CS – 211
Programming I for Engineers

Instructor: Tom Bartenstein

Course Web Page:
http://www.cs.binghamton.edu/~tbartens/CS211_Fall_2019/

iClicker Attendance (warm-up)

Please click on A if you are here:

A. I am here today.

iClicker Usage

- Attendance and Participation
- Occasional (ungraded) quiz questions to…
  - keep involved
  - Gauge how we are doing
  - Gather information about the community
  - Make the class more fun and interesting
- I will be using “iClicker Cloud” (not “iClicker classic”)
  - Enables more participation

iClicker Access

iClicker Device

- Available from the Bookstore for a nominal price
- Use for all iClicker courses*
- Two flavors – Multiple choice only vs. “Enhanced” enables other kinds of questions
- No on-line feedback available

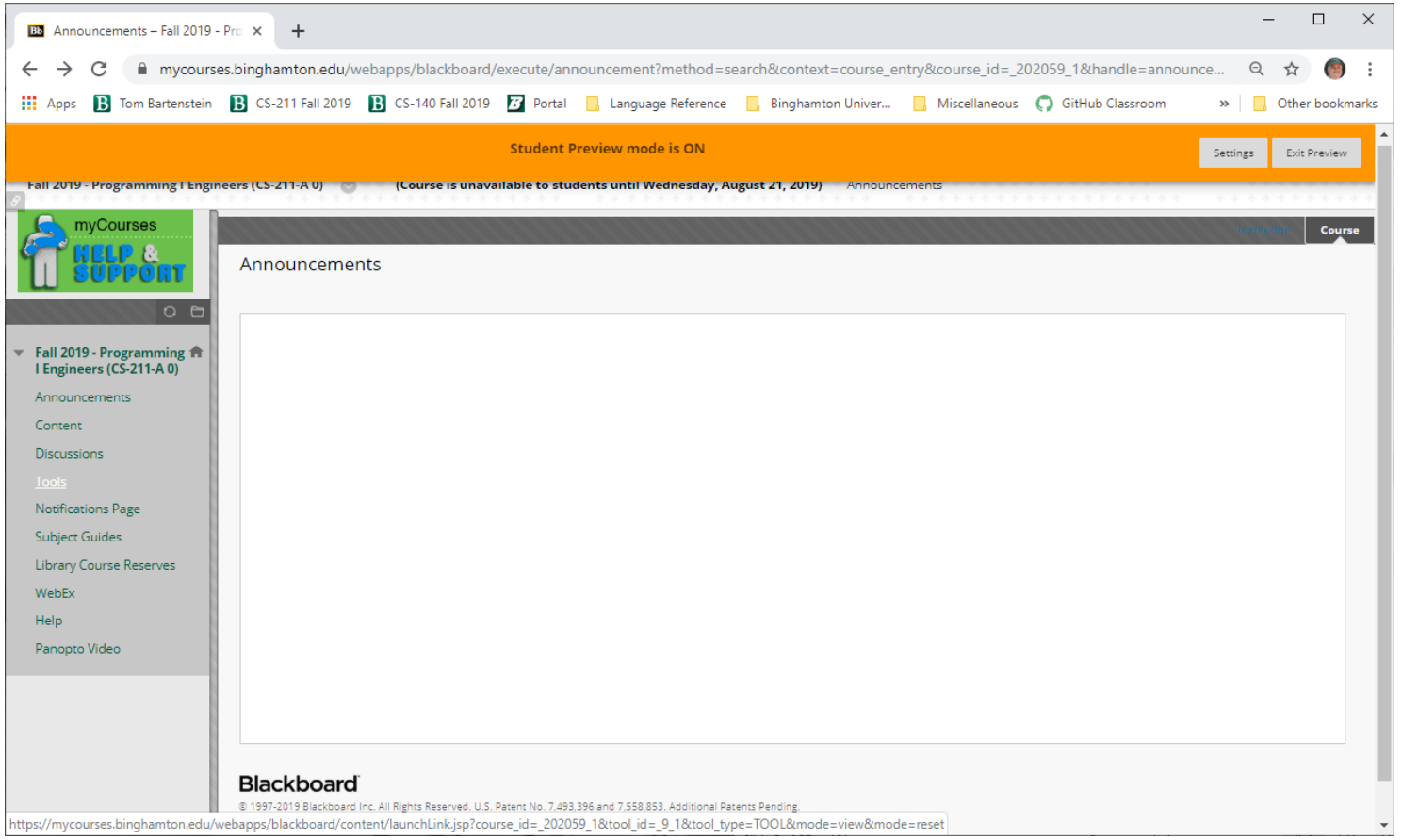
* Unless instructor requires more

iClicker Account

- Nominal cost per semester/year
- Use for all iCLicker courses
- Enables iClickr REEF (Smart phone app)
- Can be run from tablet or laptop
- Enables online feedback

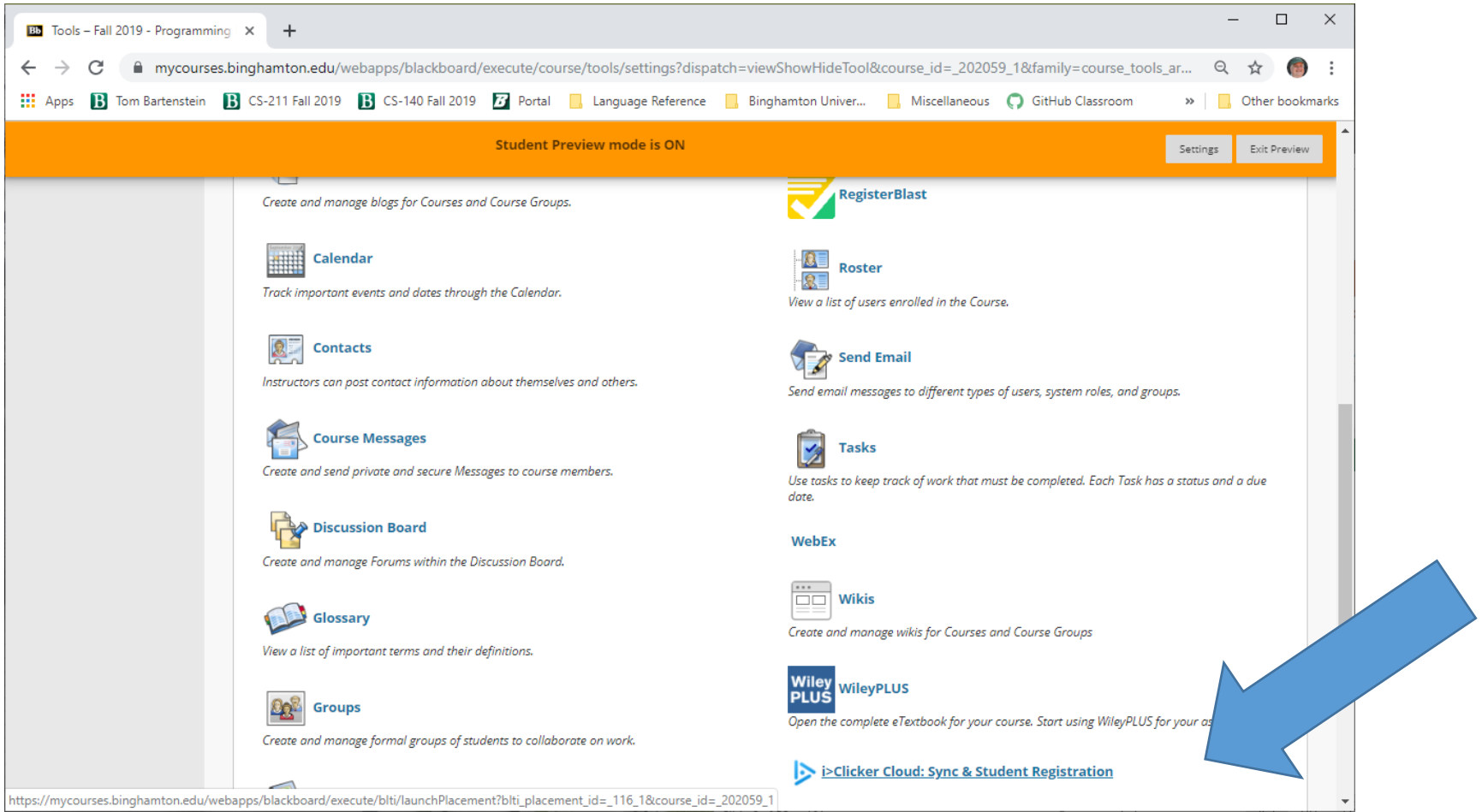
iClicker Registration

- Once you have your device or account, open myCourses CS-211, navigate to “tools”



iClicker Registration

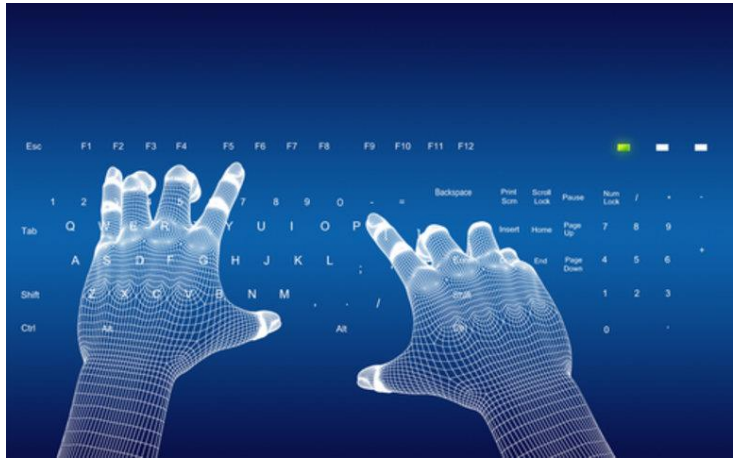
- Inside tools, scroll down to iClicker cloud Sync & Student Registration



Catalog Description

- Introduction to computer programming with engineering applications
- Programming in the procedural language C,
  - control structures,
  - functions,
  - arrays and pointers.
- Intro to abstract data types and object-oriented programming

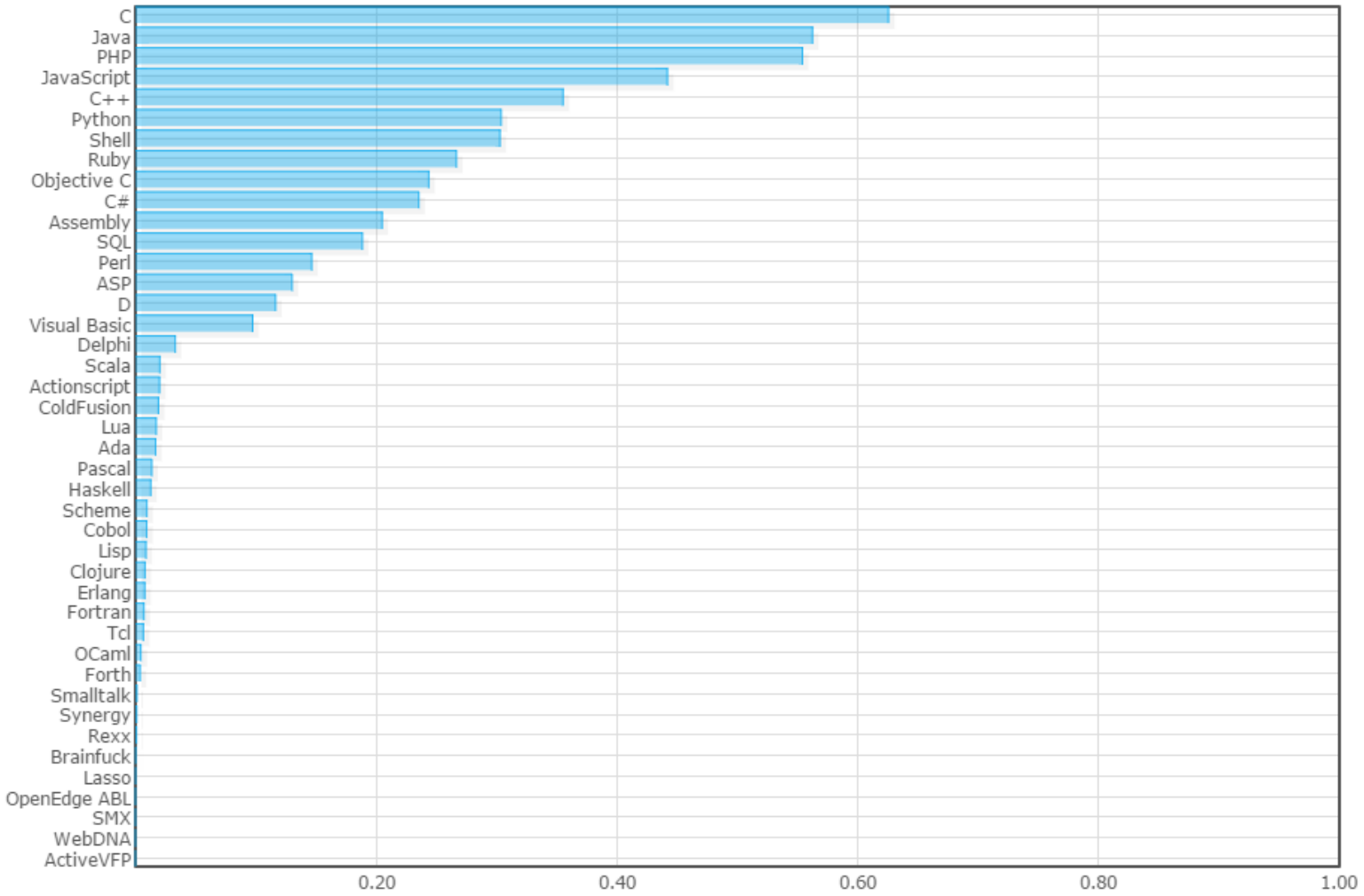
What is Programming?



- Tell a machine how to solve a problem
  - We need to know how to solve the problem : algorithm
  - We need a language that the machine understands: C
- What can a machine do?
  - Arithmetic : $+$, $-$,$\times$,$\div$, mod
  - Logic: AND, OR, NOT, XOR, equals, < , >
  - Data Manipulation: Assignment (copy)
  - Data Structures: Vectors, Matrices, lists, etc.
  - Memory: pointers, addresses, pointer dereference
  - Control: flow through a program, move to a different instruction
  - Communication: Input, Output, to/from peripheral devices

Why C?

- Popularity
- Efficiency
- Simplicity

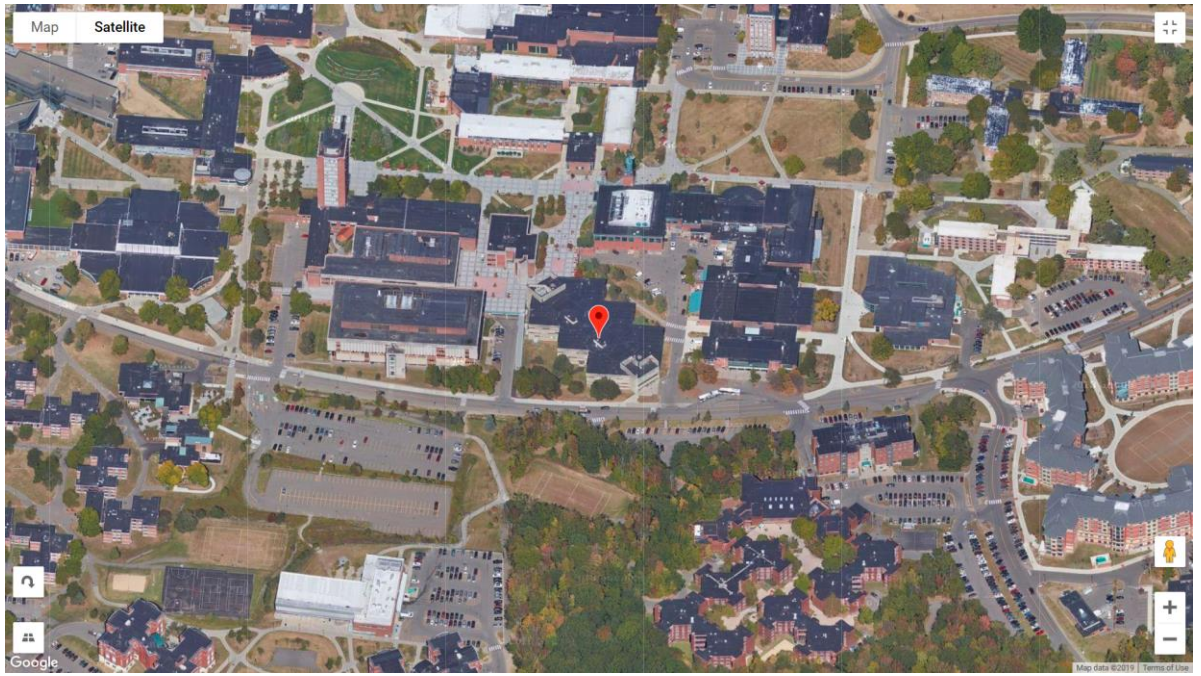


Textbooks

- Kochan,
  - "Programming in C, Fourth Edition", Addison Wesley, 2015
- Brian Kernighan and Dennis Ritchie,
  - "The C Programming Language, Second Edition", Prentice Hall, 1988

Teaching Staff

- Prof: Tom Bartenstein (tbartens@binghamton.edu)
  - Office Hours: MW 3:30 – 5:30 pm @ EB-Q06 or by appointment
- Course Assistants
  - Micheal Chung, Milton Pereza
  - More to come



Professional Background

- 1975-1979 - BA in Mathematics Swarthmore College
- 1979-2001 – Programmer/Architect at IBM Endicott
  - PCB design transfer to Mfg
  - Integrated Circuit Manufacturing Test Generation and Diagnostics
  - Published papers and wrote many patents
- 2001-2011 – Architect at Cadence Design Inc. Endicott
  - Won Cadence “Innovator of the Year” aware in 2010
- 2011-2017 – PhD at Binghamton University in CS
  - Dissertation: Rate Types in Stream Programming
- 2015-???? – Adjunct Lecturer at BU
  - Architecture and Programming Languages

Lectures

- Mon & Wed evening: 5:50-7:15
- Much content will be available via videos
  - Make class more interactive
  - Please watch videos BEFORE class!
- Notes will be published on class web page
- Attendance is required!
  - I-Clickr attendance will be taken each class
  - E-mail prior to class if you cannot make it
- Questions and Discussion are *strongly* encouraged!
- Opportunity to participate!
- Will include demonstrations & group problem solving

Labs

- Attend assigned lab section (one of four: Mon - Wed)
- Attendance required!
- Instructions will be passed out at the start of the lab (hardcopy)
- Held in LNG-103
- Run by Course Assistants (CA's)
- Do your own work!
- Submission of results on myCourses
  - Due Friday at 11:59 PM

Homework

- Watching videos
- Reading relevant text
- Assignments posted on class web page
  - Practice for tests and exams
  - Answers will be posted and reviewed in lecture

Project: Simulating Circuits

- Four installments will be assigned, each graded separately
- Each installment will take significant effort – need to manage time effectively
- Instructions will be posted on the class web page
- Installments get harder as the semester progresses
- Do your own work!
- Lowest project grade will be dropped!

Grading

Quizzes, Attendance, Participation	15%
Labs	15%
Project	30%
Tests	20%
Final Exam	20%

There is no predefined average number to letter mapping!
Letter Grades depend on comparisons with students in previous and current semester, difficulty of tests, etc.

Plagiarism

- Work together!
  - You will learn more from each other than you learn from me
  - Ask Prof, CA's or classmates questions about tools, specifications, algorithms, strategies, debugging, error messages, etc.
- Write your own code!
  - I will check for copied code
  - Changing variable names, white-space, or comments does NOT negate copying!
- Copied code will be considered plagiarism
  - Violation of the Watson Student Academic Honesty Code
  - Both copy-er and copy-ee will receive a zero grade

CS-211 Course Goals

- Goal: Learn how to write a computer program
  - Provide capability to create tools for future use
  - Make it easier to understand how tools work
  - Make it easier to specify tools to tool providers
- Goal: Learn more about computers
  - Learn about how computers are used
  - Learn about how simple hardware can enable complex software
- Goal: Learn about problem solving
  - How do we approach a problem
  - How do we specify the answer so that a machine can understand
- Goal: Learn about Computer Science
  - Techniques to make computers useful: Data structures, algorithms, etc.

How to learn a Language

- You can’t teach a language!
- You can teach syntax , grammar, and vocabulary
- You cannot teach style/voice
- You cannot learn to program without practice!
  - You can’t read without practice
  - You can’t write without practice

Development Environment

- Operating System: Linux (UNIX)
- Editor: gedit (or your favorite editor)
- Compiler: gcc (GNU C Compiler)
- Builder: make (using Makefiles)
- Execution: UNIX Command Line
- Debugger: gdb (GNU Debugger)

Environment 1: Linux Lab (LNG 103)

- Access when library is open
  - unless lab in session
- Use your PODS userid/password to log in
  - New Windows machines installed in LNG103
- Will use "VMware/Horizon" to access a UNIX environment
  - See Using Vmware for details
  - Only available in the LNG-103 lab!
- Note: This is the environment where your assignments will be graded!

Network U Drive

- Your “home” directory in LNG103 Linux is a 5G “U Drive”
- All students should have a U-Drive (provided as part of the course)
  - If you registered late, you may need to request your own U-Drive
- Accessible from other hardware/operating systems
  - Available from Windows (& Apple?) PODS machines
  - See IT U-Drive web page for instructions to connect to your machine
- At the end of the year, contents of your U-Drive are zipped and stored on your Google Drive.

Environment 2: SSH/harveyv

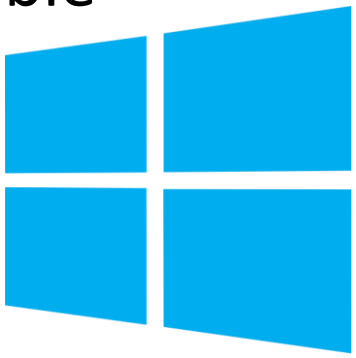
- Open a window on “harveyv.binghamton.edu”
- If you have a U-Drive, the U-Drive will be your “home” directory
- I use MobaXterm from Windows… enables full-screen apps (but slow)
  - I used to use PuTTy and Xlaunch, but it was slower and harder to use
- Use mounted copy of U-Drive on local machine for editing
  - Or SFTP tool to copy files to local machine and edit, and return
  - I use WinSCP from Windows

Environment 2A: SSH/harveyv on PODS

- From a Windows PODs machine (e.g. at the library)
  - See IT PODS Public Computing Labs web page for list of PODs sites
  - I'm not sure if MobaXterm is installed… if not, PuTTy is installed
- Open a window on "harveyv.binghamton.edu"
  - The U-Drive will be your "home" directory
- Use mounted copy of U-Drive on local machine for editing
  - Use notepad++ (installed on PODs machines in library) for editing

Environment 3: Windows/Cygwin

- Free Windows UNIX emulator : https://www.cygwin.com/
- Long initial download/installation, but very nice afterwards
- Configure to include tools: gcc, gdb, and gmake
- Does not support full-screen X without lots of extra work
  - Use a windows editor instead
- C may not be 100% compatible

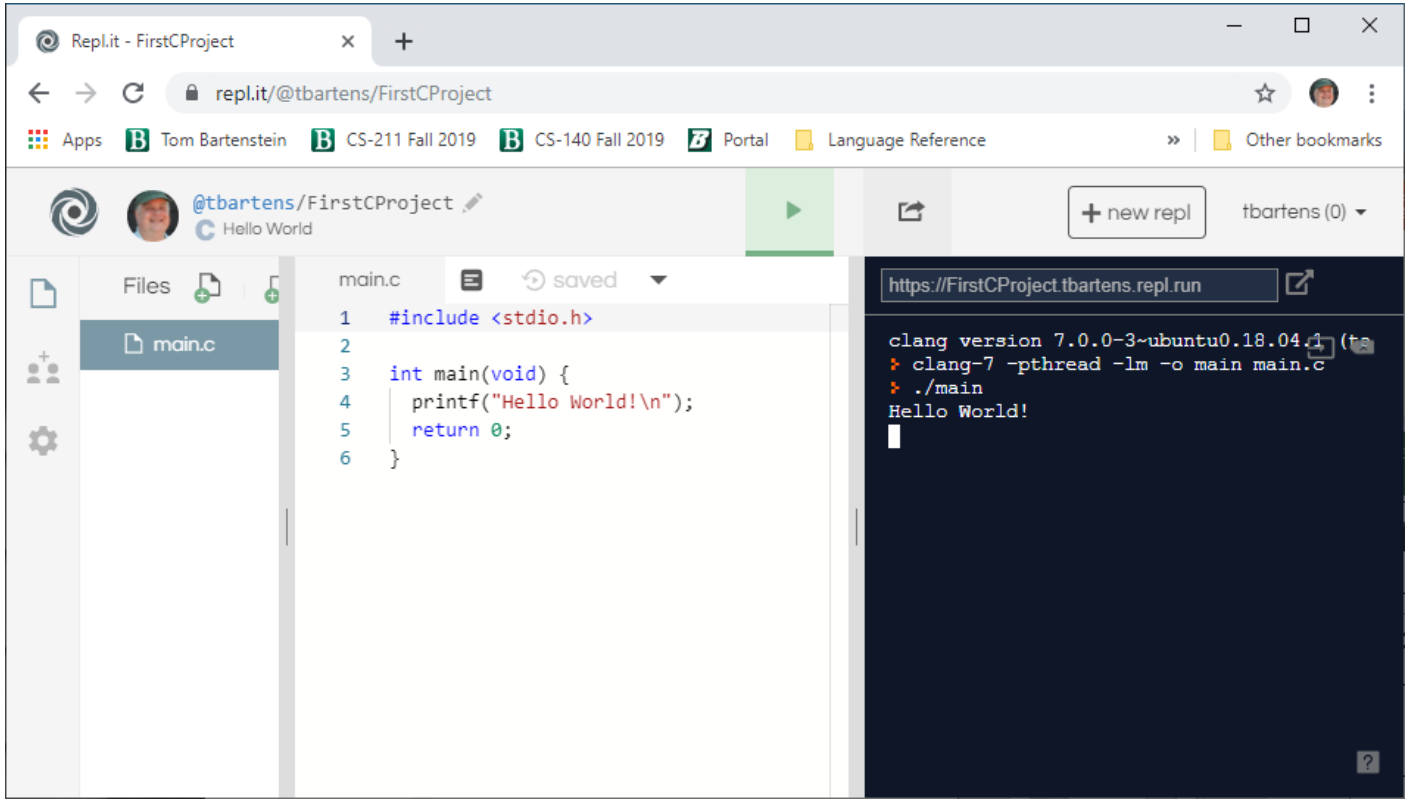




Environment 4: Apple / IOS

- I know nothing
- IBM wouldn’t let me use Apple products

Environment 5: Web Based IDE

- For instance, https://repl.it/
- Beware compatibility issues!



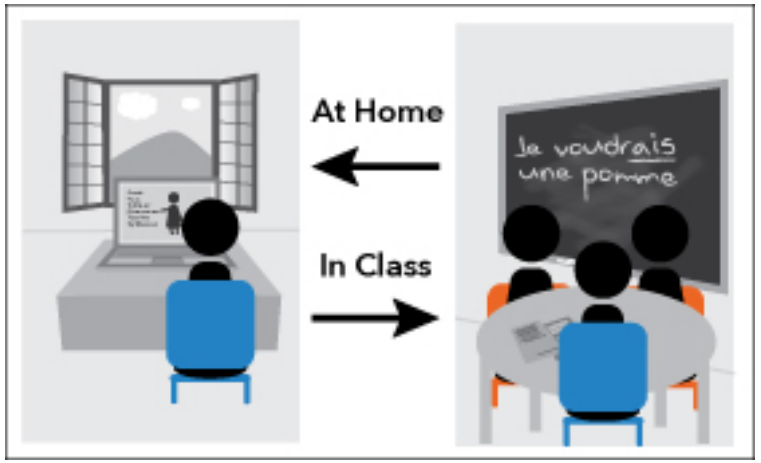
iClicker Question

When working on homework or projects:

A. I have a Windows based computer that I can use
B. I have an Apple based computer that I can use
C. I have a UNIX based computer that I can use
D. None of the above.

"Flipped" Classroom

- The concept of watching lectures on video outside of class and…
- Doing homework in the classroom, with the help of the professor and other classmates



- Flipping the classroom is much harder when doing homework requires hardware
- Hence…

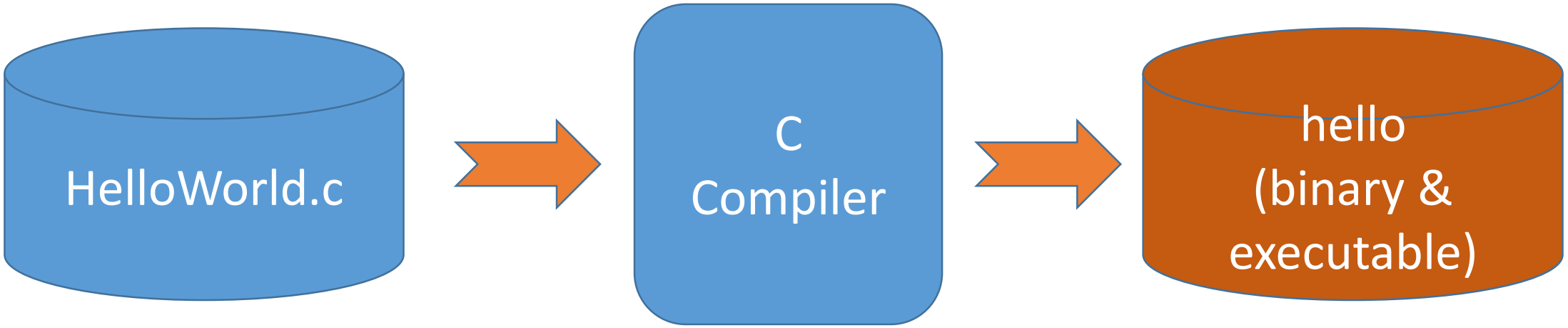
iClicker Question

If I ask you to write programs in class:

A. I have a Windows based laptop or notebook that I can use
B. I have an Apple based laptop or notebook that I can use
C. I have a UNIX based laptop or notebook that I can use
D. None of the above.

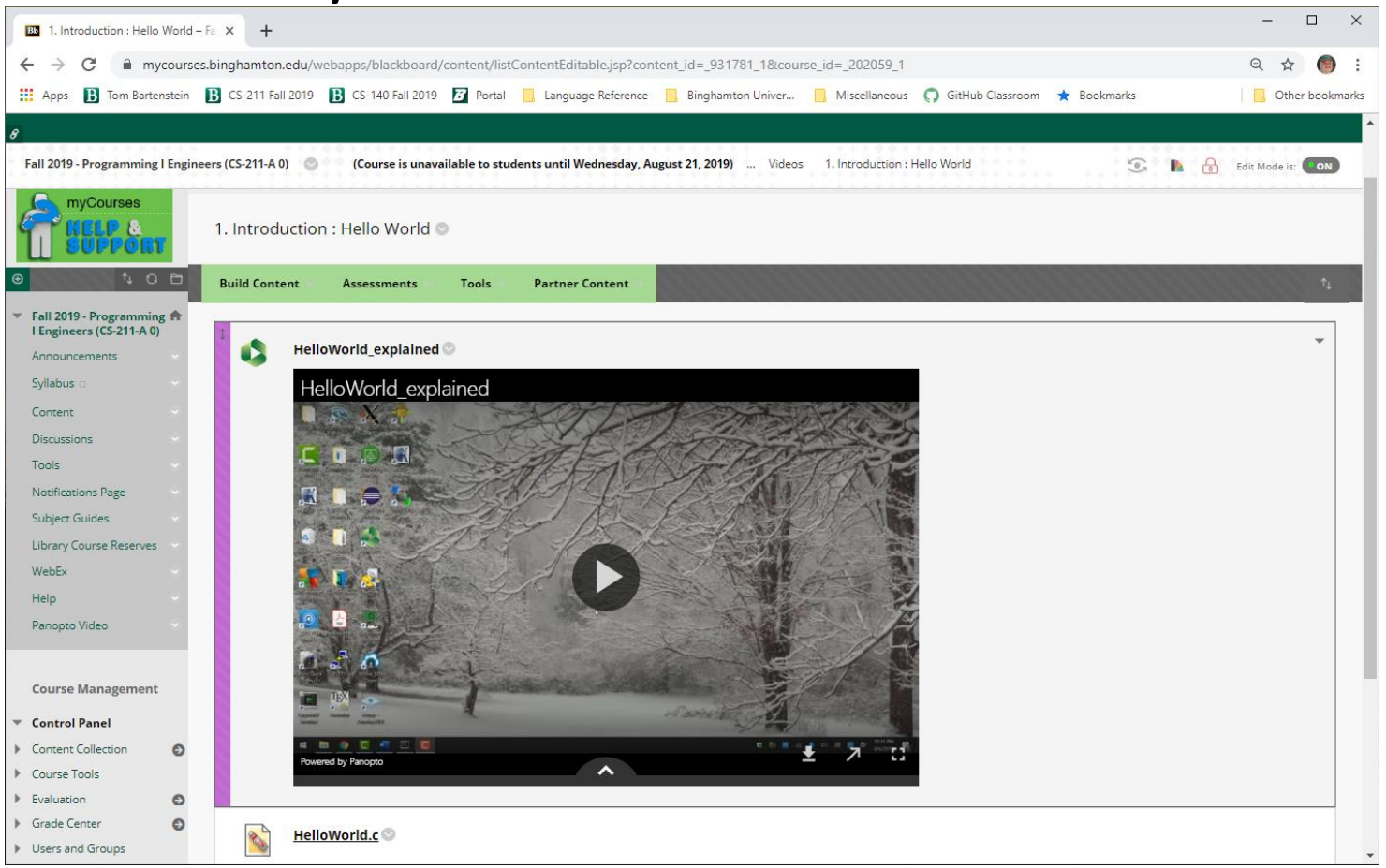
Introductory Terminology

- File: text (readable) or binary (unreadable) data on disk
- Source file: text file that contains your program… <xyz>.c
- Executable file: “Machine code” binary version of your program,
  - Machine language generally not readable by humans
  - treated like a command in Unix
- Compiler: Translates source file into executable file



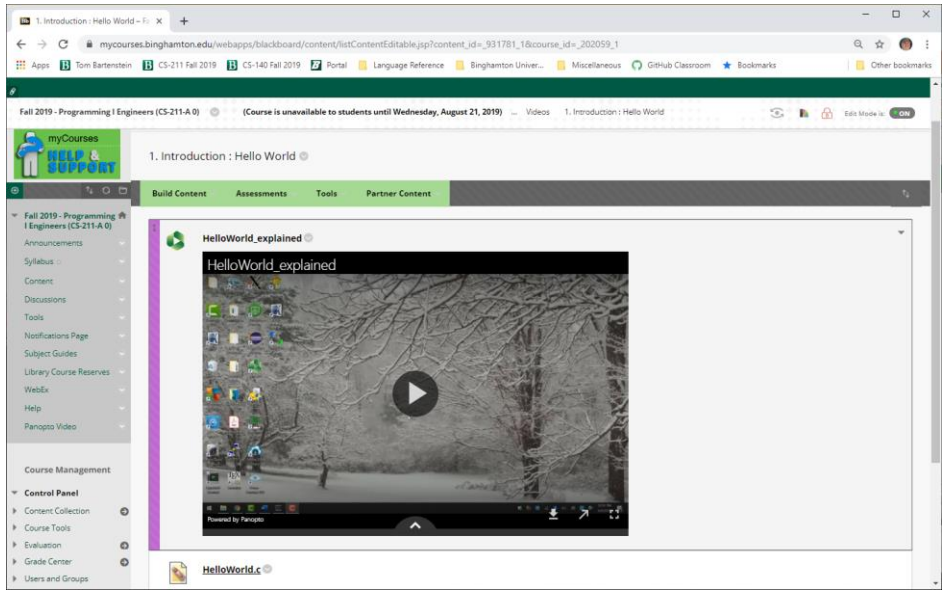
Introduction: Hello World

- Watch the video, available on myCourses
  - Content
    - Videos
      - 1. Introduction



Resources

- Programming in C: Chapter 2
- The C Programming Language: Section 1.1
- Wikipedia "Hello World!" program
- Wikipeda C (programming language)



Introduction: Hello World

Summary Notes

helloWorld.c

```
#include <stdio.h>


int main() {
        printf("Hello World!\n");
        return 0;
}
```

helloWorld.c

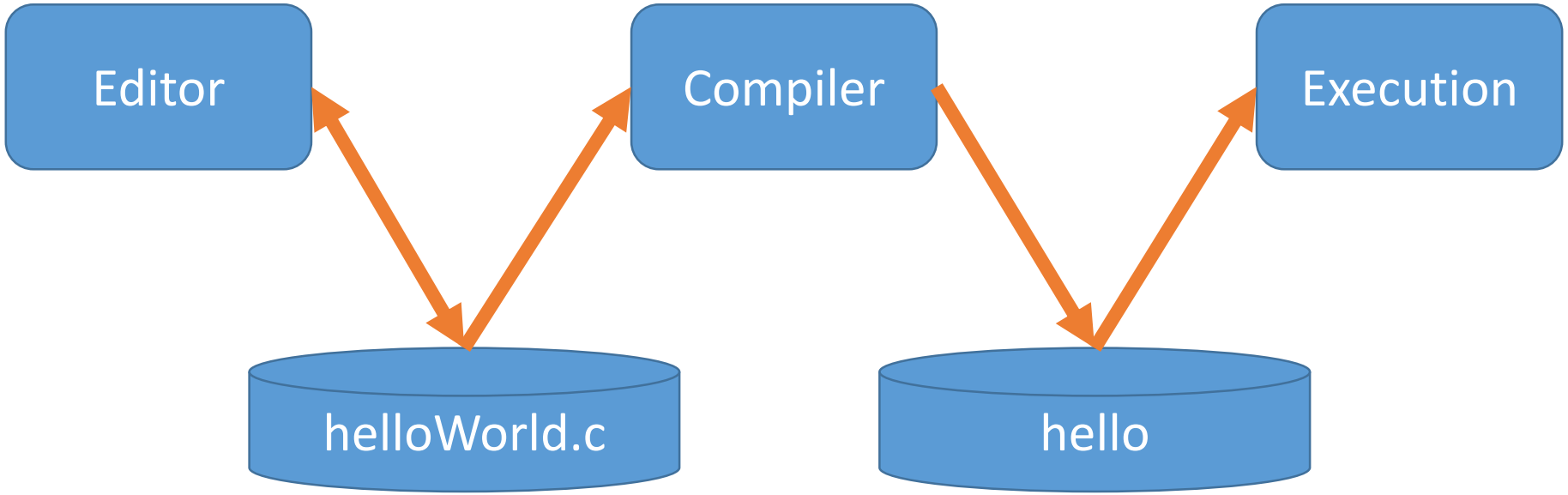
```
#include <stdio.h>
```

Tells compiler to use standard Input/Output (IO) library functions

```
int main() {
        printf("Hello World!\n");
        return 0;
}
```

Definition of the "main" function

C Program Commands (Linux Lab)



gedit helloWorld.c&

gcc –g –Wall –o hello helloWorld.c

./hello arg1 arg2