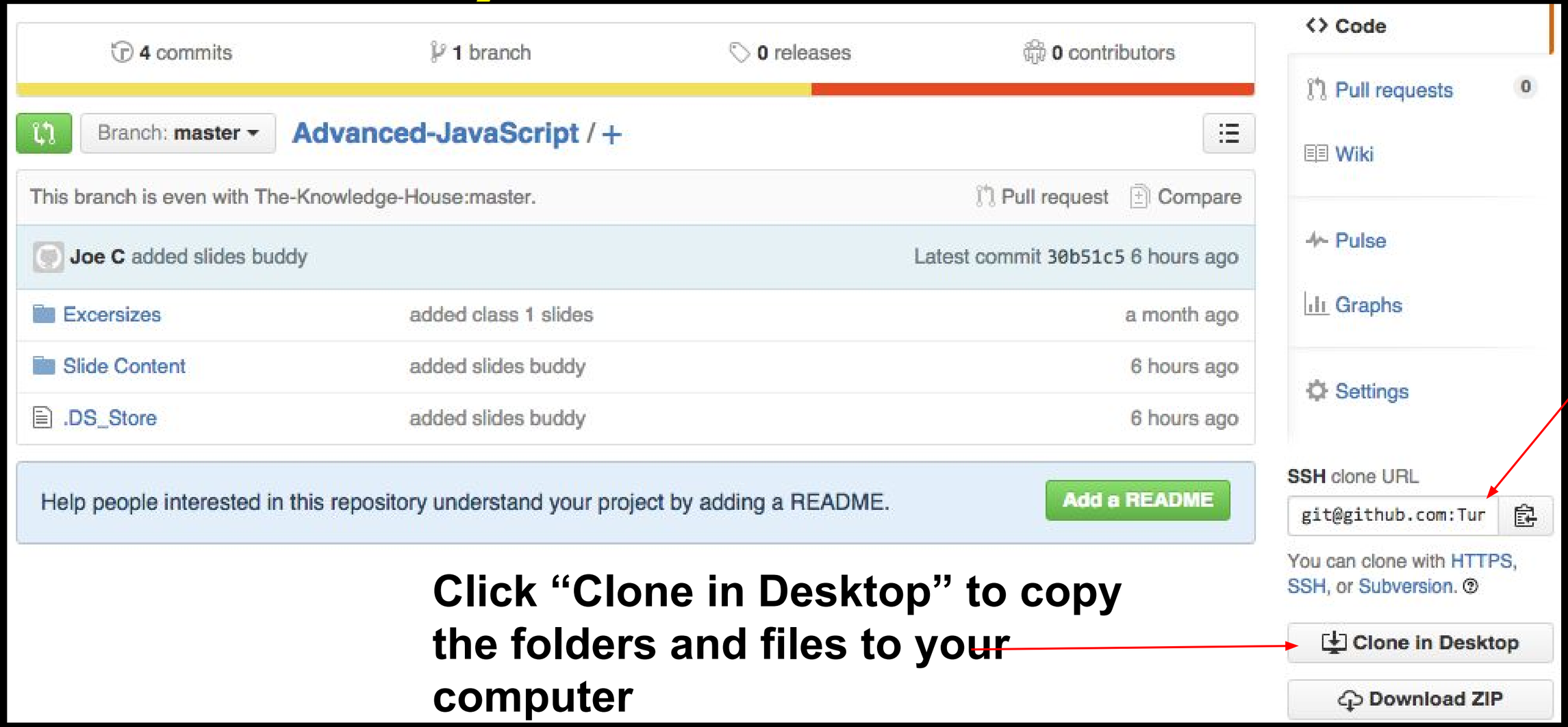
KNOWLEDGEHOUSE

Result



KNOWLEDGEHOUSE

Two Ways to Clone



Type: git clone "SSH clone URL"

Click "Clone in Desktop" to copy the folders and files to your computer

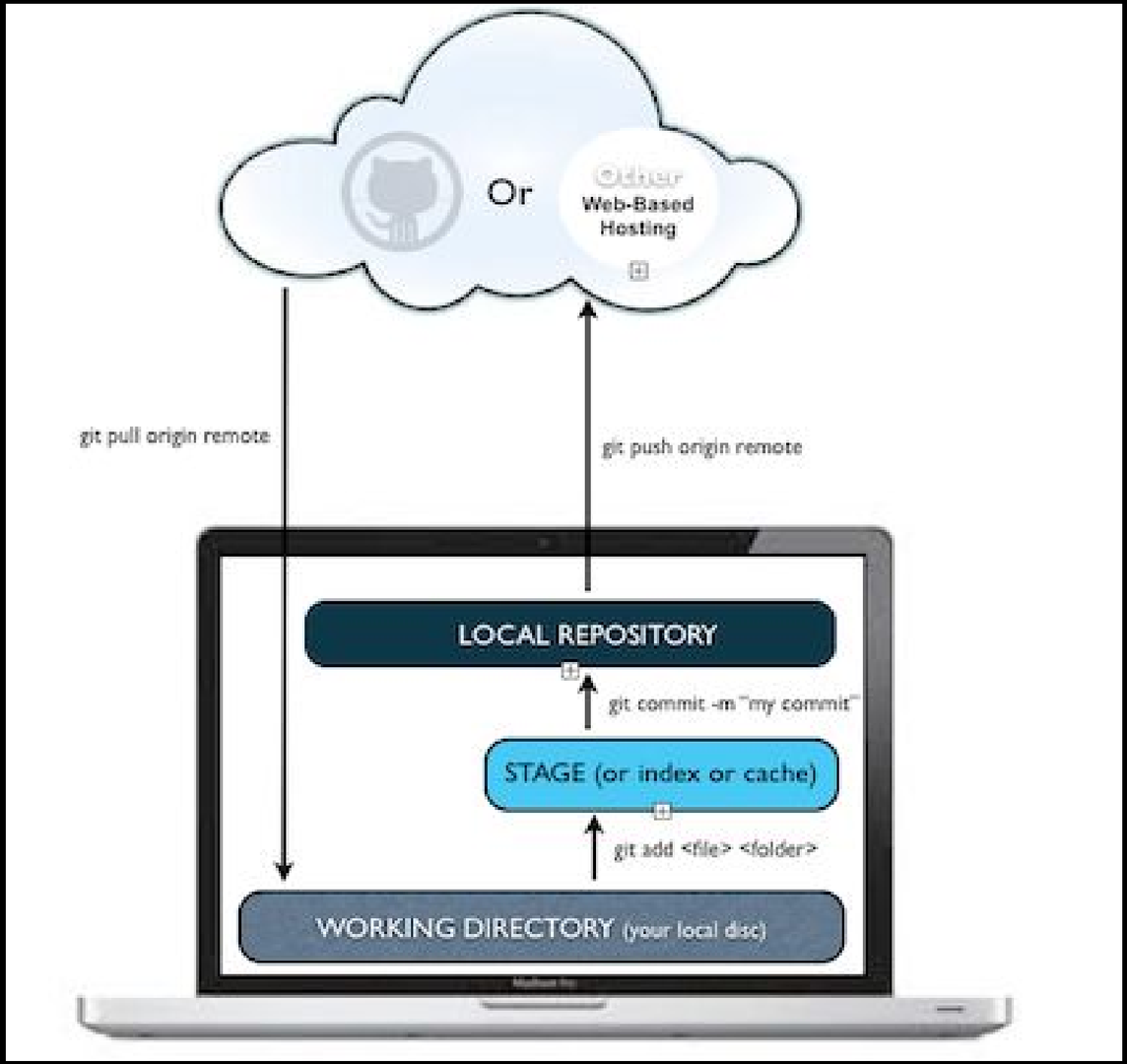
KNOWLEDGEHOUSE

Cloning a local project to GitHub

- Create new repo on github
- Cd into the directory
- Initialize a git repo in the directory: git init
- Add & Commit : git add . , git commit -m “first commit”
- Connect to your repo: git add remote origin <remote repo url>
- Verify URL: git -v
- Push: git push origin master

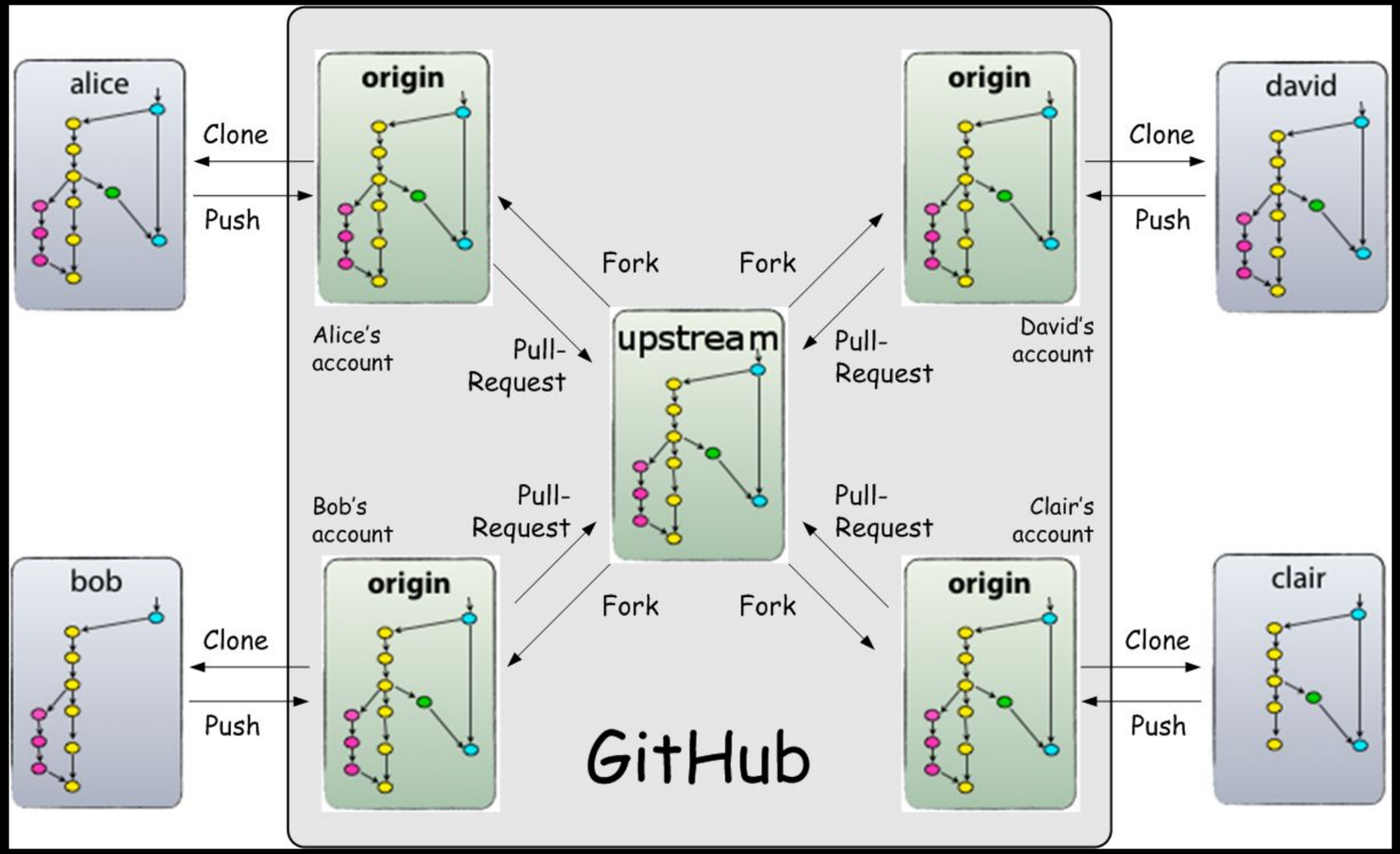
KNOWLEDGEHOUSE

GitHub Workflow



KNOWLEDGEHOUSE

GitHub Workflow w/ Teammates



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

```
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
```

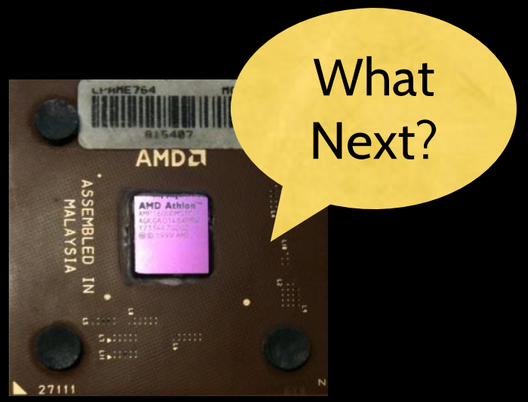
```
python words.py
Enter file: words.txt
to 16
```

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

http://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

Definitions

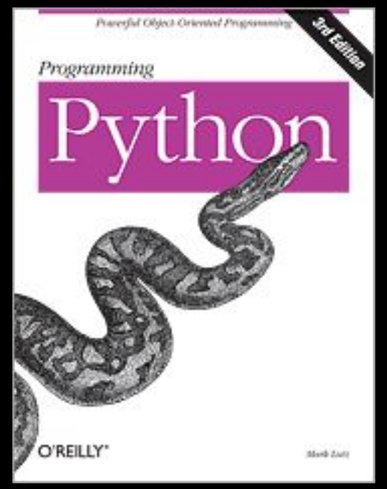
- Central Processing Unit: Runs the Program - The CPU is always wondering “what to do next”? Not the brains exactly - very dumb but very very fast



- 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

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 Pythonist. It is a very uncommon skill, and may be hereditary. Nearly all known Pythonists 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...

```
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.

Python Download Help!!

If you don't have Python Installed, go to the following links:

Windows: https://www.python.org/downloads/

Mac OS: https://www.python.org/downloads/mac-osx/

Linux/UNIX: https://www.python.org/downloads/source/

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
x = x + 2
print x
```

← Assignment statement

← Assignment with expression

← Print statement

Variable

Operator

Constant

Reserved Word

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.

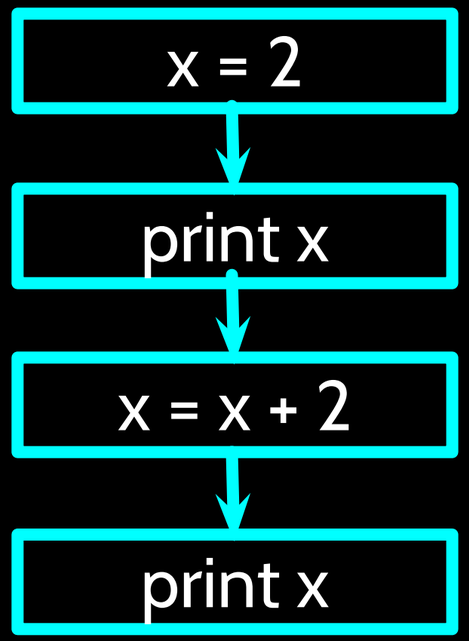
Interactive versus Script

- 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 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:

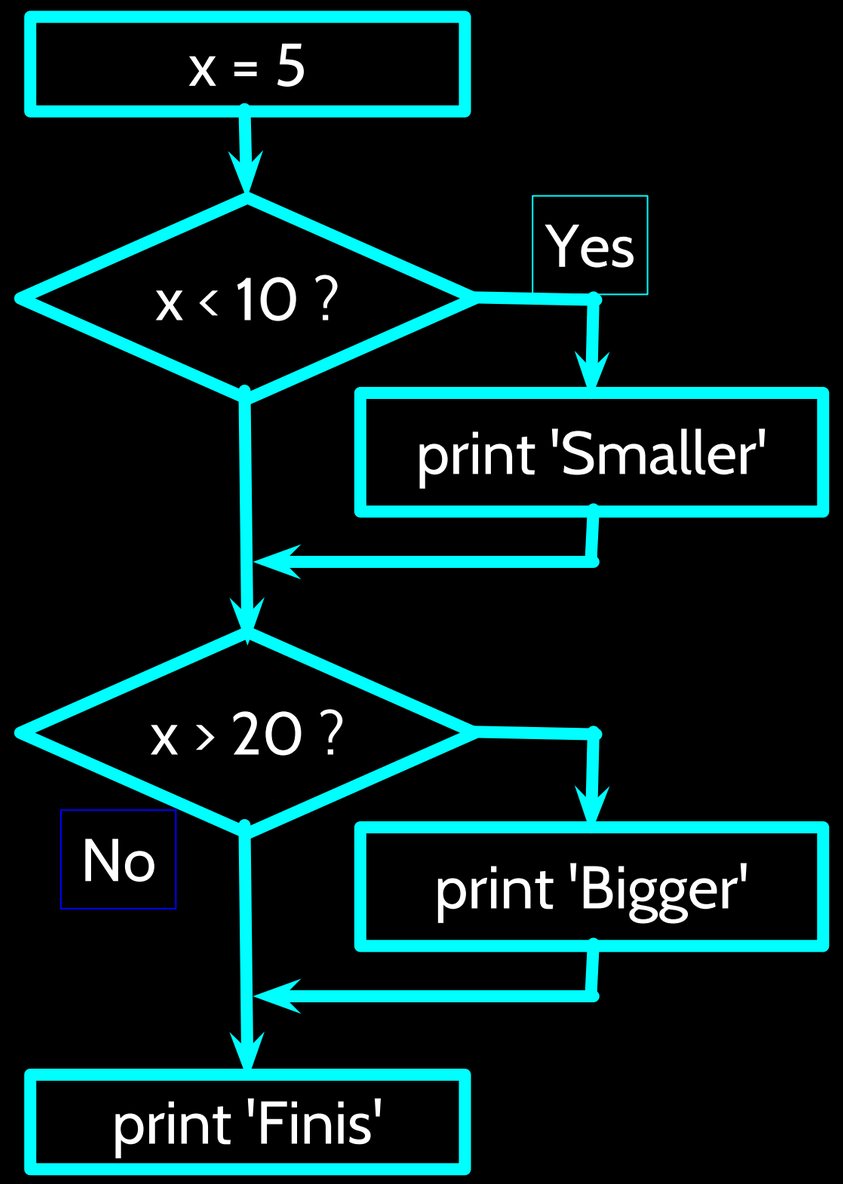
```
x = 2
print x
x = x + 2
print x
```

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



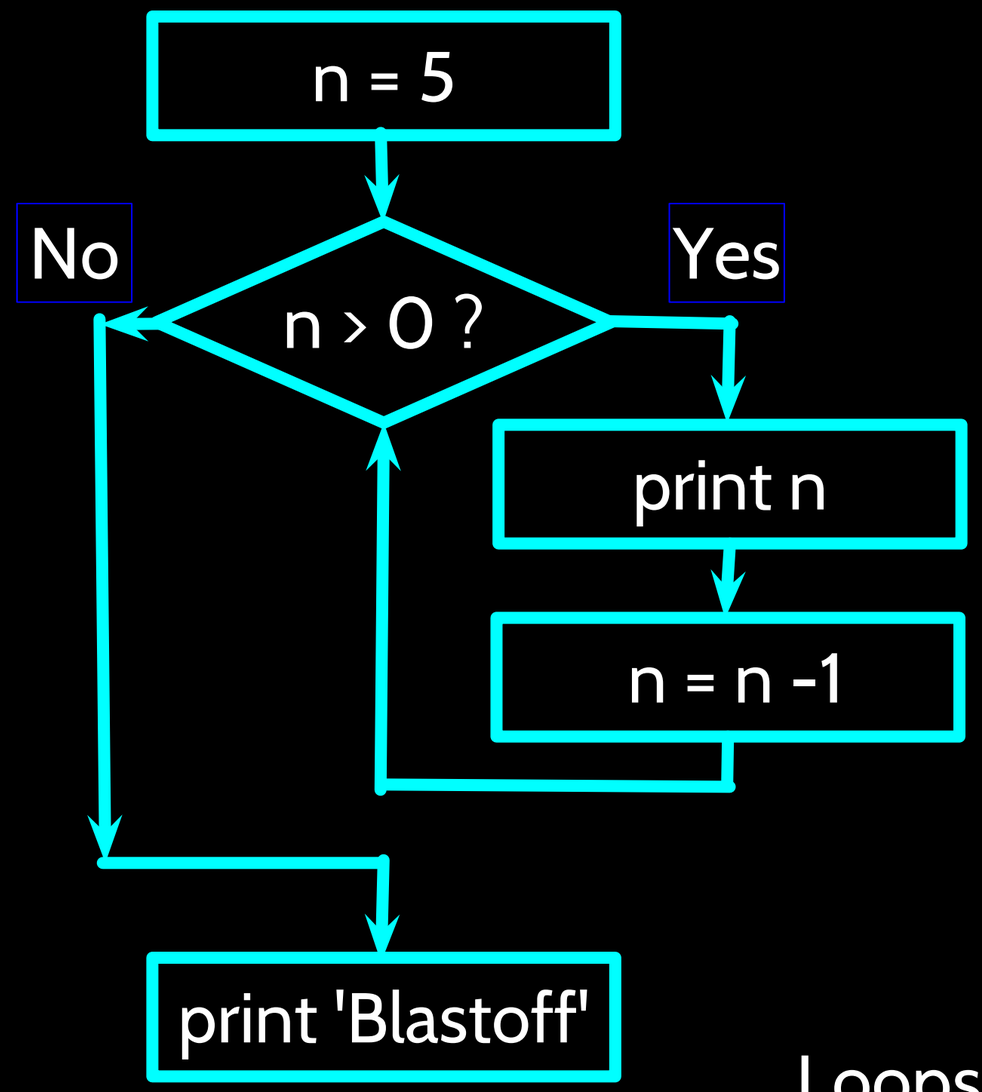
Program:

```
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()

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
```

Sequential

Repeated

Conditional

```
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) of the University of Michigan School of Information and 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

...