UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC201 Computer Science I for Majors

Lecture 01 – Introduction

Introductions

- Dr. Katherine Gibson
  - Education
    - BS in Computer Science, UMBC
    - MS & PhD in CS, University of Pennsylvania
  - Likes
    - Video games
    - Dogs
    - Nail polish

Introductions

- Prof. Michael Neary
  - Education
    - BS in Computer Science, UMBC
    - MS in Computer Science, UMBC (in progress)
    - PhD in Computer Science, somewhere (eventually)
  - Likes
    - Chocolate
    - Broadway
    - Improv

Introductions

- Dr. Penny Rheingans
  - Education
    - AB in Computer Science, Harvard
    - PhD in Computer Science, UNC
  - Likes
    - Cheese
    - College sports
    - Data visualization

Introductions

- Dr. Krystle Wilson
  - Education
    - MS, PhD in Computer Science
      Mississippi State University
  - Likes
    - Teen Titans Go!
    - Sports

Course Overview

Course Information

- First course in the CMSC intro sequence
  - Followed by CMSC 202
- CMCS majors must get a B or better
- CMPE majors must get a B or better
  - Unless you entered UMBC prior to Fall 2016
- No prior programming experience needed
  - Some may have it

What the Course is About

- Introduction to Computer Science
  – Problem solving and computer programming
- We’re going to come up with algorithmic solutions to problems
  – What is an algorithm?
- We will communicate our algorithms to computers using the Python language

Class Objectives

- By the end of this class, you will be able to:
  - Use an algorithmic approach to solve computational problems
  - Break down complex problems into simpler ones
  - Write and debug programs in the Python programming language
  - Be comfortable with the UNIX environment

Why Learn to Program?

- Programming skills are useful across a wide range of fields and applications
  - Many scientific professions utilize programming
  - Programming skills allow you to understand and exploit "big data"
  - Logical thinking learned from programming transfers to many other domains

Grading Scheme

- This class has:
  - 8 Homeworks (40 points each)
    - Small programming assignments
  - 2 Projects (80 points each)
    - Larger programming assignments
  - 10 lab assignments (10 points each)
  - 4 mandatory surveys (5 points each)
  - A midterm (200 points)
  - A comprehensive final exam (200 points)

A Note on Labs

- Your “discussion” section is actually a lab
  - In the Engineer building (ENG)
- Labs are worth 10% of your grade
- You must attend your **assigned** section
  - No credit for attending other sections

Submission and Late Policy

- Homeworks and projects will be submitted over the GL server with the **`submit`** command

- Homeworks will always be due at <u>8:59:59 pm</u>
- Late homeworks will receive a ***<u>zero</u>***
- (In other words, there are no late homeworks)

Submission and Late Policy

- It is <u>not</u> recommended that you submit close to the deadline
  - Sometimes the server gets overloaded with everyone trying to submit
  - Developing programs can be tricky and unpredictable
- Start early and submit early (and often!)

Academic Integrity

Academic Integrity

- We have homeworks and projects in this class
- You should never, *ever*, ***ever*** submit work done by someone else as your own
- If you submit someone else's code, both students will get a 0 on the assignment

Things to Avoid

- Downloading or obtaining anyone else’s work
- Copying and pasting another person’s code
- Leaving your computer logged in where another student can access it
- Giving your code to another student
  - Or explaining it in explicit detail to another student
- Attempting to buy code online
  - This will result in an immediate F in the class

Things that are Always Okay

- And encouraged!

- Talking to a classmate about a concept
- Getting help from a TA or instructor
- Comparing program output
- Discussing how to test your program
- Working on <u>practice</u> problems together

Collaboration Policy

- We want you to learn all these things:
  - The course material
  - How to work independently
  - How to work collaboratively
- Some assignments will be “individual work” while others will be “collaboration allowed”
  - These will be clearly marked on each assignment
  - You may only collaborate with current 201 students

What Is Allowed?

Action	Allowed for Individual Work	Allowed when Collaborating
Getting help from an instructor or TA	Allowed	Allowed
Brainstorming general solutions to the assignment		
Creating, sharing, or copying course notes		
Purchasing solutions		
Borrowing verbatim from the course slides or book		
Giving (or receiving) a detailed explanation of a solution		
Looking for solutions or help online		
Looking at someone else's code		

What Is Allowed?

Action	Allowed for Individual Work	Allowed when Collaborating
Getting help from an instructor or TA	**Allowed**	**Allowed**
Brainstorming general solutions to the assignment	Not Allowed	**Allowed**
Creating, sharing, or copying course notes	**Allowed**	**Allowed**
Purchasing solutions	Not Allowed	Not Allowed
Borrowing verbatim from the course slides or book	**Allowed**	**Allowed**
Giving (or receiving) a detailed explanation of a solution	Not Allowed	Not Allowed
Looking for solutions or help online	Not Allowed	Not Allowed
Looking at someone else's code	**It Depends**	**It Depends**

What Is Allowed?

Action	Allowed for Individual Work	Allowed when Collaborating
Getting help from an instructor or TA	Allowed	Allowed
Looking at someone else’s code	**It Depends**	**It Depends**

You may <u>never</u> look at someone else’s code without their permission
You may <u>never</u> look at someone else’s code on your computer

When collaborating, you may look at someone else’s code on their screen and with their permission
When working individually, you may not look at anyone else’s code

Acknowledging Collaboration

- In every file you turn in for this course, you must have a line near the top of your file stating one of the following three things:

1. **`Collaboration was not allowed on this assignment`**
   - On assignments where collaboration was not allowed, you must acknowledge this.

Acknowledging Collaboration

2. `I did not collaborate with anyone on this assignment part`
   - If you did not work with anyone on the part of the assignment the header comment is located in, you must clearly state this.
   - Getting help from a TA or instructor does not count as collaboration.

Acknowledging Collaboration

```
3.  I collaborated with Fox Mulder (fmulder1@umbc.edu);
    I helped him understand the loop.
    I collaborated with Dana Scully (scully18@umbc.edu);
    we helped each other with debugging.
```

- If you worked with anyone on the part of the assignment the header comment is located in, you must state their name and UMBC email, and give a brief description of what the collaboration was.
- Both students need to note this collaboration in their header comment.

Why So Much About Cheating?

- Every semester, around 20 students get caught sharing code. Typically, they are stressed, confused, and just wanted to take a shortcut or help a friend. These students endanger their entire academic career when they get caught.
- If you feel like you can't possibly finish a project or homework on your own, contact someone in the course staff for help.

Becoming a Good Programmer

- We are strict about academic integrity because we want everyone to succeed in this class

- Understanding the assignment solutions means you will do better on the exams
- Learning the course material means you will do better in your future courses and career
- Seeking help when you need it will help you grow as a student and as a computer scientist

Getting Help

Where to Go for Help

- There are a number of places you can go if you are struggling!
  - All of the TAs happy to help
  - If the TAs aren't working out, come by the instructors’ office hours (this should not be your first resort for help)
- All office hours will be posted on the website

CMSC 201 TAs

- You are welcome to go to ITE 240 whenever **any** TA is available to get additional help
- We highly encourage going to them if you have any questions regarding assignments

- The final schedule will be posted later, but there should be a TA in ITE 240 from 10 to 5 Monday-Thursday and a few hours on Friday

ITE 240

- This is a computer lab in the ITE building used to hold 201, 202, and 341 office hours
- The 201 TAs will…
  - Be wearing bright yellow lanyards
  - Have their names on the whiteboard in the front

Additional Help

- Tutoring from the Learning Resources Center
  - By appointment
- Computer help from DoIT
  - By phone or in person
- See the syllabus for more info

Announcement: Note Taker Needed

A peer note taker has been requested for this class. A peer note taker is a volunteer student who provides a copy of his or her notes for each class session to another member of the class who has been deemed eligible for this service based on a disability. Peer note takers will be paid a stipend for their service.

Peer note taking is not a part time job but rather a volunteer service for which enrolled students can earn a stipend for sharing the notes they are already taking for themselves.

If you are interested in serving in this important role, please fill out a note taker application on the Student Disability Services website or in person in the SDS office in Math/Psychology 212.

UMBC Computing Environment

- We develop our programs on UMBC's GL system
  - GL is running the Linux Operating System
    - GUI – Graphical User Interface
    - CLI – Command-Line Interface

- Lab 1 will walk you through using the UMBC computing environment

How Do I Connect to GL?

- Windows
  - Download Putty (Lab 1 has a video about this)
  - Hostname:
    `gl.umbc.edu`
  - Make sure you pick “SSH”
  - Put in username and password

- Mac
  - SSH client is already installed
  - Go to the Application folder and select Utilities
  - Open up a terminal window
  - Enter the following:
    `ssh -l` <u>`username`</u> `gl.umbc.edu`
  - Put in your password

You won’t see any asterisks appear when you type in your password, but it is working!

Linux Commands

- See: http://www.csee.umbc.edu/resources/computer-science-help-center/#Resources

- Here's a few basic commands:

  `ls` – list contents

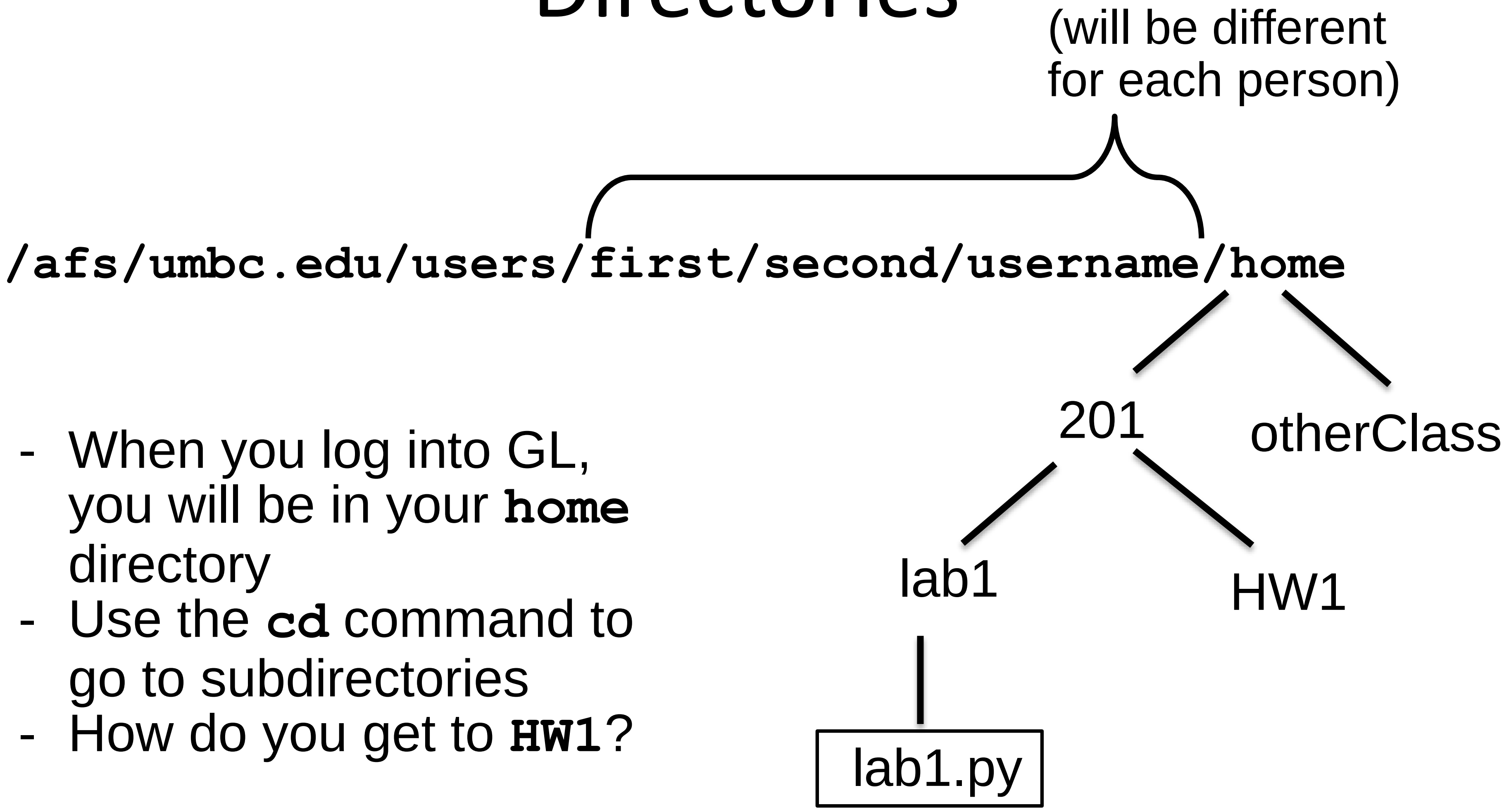
  - List files and directories in your current directory
  - Directory is just another word for folder

More Basic Commands

- **Important!!** Commands are case sensitive

`cd NAME`	– change directory
`cd ..`	– go to parent directory
`cd .`	– stay in current directory
`mkdir NAME`	– make a new directory

Directories


(will be different for each person)
/afs/umbc.edu/users/first/second/username/home
201
otherClass
lab1
HW1
lab1.py

- When you log into GL, you will be in your **`home`** directory
- Use the **`cd`** command to go to subdirectories
- How do you get to **`HW1`**?

emacs – A Text Editor

- Will use emacs to write our python code
- emacs is CLI, not GUI
  - Need to use keyboard shortcuts to do things
- Reference:
  - http://www.csee.umbc.edu/summary-of-basic-emacs-commands/

Keyboard Shortcuts for emacs

- To open a file (new or old)

  `emacs filename_goes_here.txt`
- To save a file

  `CTRL+X` then `CTRL+S`
- To save and close a file

  `CTRL+X` then `CTRL+C`
- To undo

  `CTRL+_` (that “CTRL + Shift + -” for underscore)

Computers and Programs

Today’s Objectives

- To understand how data is represented and stored in memory
- To be aware of elements of the UMBC computing environment
- To start thinking algorithmically

Binary Numbers

- Computers store all information (code, text, images, sound,) as a binary representation
  - “Binary” means only two parts: 0 and 1
- Specific formats for each file help the computer know what type of item/object it is
- But why use binary?

Decimal vs Binary

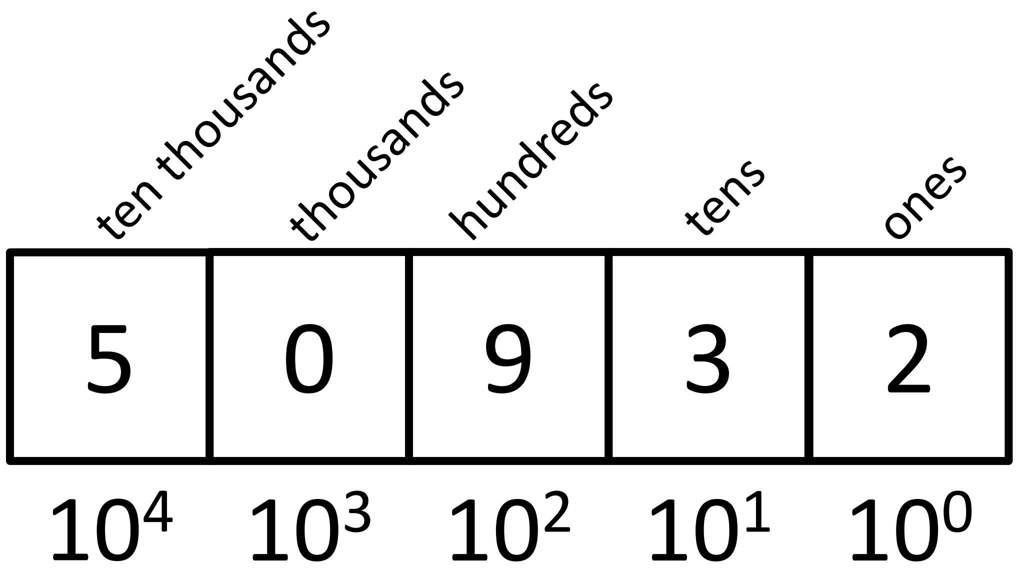
- Why do we use decimal numbers?
  - Ones, tens, hundreds, thousands, etc.

- But computers don’t have fingers…
  - What do they have instead?

- They only have two states: “on” and “off”

Decimal Example

- How do we represent a number like 50,932?



$$
\begin{array}{rcrcr}
2 & \times & 10^0 & = & 2 \\
3 & \times & 10^1 & = & 30 \\
9 & \times & 10^2 & = & 900 \\
0 & \times & 10^3 & = & 0000 \\
5 & \times & 10^4 & = & 50000 \\
 & & & & \text{------} \\
 & & \text{Total:} & & 50932
\end{array}
$$

Decimal uses 10 digits, so…

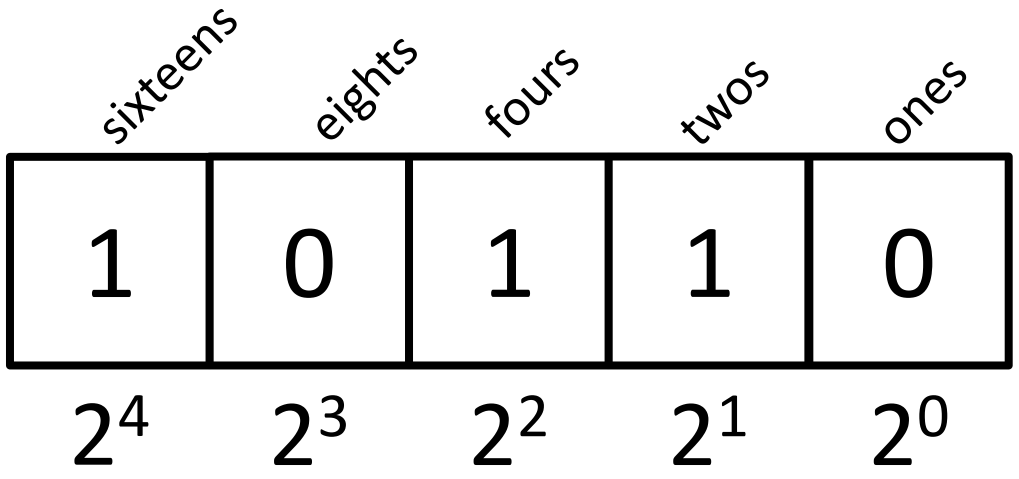
Another Decimal Example

6	7	4	9	3
10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1
60000	7000	400	90	3

60000+7000+400+90+3 = 67493

Binary Example

- Let's do the same with 10110 in binary



```
0 x 2^0 =  0
1 x 2^1 =  2
1 x 2^2 =  4
0 x 2^3 =  0
1 x 2^4 = 16
          --
 Total:   22
```

Binary uses 2 digits, so our base isn't 10, but…

Binary to Decimal Conversion

- Step 1: Draw Conversion Box
- Step 2: Enter Binary Number
- Step 3: Multiply
- Step 4: Add

1	0	1	0	0	0	1	1	0	1
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
512	0	128	0	0	0	8	4	0	1

512 + 0 + 128 + 0 + 0 + 0 + 8 + 4 + 0 + 1 = 653

Decimal to Binary Conversion

- Step 1: Draw Conversion Box
- Step 2: Compare decimal to highest binary value
- Step 3: If binary value is smaller, put a 1 there and subtract the value from the decimal number
- Step 4: Repeat until 0

Convert 643 to binary

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	0	1	0	0	0	0	0	1	1

643-512 = 131 131-128 = 3 3-2=1 1-1=0

Exercise: Converting From Binary

- What are the decimals equivalents of…

```
101
1111
100000
101010
1000 0000
```

(Longer binary numbers are often broken into blocks of four digits for readability.)

Exercise: Converting From Binary

- What are the decimals equivalents of…

```
101       = 4+0+1         = 5
1111      = 8+4+2+1       = 15
100000    = 32+0+0+0+0+0  = 32
101010    = 32+0+8+0+2+0  = 42
1000 0000 = 128+. . .+0+0 = 128
```

(Longer binary numbers are often broken into blocks of four digits for readability.)

Converting to Binary

- What are the binary equivalents of…

  `9`

  `27`

  `68`

  `1000`

Converting to Binary

- What are the binary equivalents of…

```
9    = 1001 (or 8+1)
27   = 0001 1011 (or 16+8+2+1)
68   = 0100 0100 (or 64+4)
1000 = 0011 1110 1000
        (or 512+256+128+64+32+8)
```

"Levels" of Languages

- Machine Code (lowest level)
  - Code that the computer can directly execute
  - Binary (0 or 1)
- Low Level Language
  - Interacts with the hardware of the computer
  - Assembly language
- High Level Language
  - Compiled or interpreted into machine code
  - Java, C++, Python

Compilation vs Interpretation

- Compiler
  - A complex computer program that takes another program and translates it into machine language
  - Compilation takes longer, but programs run faster
- Interpreter
  - Simulates a computer that can understand a high level language
  - Allows programming “on the fly”

Algorithmic Thinking

- Algorithms are an ordered set of clear steps that fully describes a process
- Examples from real life?
  - Recipes
  - Driving directions
  - Instruction manual (IKEA)

Exercise: PB&J Algorithm

- English speaking aliens are visiting Earth for the first time. They want to know how to make a peanut butter and jelly sandwich.
- Explicitly, what are the required steps for building a peanut butter and jelly sandwich?

Announcements

- Lab 1 this week is an online lab
- In-person labs won't begin until the week after Labor Day

- Make sure to log into the course Blackboard
  - Let us know if you have any problems
  - (Students on the waitlist may not have access yet)