COMPUTER SCIENCE 1: STARTING COMPUTING CSCI 1300

Ioana Fleming / Vipra Gupta
Spring 2018
Lecture 2

University of Colorado
Boulder

Administrative Details

Course syllabus on Moodle http://moodle.cs.colorado.edu/

CSCI 1300 – Fleming/Gupta - CS 1: Starting Computing Spring 2018

Enrollment key: 1300fg

Dr. Ioana Fleming - ioana.fleming@colorado.edu Office Location: ECOT 735
Vipra Gupta – vipra.gupta@colorado.edu Office Location: ECOT 524
Office Hours: posted on Moodle – Office Hours Calendar
or by appointment

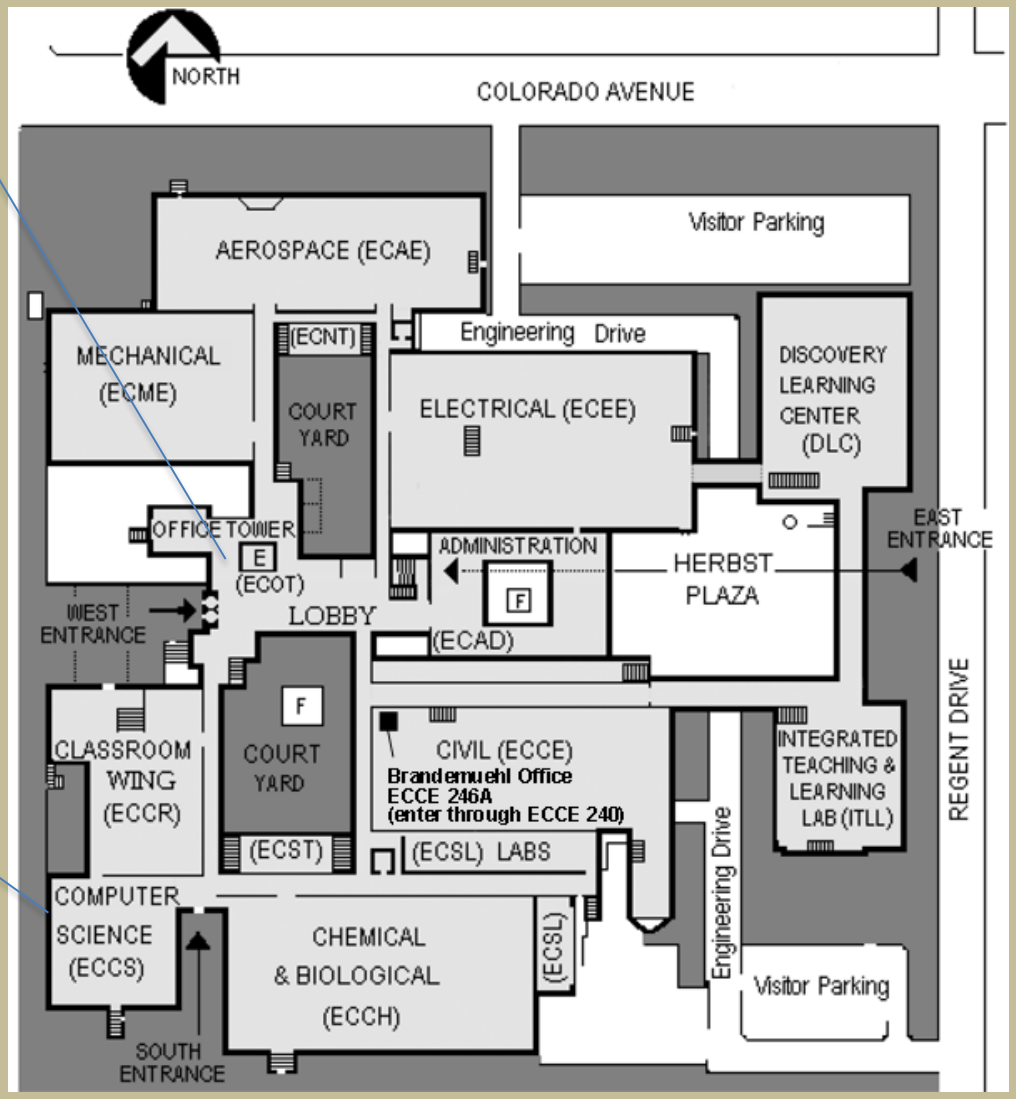
Recitation:
Weekly, mandatory 75 minute lab with recitation activity.
Ask questions about assignments and get extra help.

Office hours:
Homework and topic help from TAs, CAs, and me.
GO! Seriously, GO! *Just make sure you GO prepared*.

Math Building

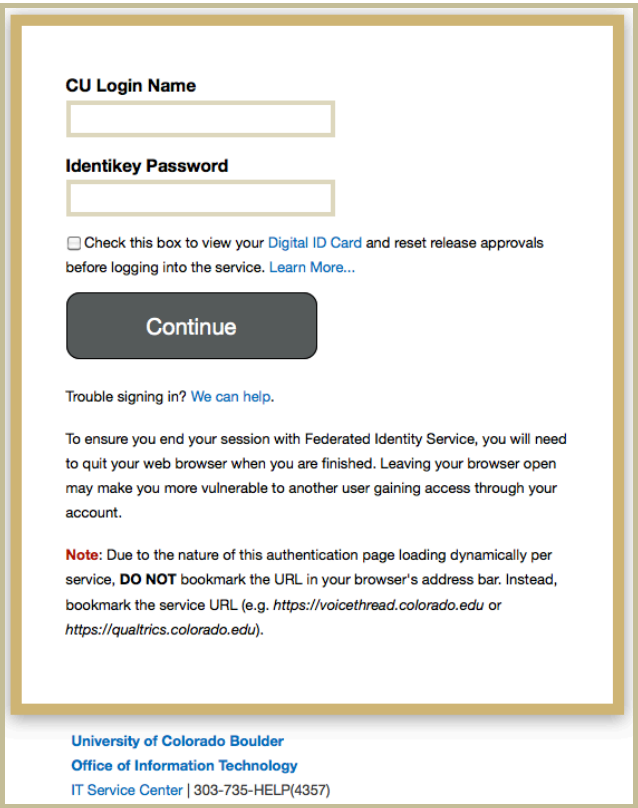
Elevators to the 7th Floor to find CS Department. Some OH rooms on the 8th floor

Area where some CA and TA office hours will be held



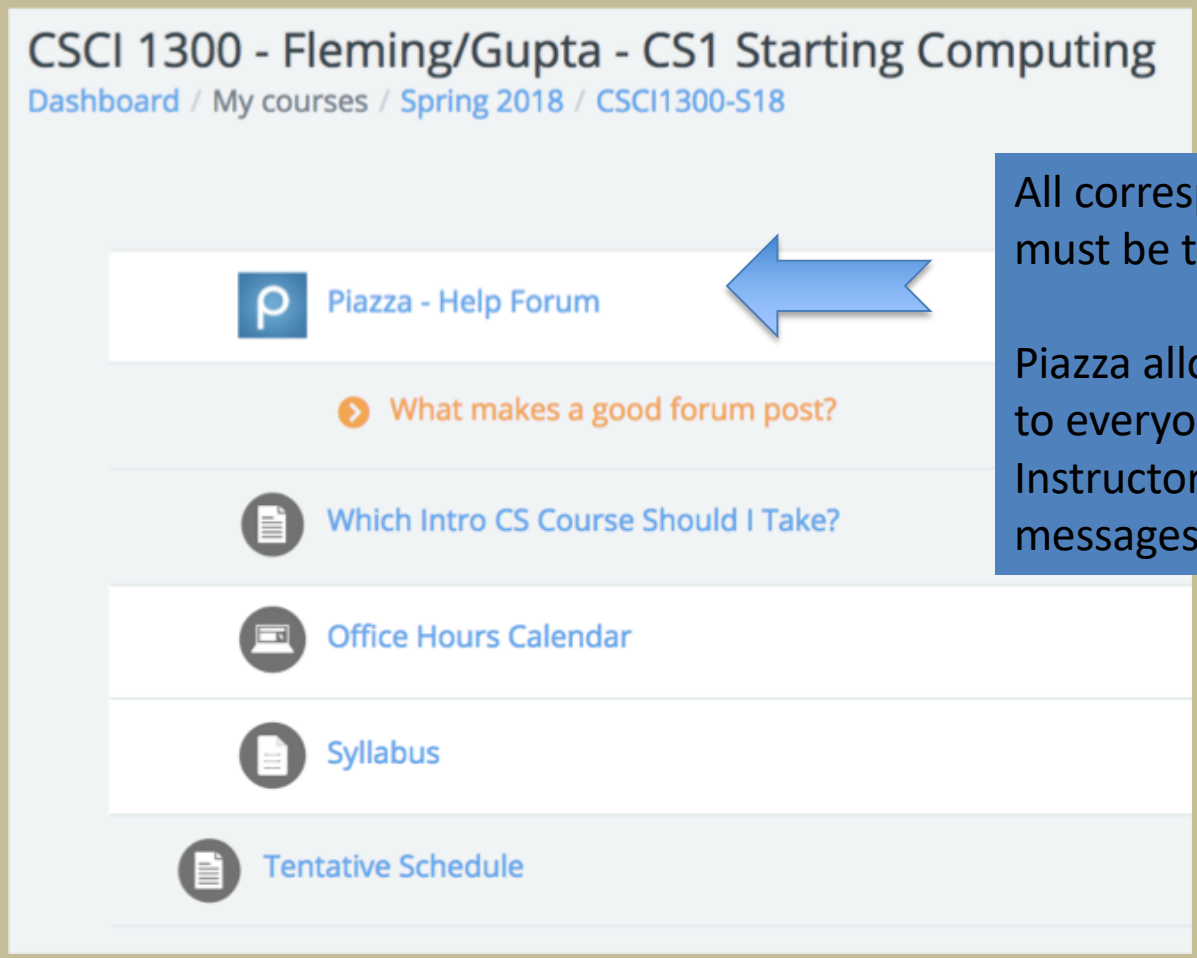
Moodle

Access:
https://moodle.cs.colorado.edu



Enrollment Key: 1300fg

Piazza: CSCI 1300 S18
CSCI 1300 - Fleming/Gupta - CS1 Starting Computing



All correspondence for this course must be through Piazza.

Piazza allows you to post question to everyone (peers, CAs, TAs, Instructor) and also send private messages to instructor or TA.

Students of Concern Team

https://www.colorado.edu/studentaffairs/student-concern

What if you just want to learn a little bit of computer programming (or you are locked out of this course)

CSCI 1200: The Art of Computational Thinking

Teaches computational thinking and techniques for writing computer programs using the Python programming language. The course is intended for students who realize that obtaining computational skills is beneficial to all fields of study, but who have little or no experience in programming or are not Computer Science majors. Students will be expected to create computer programs to solve problems in a range of disciplines.

Note: CSCI 1200 does not count toward Computer Science credit requirements for the Computer Science B.A., B.S., or minor

CSCI 1320: Intro Progr. Engineers (Matlab, C++)

Recitation

- You must attend recitation each week
- Your TA will take attendance each week
- The recitation materials will be posted at the beginning of the week
    - You can finish everything before your recitation (especially if your Recitation is on Thursday) – just come and show your TA. Stay to help others.
    - You can start working in recitation and finish by the end
    - You can start working and finish later. You must submit by the end of Saturday
- If you are done, you can ask questions about homework
- If you need to miss, make arrangements to go to another recitation, *with both TAs.*

Agenda

- Abstraction and Computational Representations
- Algorithms
- Picobot

Next week:

- Pseudocode - Monday
- C++ basics

Abstraction

- Preserving what is relevant in a given context and forgetting the irrelevant information in that context
- Replacing a complex real world object or task with a simpler and understandable model
- Moving from the specific to more general description
- Establishing a level of complexity in which to interact with an object, suppressing the more complex details below the current level.

High Level Abstraction for Making a Cake

Make a Cake:

- Drive to Store
- Buy Ingredients
- Drive Home
- Bake the Cake

This is the ***algorithm*** for making a cake.
It uses high level abstractions to make the algorithm easy to understand.

High Level Abstraction for Making a Cake

Make a Cake:

- Drive to Store
- Buy Ingredients
- Drive Home
- Bake the Cake

- Get into and Start the Car
- Drive to King Soopers
- Park the Car
- Turn the Car Off
- Get Out of Car

The high level abstractions can be refined to provide more details. The steps on the right describe the algorithm for "Drive to Store".

High Level Abstraction for Making a Cake

Make a Cake:

- Drive to Store
- Buy Ingredients
- Drive Home
  - Get into and Start the Car
  - Drive to Your House
  - Park the Car
  - Turn the Car Off
  - Get Out of Car
- Bake the Cake

High Level Abstraction for Driving Somewhere

- Get into and Start the Car
- Drive to Store
- Park the Car
- Turn the Car Off
- Get Out of Car

- Get into and Start the Car
- Drive to Your House
- Park the Car
- Turn the Car Off
- Get Out of Car

High Level Abstraction for Driving to a ***Destination***

- Get into and Start the Car
- Drive to ***Destination***
- Park the Car
- Turn the Car Off
- Get Out of Car

Create a High Level Abstraction for "Getting INTO and STARTING Car"

- Discuss this problem with 2 or 3 of your neighbors

5 Building Blocks for Computational Representations

1. Create a variable to store a value for later use
2. Modify the value of a variable
3. Get input or generate output
4. Check if a statement is True or False
5. *Repeat a statement or collection of statements*
6. *Encapsulating a collection of statements*

5 Building Blocks for Computational Representations

1. Create a variable to store a value for later use

Examples:

```
lemons = 5

oranges = 4

fruit = lemons + oranges
```

```
celsius = 15

fahrenheit  = celsius * 9 / 5 + 32
```

5 Building Blocks for Computational Representations

2. Modify the value of a variable

Examples:

lemons = 5

oranges = 4

fruit = lemons + oranges

fruit = fruit + bananas

celsius = 15

fahrenheit = celius * 9 / 5 + 32

5 Building Blocks for Computational Representations

3. Get input or generate output

Examples:

lemons = 5

oranges = 4

fruit = lemons + oranges

fruit = fruit + bananas

Print out the number of fruits

get the celsius number from user

fahrenheit = celius * 9 / 5 + 32

Print the fahrenheit value

5 Building Blocks for Computational Representations

4. Check if a statement is True or False

Examples:

```
lemons = 5
oranges = 4
fruit = lemons + oranges
fruit = fruit + bananas
Print out the number of fruits

if the number of fruits is larger than 10
    print "lets make a fruit salad"
```

```
get the celsius number from user
fahrenheit  = celius * 9 / 5 + 32
Print the fahrenheit value

If fahrenheit is less than or equal 32
    display "its freezing in here"
```

5 Building Blocks for Computational Representations

5. Repeat a statement or collection of statements

Examples:

```
lemons = 5
oranges = 4
fruit = lemons + oranges
fruit = fruit + bananas
Print out the number of fruits

if the number of fruits is larger than 10
    print "lets make a fruit salad"
    for each fruit
        cut fruit into pieces
```

```
While any piece of fruit is bigger than bite sized
    select largest piece of fruit
    cut selected piece of fruit into two pieces
```

Pseudo Code

pseu·do·code
ˈsōōdōˌkōd/

A notation resembling a simplified programming language for describing algorithms

- Intended for human readability, not a computer's
- Does not need to be syntactically correct code
- Provides a language independent way to describe the steps of an algorithm

Algorithms

- Step-by-step procedure for solving a problem or accomplishing some task
- When your algorithm has enough detail, you are usually writing in pseudo code

Algorithm for Finding something to watch on TV

Algorithm 1

- Turn on TV
- Watch TV

Algorithm 2

- Turn on TV
- Flip thru all stations, rating each one
- Watch Highest rated program

Algorithm 3

- Turn on TV
- Flip thru your top 5 favorite stations, rating each one
- Watch Highest rated program

Which is best? What are pros and cons of each?

Algorithm for Counting Number of Students in the Lecture Hall

1. Starting at the first row
2. Count each student in the row
3. Add it to the total
4. If there are more rows, proceed to the next row and repeat from step 1

Algorithm for Counting Number of Students in the Lecture Hall

- Count each student one-by-one
- Count each student by two's
  - Takes half the time

Can we do it more efficiently?

Algorithm for Counting Number of Students in the Lecture Hall

1. Everyone stand up and assign yourself the the count of 1
2. Find someone else standing up and exchange numbers, each person should sum them together, agree on and remember the sum
3. The person farthest from the podium sits down
4. If more than one person is standing, repeat from step 2

Before Classes Next Week

- Read the **Syllabus** – post messages with any questions to Piazza general forum or to your TA
- Read the Pseudocode examples (link on Moodle)

- Read pages 7-15 of the **C++ Tutorial**

Picobot

- Let's go to the Picobot page
- Questions?