Preliminaries and Intro. Material

- Course Documents
- Taking Notes
- Cheating
- Accommodations
- E-mail
- Attendance
- Time and Other Considerations
- Programs, Software, and Software Development
Course Documents

• Everything I create for this class is made available online
• All of it can be accessed from the Class Web Page, whose address will be given in class
• You should bookmark this page because the page will function as our syllabus, instead of a paper syllabus
• It is a lot of material, but you should at least get to know the layout
Course Documents

• The "Course Policies" and "Classroom Rules" sections will give you a good idea of my rules and expectations.
• Those sections also contain some supplementary information for you to check out.
• The page will feature links to class notes, along with schedule information and assignments.
• You should also check the site frequently for updates, such as new assignments posted.
Taking Notes

• Although I make my notes available in PDF form, I want to encourage you to take notes in class.
• Studies have shown that students learn more when they take notes, even if they never look at their notes again.
• Other studies have shown that the more activities and senses are engaged when you learn something, the greater your likelihood of remembering.
Taking Notes

• Writing notes engages another part of your brain, which increases recollection
• All of you should take notes
• Probably the best practice would be for you to print the notes before coming to class.
• That way, you can write your own notes in the margins, along with any questions you have.
• Also, if you are receiving specific help for some task, it's good to write things down...
Cheating

- All students are expected to follow the University's Code of Student Conduct
- You will find this at http://www.umb.edu/life_on_campus/policies/community/code
- The Computer Science Department has the following policy on cheating
- You will be given a score of zero if you cheat on any assignment, quiz or test
Cheating

• If you cheat a second time you will receive an F in the course
• If you cheat a third time you can be expelled from the University
• I put a great deal of work into my courses, and I ask you to respect that work by not cheating
• Given the nature of this course, we need to address the topic of collaboration...
Is Collaboration "Cheating"?

• The short answer: It depends.
• Discussing concepts (things not pertaining directly and specifically to the assignment) with others is fine.
• You may also help one another with things like:
  - Using software development tools like DrJava
  - Understanding the assignment itself – the parts already provided to you
  - General programming-related issues, not pertaining to a specific assignment
Is Collaboration "Cheating"?

• You may not engage in any of the following:
  ➢ Viewing or copying one another's code
  ➢ Copying or plagiarizing any solution from any source
  ➢ Coaching someone step-by-step through writing the solution code
  ➢ Failing to acknowledge any allowed collaboration, which must include names and/or sources

• Keep in mind that we (instructors) have various ways of detecting cheating, including detection software!
Accommodations for Disabilities

• The school is legally obligated to try to accommodate students with disabilities

• If you have a disability you can get help from Ross Center for Disability Services
  ➢ Location:
    Upper Level of the Campus Center, Room 211
  ➢ Phone: 617-287-7430
  ➢ Web Site:

  https://www.umb.edu/academics/vpass/disability/
Accommodations for Disabilities

• After you have discussed the matter with them, see me
  ➢ They will usually draft a letter explaining any accommodations you should receive.
  ➢ You should get this letter to me ASAP!
  ➢ If you require extra time for an exam, then it is your responsibility to arrange for this at least a week in advance!

• Also, you may wish to check out the page containing my own notes:
Email

• All communication outside of class will be conducted through e-mail
• For course announcements and personal correspondence, I will be using your @cs.umb.edu email
• However, you need not check that e-mail. Instead, you can have correspondence to that address forwarded to the account of your choice.
• This is why you are advised to set up your .forward file on your Linux account
Email

- It is your responsibility to check your e-mail regularly – at least once every 24 hours.
- If I have sent you an email about something concerning the class, I'll assume that you have been given adequate notice.
- Also, grades on certain assignments may be distributed via your @cs.umb.edu email
Contacting Me

• If you have a question, email me at cg.kelly2013@gmail.com

• Please be sure to:
  - Use a descriptive, meaningful subject line
  - Begin the subject with the class name (e.g., IT114, CS110, etc.)

• Don't hesitate to contact me if you are stuck and/or need help with something.

• Please allow for at least 24 hours for a response, or more, during weekends and holidays.
Office Hours

• My office is S-3-130
• My official office hours are posted on the course web page
• You do not have to make a special appointment to see me during office hours – just drop in!
• If you need my help and cannot make it to office hours, contact me and we'll work something out
Attendance

• At each class I'll take attendance
• I do this to:
  - Learn your names
  - Have a record
• Your attendance will not affect your grade directly
• However, if you find yourself struggling with the material and have not been coming to class, I'll be less sympathetic
Do You Have Enough Time to Do the Work for This Course?

- Many of you work, either part time or full time
- This cuts down on the time you have available for class work
- You should not be taking this course if you do not have enough time to do all the work
- In this course, you will be learning a new way of thinking – like a computer
  - For some this may come easily and naturally
  - For others, it may require some extra effort
Do You Have Enough Time to Do the Work for This Course?

- As such, you may have to invest more time (tutoring, office hours, practice problems, etc.) in order to
  - Learn the skills, and...
  - Complete the work – at a sufficient level of quality to earn your desired grade
- If you sign up for more work than you can achieve in the time you have, you will be cheating yourself
- It requires doing enough work to digest and understand the material
Other Considerations...

• How well do you handle minute details? Can you keep track of things like:
  - Uppercase versus lowercase
  - When to use single quotes ' ' versus double quotes " "
  - When to use parentheses ( ) versus curly braces { } versus square brackets [ ]

• How good are you at reading directions and following them specifically? Such as...
  - Coding conventions
  - File names and locations
  - Folder names and locations
  - Assignment specifications
Other Considerations...

• For example, if asked to name a file *homework_09.txt*, that means *none* of the following are acceptable:
  
  Homework_09.txt
  homework09.txt
  homework_9.txt
  homework_09.rtf
  Homework 9.doc
  ...

• Small details are especially important, considering how computers work.
Homework Assignments

• We assume that you are computer literate:
  - Word Processing, Email, Web Browsing, Downloading Applications, etc.

• Reading for today: Dawson, Chapter 1
  - We may not cover all this material in class, but you are responsible for knowing it on exams, etc.

• If you have a hard time with this material, please see me – sooner rather than later!
How Computers Work

- Computers consist of two main components: **hardware** and **software**

- **Hardware** refers to the physical parts, such as the following:
  - Monitor
  - The "case"
  - Keyboard
  - Mouse
  - Cables
  - Printers

- These parts, in turn, consist of smaller components.

- The case, for example, is home to the processor, memory, data storage, chips, wires, and so forth.
4 Main Hardware Components

1. CPU: "Central Processing Unit" - the "brains" of the computer. Carries out the actual commands of a program.

2. I/O: "Input/Output" - keyboard, mouse, monitor, speakers, and other tools that make user interaction possible

3. Main memory: Also called "Random Access Memory" (RAM). Keeps data nearby for the CPU to use

4. Storage: A device that holds data on a more permanent basis, for use and reuse
Types of Software (Programs)

• Computers are very powerful pieces of hardware that can’t do much useful work until they are properly programmed

• There are three different types of software:
  - Operating Systems
  - Application Programs
  - Software Development Tools (or Kits)

• As a computer programmer, you may need to use and/or write any or all three types of programs
Operating System Programs

• “O/S” programs control the hardware and allow application programs to be executed
• An O/S is usually built to run on a specific underlying hardware platform, e.g. PC, MAC, or server
• Generally these are the most complex types of programs to write and test
• Examples:
  ➢ M/S DOS, Windows, UNIX, Linux, Solaris, etc.
Application Programs

- “Apps” perform useful work for their users
- Apps are usually built to run on a specific operating system (and maybe a specific underlying hardware platform)
- Users typically need to provide a lot of information about their job tasks for a programmer to write a good application program for that purpose
- Examples:
  - Word, Excel, PowerPoint, Chrome, etc.
Software Development Tools

- Software Development Tools or Kits (SDK’s) are specialized application programs that allow programmers to write and test programs.
- Experienced programmers generally prefer an “Integrated Development Environment” (IDE).
- Examples (that we’ll be using in this course):
  - IDLE (packaged with Python)
  - Sublime Text 2 (used in class)
Styles of User Interface

• User Interface: How the user interacts with the underlying program logic

• There are two predominant styles:
  ➢ Command Line Interface (CLI)
  ➢ Graphical User Interface (GUI)

• As a computer programmer, you must be able to use and/or write programs for both styles of user interface
Styles of User Interface

• Command Line Interface (CLI)
  – Computer types a “Prompt” requesting input
  – User types a “Command” with “Parameters”
  – Predominantly an old style of interaction that does not require a lot of computer power, but still in use today in some O/S and applications
  – Considered to be NOT “user friendly”, but is very efficient when combined with “scripting”
  – Example: UNIX/Linux CLI, command & parameter
    $ cat file.txt (display the contents of the file)
Styles of User Interface

• Graphical User Interface (GUI)
  ➢ Computer displays a combination of text and graphical symbols offering options to the user
  ➢ User manipulates mouse and uses keyboard to select from the offered options (“hot keys”) or to enter text
  ➢ More common now (computer power is cheap)
  ➢ Considered by most to be “user friendly”
  ➢ Examples: Windows, Microsoft Office, iTunes
Software Development Tools

• Using development tools separately

Diagram:
- Programmer
- Editor
- Source File(s) (.py)
- Interpreter (python)
- Command Line Interface
- Program executes
Using Tools Separately

- Example UNIX/Linux Commands and Parameters
  
  $ nano HelloWorld.py
  
  (Create/edit “source file” via the command line)
  
  $ python HelloWorld.py
  
  Hello World

  $ exit
Software Development Tools

- Your options include IDLE (usually comes with installation) and Sublime Text 2
Live Demonstration: IDLE
Live Demonstration:
Sublime Text 2

```python
# Game Over
# Demonstrates the print function

print("Game Over")

input("\n\nPress the enter key to exit.")
```
Errors

A program can have three types of errors:

• **Compile-Time:** The dev. software will find syntax errors, type errors, and other basic problems
  (Not applicable to Python because it is an interpreted programming language)

• **Runtime:** A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally
  (For Python, this will include what would be compile-time errors, in another language)

• **Logical:** A program may run, but produce incorrect results, perhaps using an incorrect formula
Program Development Steps

• Classical “Waterfall” Development Steps

- Edit and save source code
- Build source code to create program
- Run program and evaluate results
- Errors
Program Development Steps

• An incremental approach

Requirements, “nice-to-haves”, brainstorms, diagrams, etc.

Analysis and Design

Planning and Delegating

Evaluation and Reflecting

Coding (a little bit) and Compilation

Testing

Finished product