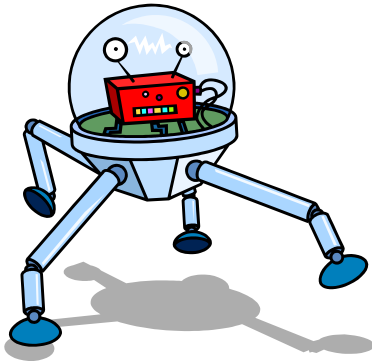
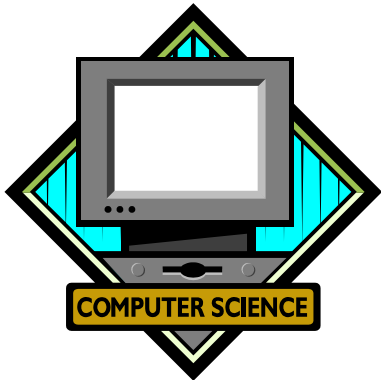
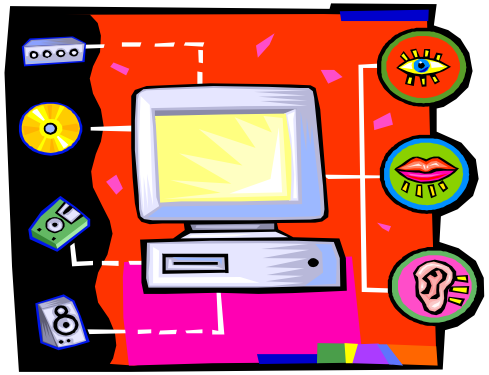
Caltech/LEAD Summer 2012 Computer Science





Lecture 1: July 10, 2012

Orientation and basics



First...

- Welcome to Caltech!

Outline

- How the course is organized
- About Computer Science
- Computer basics

LEAD
Leadership Education And Development

Course organization

People

- Instructors:
  - Mike Vanier (me)
  - Donnie Pinkston
- Our email addresses:
  - Mike: mvanier@cs.caltech.edu
  - Donnie: donnie@cs.caltech.edu

People

- Teaching assistants:
  - Max Hirschhorn
  - Jesse Salomon
- Their email addresses:
  - Max: mhirschh@caltech.edu
  - Jesse: jsalomon@caltech.edu

Course Website

- http://courses.cms.caltech.edu/lead/
- Nothing much there yet!
- We will post
  - all lecture slides (as PDF files)
  - all lab assignments
  - suggestions for projects
  - useful links to references
  - anything else we/you think would be useful

Organization

- Typical daily schedule:
  - 9-10 AM: lecture
  - 10 AM-noon: lab time
  - 1-2 PM: lecture
  - 2-4 PM: lab time
  - 6-8 PM: optional lab time if you want/need it
- Variations:
  - Some evenings we will have guest lectures (deeper discussions of particular CS areas from Caltech faculty)
  - Field trip to Google on July 16th

Organization

- First 2 weeks will be primarily lectures and lab work
- Last week will be for student projects

Textbook

- No required textbook
- If you want to buy one, we recommend **Practical Programming: An Introduction to Computer Science Using Python** by Jennifer Campbell et al
- Lots of other good Python books too!

The Pragmatic Programmers

Practical Programming

An Introduction to Computer Science Using Python

Jennifer Campbell
Paul Gries
Jason Montojo
Greg Wilson

Edited by Daniel H. Steinberg

Your Part

- Come to lectures
- Work on problem sets in lab
- Attempt every problem, even if you can't get it right away!
- Don't be afraid to ask us for help!
- Submit completed work through *csman* (http://csman.cs.caltech.edu)

Grading homework

- A problem set usually has multiple parts (*e.g.* A through C)
- Each part receives a grade between 0 and 3:

0	*incorrect (worthy of no credit)*
1	*insufficient (not passing quality, demonstrates significant bugs which must be addressed)*
2	*good (demonstrates mastery of key idea, may have a few minor bugs)*
3	*excellent (masters all important parts)*

Time on problems

- Taking excessive time on a problem is a symptom that you missed an important concept
  - It's possible to do things the wrong way (missing what we're teaching you)
    - …but it probably will take you a lot more time
- Don't spend hours on a single problem!
- If/when you make mistakes, you can rework your problem sets and resubmit up to 2 times per set

Grading philosophy

- This course is not about getting good grades!
- Grades are just our way of letting you know how well you're doing
- This course is for *your* benefit
  - take advantage of our knowledge to learn as much as you can in the time you have

About us...

- We've been teaching computer programming courses at Caltech for many years
- This is only our 2nd time teaching a LEAD course
- Let us know what works for you and what doesn't!

About you...

- Let's hear about what *you* want to get out of this course!

LEAD
Leadership Education And Development

About Computer Science

What is computer science?

- What do *you* think computer science is all about?
- In this course, we will mainly be teaching you about *computer programming*
- However, computer science is about much more than this

What is computer science?

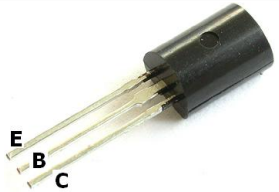
- Computer science (CS) is about understanding what "computation" means and how it works
- And also: how to make computations do useful work for us
- Computer "science" is both a science and an engineering discipline

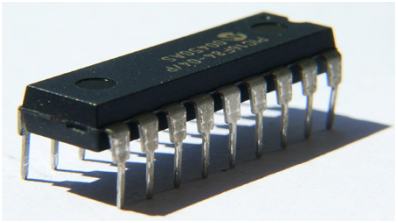
History of computers

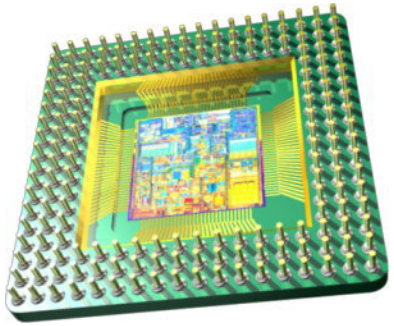
- Computer science is a *very* young field
  - much younger than math, physics, chemistry, biology
- CS didn't even exist before 1930s
  - because there were no computers!
- First computers built in late 1930s, early 1940s

History of computers

- Vacuum tubes (1940s)
- Transistors (1950s)
- Integrated circuits (1960s)
- Large scale integrated circuits (1970s – present)



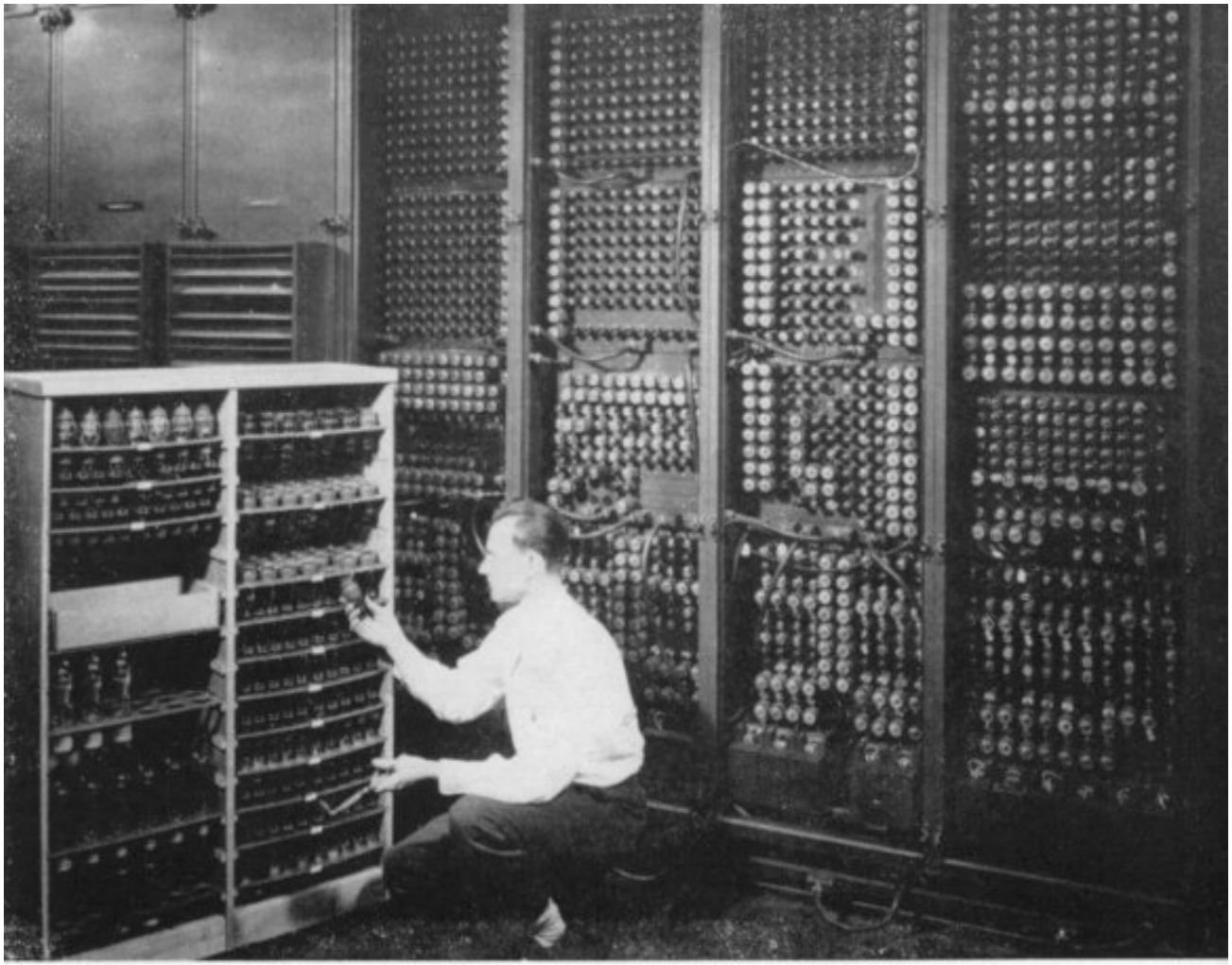




History of computers

- Early computers used thousands of vacuum tubes, filled entire rooms, and could only do very simple calculations
- As the technology progressed, computers got smaller and smaller and more and more powerful
- Now any phone is more powerful than the most powerful computer in the world in the 1950s

ENIAC (an early computer)



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Video clip!

- A blast from the past

Anatomy of a computer

- Modern computers come in many different shapes and sizes

Anatomy of a computer

- Desktop computers

Anatomy of a computer

- Laptop computers

Anatomy of a computer

- Touch-screen "pad" computers

Anatomy of a computer

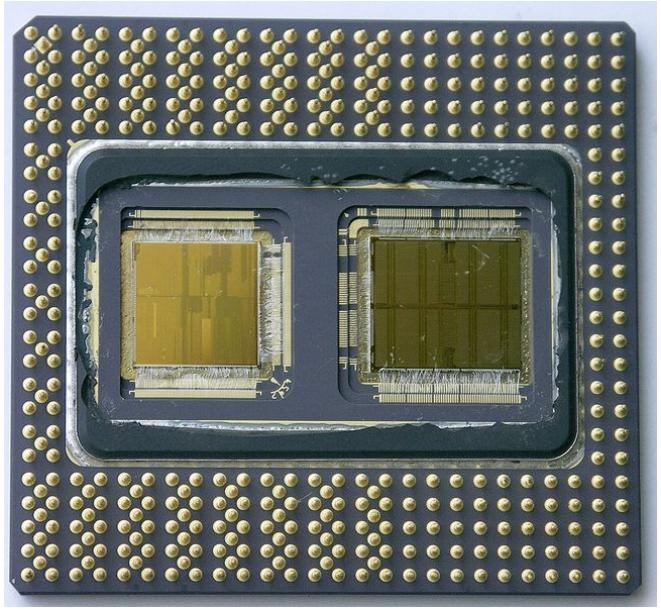
- Smart phones

Anatomy of a computer

- Whatever their appearance, internally all computers contain similar components

Anatomy of a computer

- There must be a central processing unit (CPU or just "processor") which does the actual computations



Anatomy of a computer

- There must be computer "memory" for fast, short-term storage of information being worked on

Anatomy of a computer

- There must be a "disk drive" for slower but permanent storage of information being worked on (photos, videos, data files)

Anatomy of a computer

- There must be a display so that you can see what the computer is doing

Anatomy of a computer

- There must be a way to input data and instructions to the computer

Anatomy of a computer

- or:

Anatomy of a computer

- There must also be a way for the computer to access the world-wide computer network (the *internet*)
- There may be other peripherals that allow the computer to get different kinds of input or send different kinds of information out
  - *e.g.* a touchpad for input, a sound card for audio output

What is programming?

- A computer program is a *sequence of instructions* that tell the computer how to perform some particular task or set of tasks
- Computers only directly understand a "machine language" which is extremely primitive
  - *e.g.* add a number to a location in memory
- It's possible to program directly in machine language, but it's no fun!

What is programming?

- Managing all that complicated hardware in order to get the computer to do something is a complex task!
- If you had to tell every part of the computer exactly what to do, it would be almost impossible
- To make it practical to get computers to do what you want them to do, computer *programming languages* have been developed

What is programming?

- Over several decades (starting in the 1950s), "high-level" programming languages have been developed which make it much easier to tell the computer how to solve problems
- You don't have to worry about the details of displays, disk drives, memory, CPUs
  - or at least, not much!
- Just write the instructions to solve the problem!

Why learn programming?

- Programming is one of the most practical skills you can ever learn!
- Business, science, entertainment, government, and just about everything else require computers to enable them to do their work effectively
- Thus, there are lots of jobs for computer programmers available
  - and even a few good ones! ☺

Why learn programming?

- Programming is also the gateway into the field of computer science
- Much of computer science evolved out of the problems of
  - people trying to write programs to solve problems
  - people trying to understand the programs they had written!

Why learn programming?

- Programming leads you to almost every area of computer science
  - computer hardware
  - algorithms
  - graphics
  - networking
  - operating systems
  - computer security
  - user-interface design (human-computer interaction)
  - theory of computation

Why learn programming?

- Computer science is not just about programming!
- ...but most areas of computer science wouldn't exist if it weren't for programming

Why learn programming?

- There is one other reason to learn programming that hardly anyone will tell you
- It's a lot of fun!
- Programming is a very satisfying intellectual activity (like solving puzzles)
- It's also a satisfying *design* activity when done well (like creating art)
- We hope you'll learn to enjoy it like we do!

LEAD
Leadership Education And Development

Computer Basics

Computers we'll be using

- The computers we'll use for this course are "Windows" computers, graciously donated by Google

Our sponsor

Computers we'll be using

- These computers are "laptop" computers and contain all the hardware you need (keyboard, touchpad, display, CPU, memory, disk drive, etc.)

Operating systems

- What do we mean when we say "this computer is a Windows computer"?
- There are many different computer hardware companies that make "Windows computers"
- What it means is that the computer is running the Microsoft Windows "operating system"
- What's an operating system?

Operating systems

- We said before that programming would be too hard if we had to manage every piece of hardware ourselves in order to get anything done
- The operating system's job is to make this easier, for both programmers (writers of programs) and users (people who run the programs created by programmers)

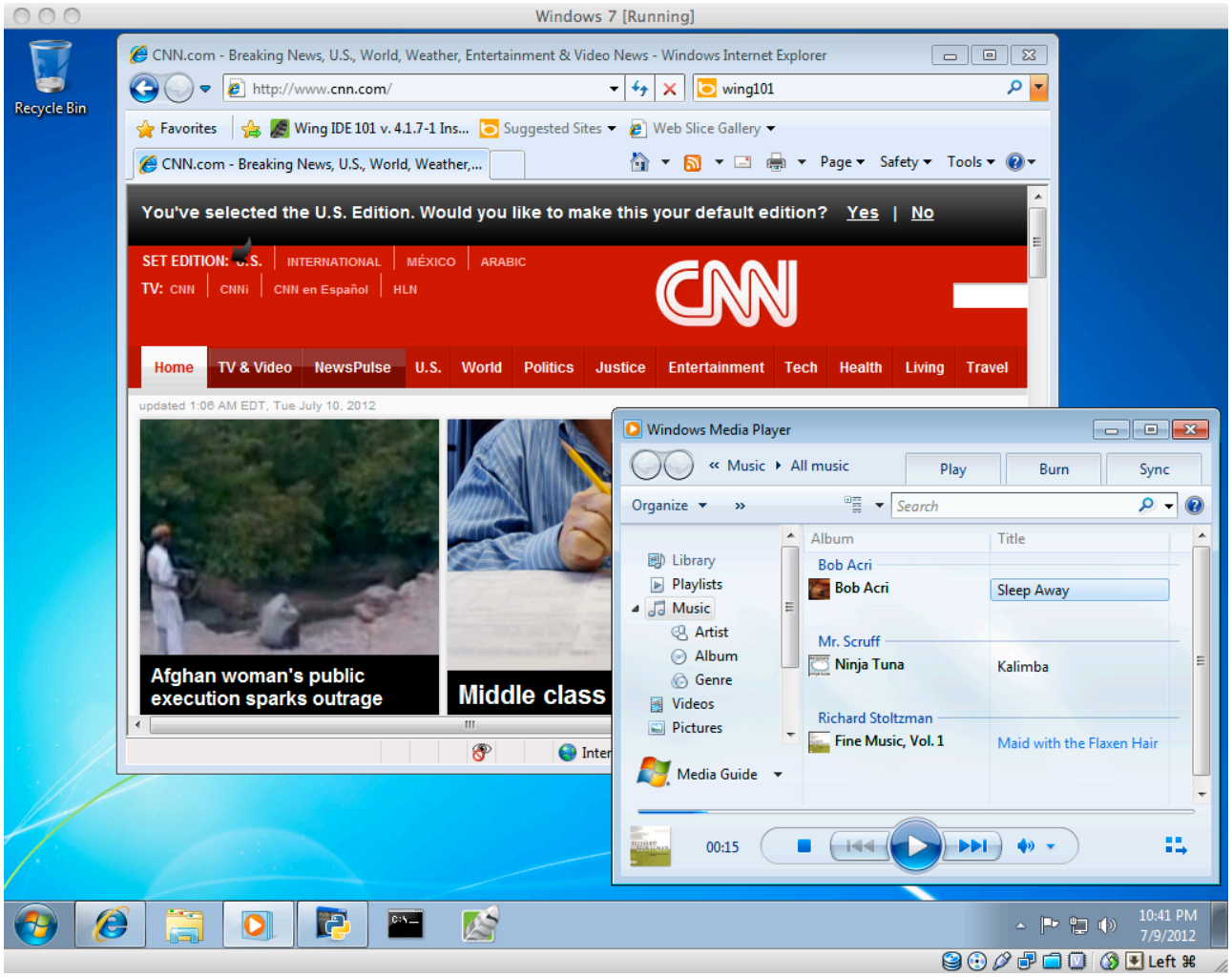
Operating systems

- The operating system (OS) is itself a bunch of programs!
- They handle all the hardware-related aspects of computing, allowing programmers and users to focus on interesting problems
- There are several different (and incompatible) operating systems in use today

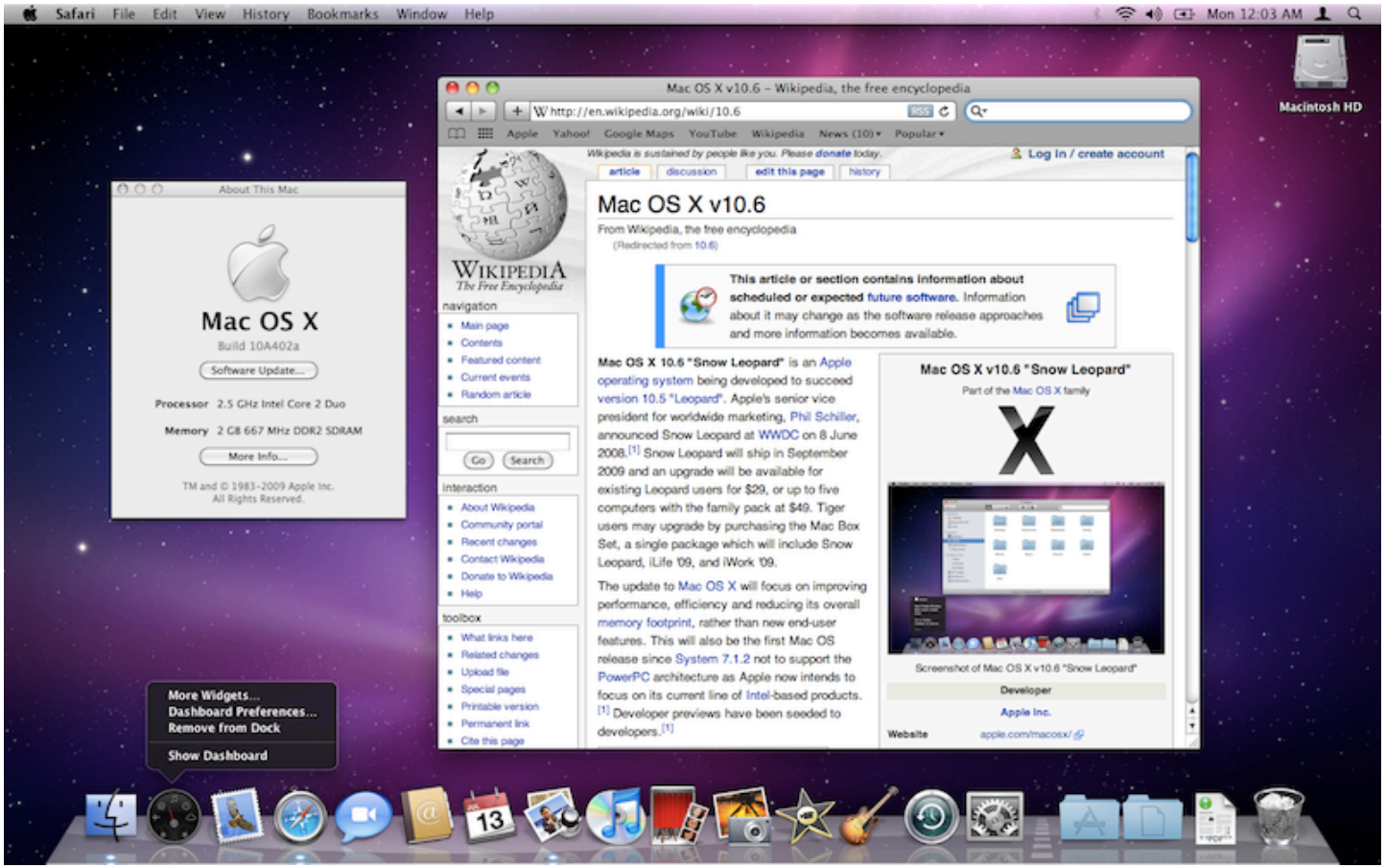
Popular operating systems

- Operating systems include:
  - Microsoft Windows
  - Mac OS X
  - Linux
  - iOS (for Apple mobile devices *e.g.* iPad, iPhone)
  - Android (for Google smartphones and tablets)
- They all look different (present a different *user interface*)
- Internally, they also all run differently

Microsoft Windows



Mac OS X


Safari File Edit View History Bookmarks Window Help
Mon 12:03 AM
Macintosh HD
About This Mac
Mac OS X
Build 10A402a
Software Update...
Processor 2.5 GHz Intel Core 2 Duo
Memory 2 GB 667 MHz DDR2 SDRAM
More Info...
TM and © 1983-2009 Apple Inc.
All Rights Reserved.
Mac OS X v10.6 – Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/10.6
Apple Yahoo! Google Maps YouTube Wikipedia News (10) Popular
Wikipedia is sustained by people like you. Please donate today.
Log in / create account
article discussion edit this page history
WIKIPEDIA
The Free Encyclopedia
navigation
Main page
Contents
Featured content
Current events
Random article
search
Go Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Cite this page
Mac OS X v10.6
From Wikipedia, the free encyclopedia
(Redirected from 10.6)
This article or section contains information about scheduled or expected future software. Information about it may change as the software release approaches and more information becomes available.
Mac OS X 10.6 "Snow Leopard" is an Apple operating system being developed to succeed version 10.5 "Leopard". Apple's senior vice president for worldwide marketing, Phil Schiller, announced Snow Leopard at WWDC on 8 June 2008.[1] Snow Leopard will ship in September 2009 and an upgrade will be available for existing Leopard users for $29, or up to five computers with the family pack at $49. Tiger users may upgrade by purchasing the Mac Box Set, a single package which will include Snow Leopard, iLife '09, and iWork '09.
The update to Mac OS X will focus on improving performance, efficiency and reducing its overall memory footprint, rather than new end-user features. This will also be the first Mac OS release since System 7.1.2 not to support the PowerPC architecture as Apple now intends to focus on its current line of Intel-based products.[1] Developer previews have been seeded to developers.[1]
Mac OS X v10.6 "Snow Leopard"
Part of the Mac OS X family
X
Screenshot of Mac OS X v10.6 "Snow Leopard"
Developer
Apple Inc.
Website
apple.com/macosx/
More Widgets...
Dashboard Preferences...
Remove from Dock
Show Dashboard

Linux


Recent Apps
See 20 more results
GNU Image Manipulation Program
CompizConfig Settings Manager
Illumination Software Creator
GDebi Package Installer
Files
Google Chrome
Recent Files
See one more result
PNG
Ubuntu HUD Menus 12.04 Precise.png
work
Pictures
acpid
bootlogd
speech-dispatcher

iOS

Android


htc
8:43 PM
talk
Google Talk
Browser
Maps
Gallery
Mail
New & Weather
ANDROID
Calendar
YouTube
menu
SABER HD


T-Mobile
2:47 PM
myFaves
Dialer
Contacts
Browser
Maps
MENU

Operating system tasks

- Despite the visual differences, the tasks that the different operating systems have to do is quite similar
- They have to
  - manage the hardware (memory, CPU, disk, monitor)
  - allow users to run multiple programs at once
  - create a *filesystem* and allow users to store and retrieve their files there
  - and many other tasks!

Programming languages

- In order to get the computer to do some interesting task that it doesn't already know how to do, we have to write a *computer program*
- In order to do that, we have to first choose a particular *programming language* to write the program in
- Which programming language should we choose?

Programming languages

- There are literally *hundreds* of computer languages you could write programs in!
- There are at least two dozen languages that are reasonably popular
- Choosing the right language to write your program in is not a trivial task!
- No language does everything best!

Programming languages

- Some programming languages:
  - C, C++, Objective-C
  - Java, C#
  - Python, Ruby, Perl
  - Visual Basic
  - PHP
  - Javascript
  - Scheme, Lisp, Ocaml, Haskell, Erlang, Scala
  - Matlab, Mathematica, Fortran
  - Assembly language, machine languages

Programming languages

- Programming languages often have weird or funny names
  - Python is named after the TV show "Monty Python's Flying Circus"
- Many programming languages are quite similar to each other
  - *e.g.* Java and C#, Python and Ruby, Scheme and Lisp
- But there are major differences between languages too

Python

- In this course, we will use the Python programming language for our problem sets and our projects
- We will talk about other languages too, but not in nearly as much detail
- If you become a programmer, you are going to have to learn more than one language!
  - probably at least 5 or 6 (no kidding!)

Python

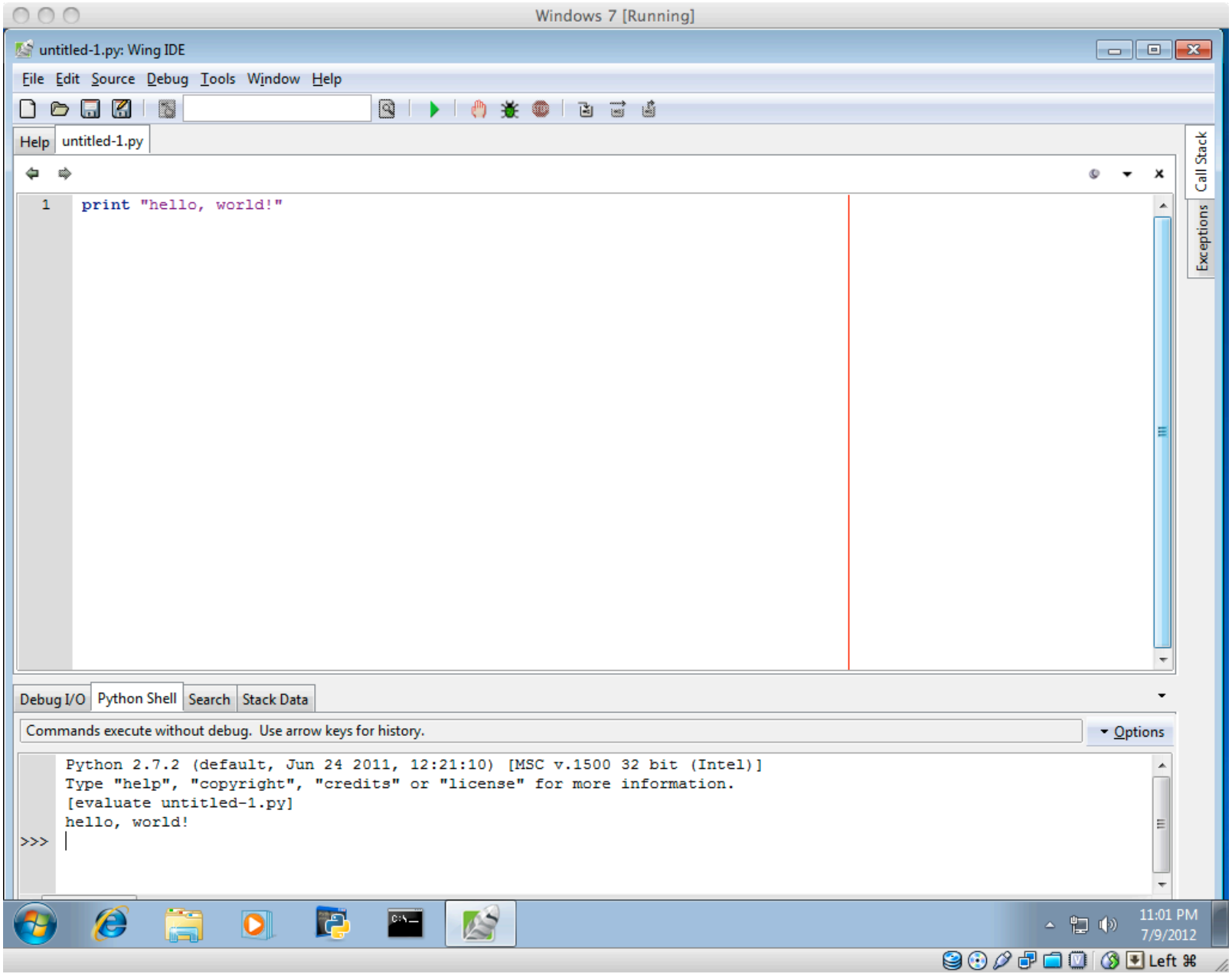
- We will use Python because
  - It's by far the best language for beginning programmers!
  - It's easy to learn
  - It's very powerful
  - It's a "real" language (not just a teaching language)
  - Lots of freely available *code libraries* for different problem domains (graphics, games, internet etc.)
  - It's fun to program in!

Writing programs

- Before talking about Python, we need to discuss how programs are actually written
- The program you write is a text file (plain text) called *source code*
- The particular language you're using (Python in this case) knows how to interpret text files written in the Python language
- There are different ways to write the text file in the first place

Writing programs

- The most basic way to write a program is to use a *text editor*, which is a program which allows us to write text files (of any kind!)
- There are dozens of text editors you could use
- Which one to use is purely a matter of personal preference
- For this course, we'll be using a text editor called *WingIDE*
- WingIDE is free and very powerful!


Windows 7 [Running]
untitled-1.py: Wing IDE
File Edit Source Debug Tools Window Help
Help
untitled-1.py
1 print "hello, world!"
Call Stack
Exceptions
Debug I/O
Python Shell
Search
Stack Data
Commands execute without debug. Use arrow keys for history.
Options
Python 2.7.2 (default, Jun 24 2011, 12:21:10) [MSC v.1500 32 bit (Intel)]
Type "help", "copyright", "credits" or "license" for more information.
[evaluate untitled-1.py]
hello, world!
>>>
11:01 PM
7/9/2012
Left ⌘

Writing programs

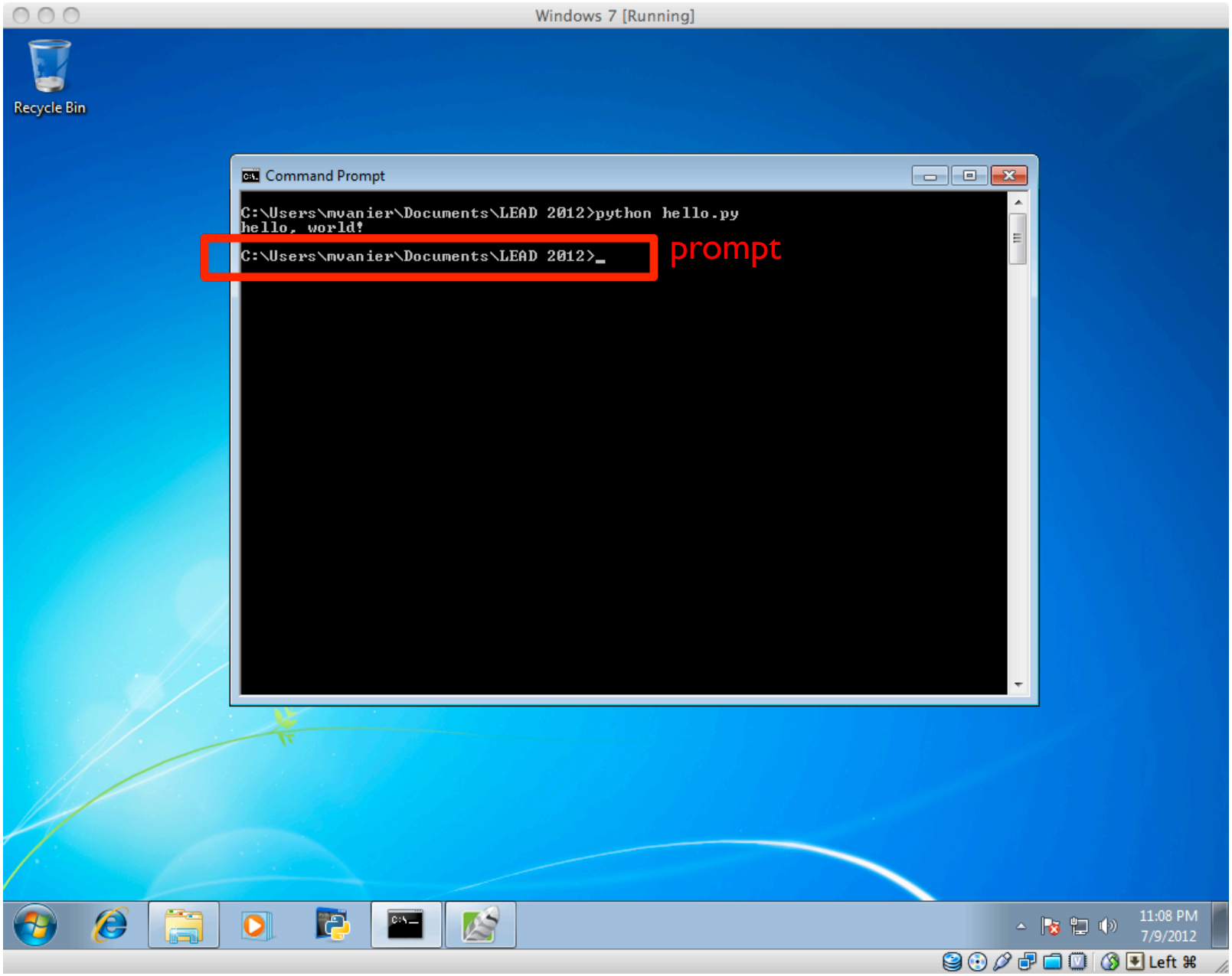
- You'll learn how to use WingIDE in the lab
- Note one cool feature: WingIDE understands Python syntax, so it can color the text in different ways depending on the meaning
- This is called *syntax coloring* and it makes it much easier to write programs
- Never use an editor that doesn't do syntax coloring!

Writing programs

- Once you've written a program, you have to save it to a *file*, which is data that exists permanently on the hard drive of the computer
- Python source code files have names which end in .py (hello.py in the previous example)
- Now you have the file on your hard drive
- How do you run the program?

The terminal

- The most basic way to run a program is to bring up a *terminal*, which is a special program that lets you give commands to the computer
- On Windows computers, the terminal program is called Command Prompt
- It's just an empty window with a place where you can enter commands
- It reads the commands, executes them, and prints the results in the window


Windows 7 [Running]
Recycle Bin
Command Prompt
C:\Users\mvanier\Documents\LEAD 2012>python hello.py
hello, world!
C:\Users\mvanier\Documents\LEAD 2012>_
prompt
11:08 PM
7/9/2012
Left ⌘

The terminal

- If you have a Python program called hello.py, you can run it by typing this at the terminal prompt (not including the >):

```
> python hello.py
```

- To which the program may print something like:

```
hello, world!
```

- and then return you to the prompt:

```
>
```

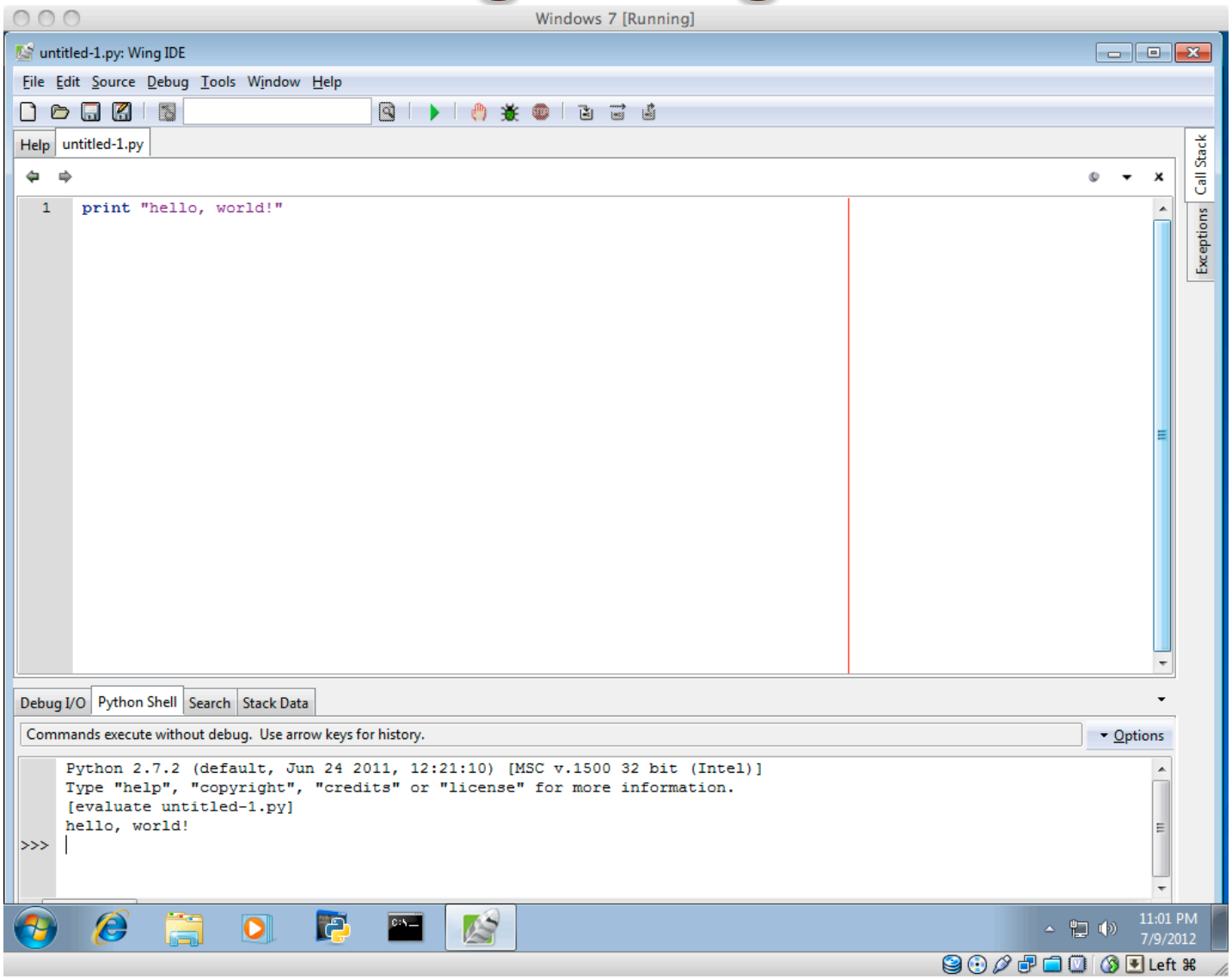
The terminal

- In addition to running Python programs, there are a bunch of other commands you can execute from the terminal:
  - copying files
  - moving files
  - removing files
  - changing the directory
- We will cover these commands in the lab

IDEs

- Writing programs using a terminal and a text editor is the most basic way to do it
- A different way is to use a program called an *Integrated Development Environment* or IDE
- These programs consist of a text editor, various program development tools (*e.g.* a debugger) and a way to run programs without using a terminal
- WingIDE is actually an IDE, not just a text editor

WingIDE again



WingIDE

- The editor in WingIDE is in the upper part of the window
- In the lower part is the *Python shell* where you can run Python programs and experiment with them interactively
- There are other tools in WingIDE as well
- Most of the time, you'll only have to use WingIDE and won't need the terminal
  - (We'll tell you when you need to use the terminal)

Next lecture

- Next lecture, we'll start learning Python!