

COMPUTER SCIENCE 1: STARTING COMPUTING CSCI 1300



Ioana Fleming / Vipra Gupta
Spring 2018
Lecture 3



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.*

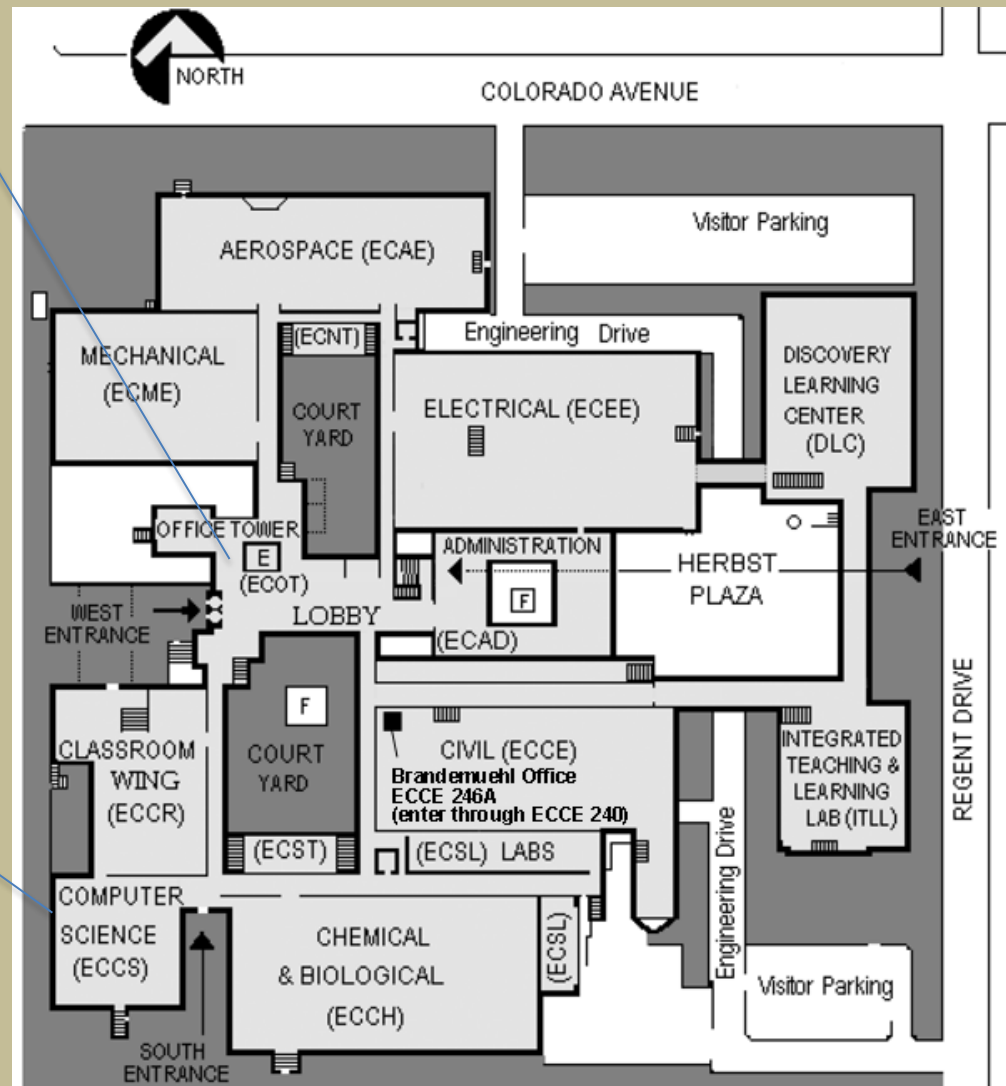


University of Colorado
Boulder

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>

CU Login Name

Identikey Password

☐ Check this box to view your [Digital ID Card](#) and reset release approvals before logging into the service. [Learn More...](#)

Continue

Trouble signing in? [We can help.](#)

To ensure you end your session with Federated Identity Service, you will need to quit your web browser when you are finished. Leaving your browser open may make you more vulnerable to another user gaining access through your account.

Note: Due to the nature of this authentication page loading dynamically per service, **DO NOT** bookmark the URL in your browser's address bar. Instead, bookmark the service URL (e.g. <https://voicethread.colorado.edu> or <https://qualtrics.colorado.edu>).

University of Colorado Boulder
Office of Information Technology
IT Service Center | 303-735-HELP(4357)

Enrollment Key: 1300fg





University of Colorado
Boulder


CSCI 1300 Fall 2017


Piazza: CSCI 1300 S18


CSCI 1300 - Fleming/Gupta - CS1 Starting Computing


CSCI 1300 - Fleming/Gupta - CS1 Starting Computing
[Dashboard](#) / [My courses](#) / [Spring 2018](#) / [CSCI1300-S18](#)


 [Piazza - Help Forum](#) 

 [What makes a good forum post?](#)

 [Which Intro CS Course Should I Take?](#)

 [Office Hours Calendar](#)

 [Syllabus](#)

 [Tentative Schedule](#)

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.



Agenda

- Algorithms
- Pseudocode
- Picobot

Rest of the week:

- C++ basics



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*



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



Pseudo Code

pseu·do·code
'soodō,kōd/

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



Create an Algorithm for

- Summation of all values in a list

sum = 0

for each value in the list

sum = sum + value



Create an Algorithm for

- Alternative

Create a variable names sum and give it the value 0

For each value in the list (loop begins):

 add the value to the existing value of sum

Loop ends here

Note: for product – similar, but *prod* = 1



Create an Algorithm for

- Finding the max value in a list

max = first value in list

for each value in the list

if max < value

max = value



Example 2 (from Moodle link)

You put \$10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?



Bank example

Do it by hand:

year	interest	balance
0		10000.00
1	$10000.00 \times 0.05 = 500.00$	$10000.00 + 500.00 = 10500.00$
2	$10500.00 \times 0.05 = 525.00$	$10500.00 + 525.00 = 11025.00$
3	$11025.00 \times 0.05 = 551.25$	$11025.00 + 551.25 = 11576.25$
4	$11576.25 \times 0.05 = 578.81$	$11576.25 + 578.81 = 12155.06$

You keep going until the balance is at least \$20,000. Then the last number in the year column is the answer.



Bank example

year	interest	balance
0		10000.00
1	$10000.00 \times 0.05 = 500.00$	$10000.00 + 500.00 = 10500.00$
2	$10500.00 \times 0.05 = 525.00$	$10500.00 + 525.00 = 11025.00$
3	$11025.00 \times 0.05 = 551.25$	$11025.00 + 551.25 = 11576.25$
4	$11576.25 \times 0.05 = 578.81$	$11576.25 + 578.81 = 12155.06$

Start with a year value of 0, a column for the interest, and a balance of \$10,000. Repeat the following steps while the balance is less than \$20,000:

- Add 1 to the year value.
- Compute the interest as $\text{balance} \times 0.05$ (i.e., 5 percent interest).
- Add the interest to the balance.

Report the final year value as the answer.



Bank example

Start with a year value of 0, a column for the interest, and a balance of \$10,000.

Repeat the following steps while the balance is less than \$20,000:

- Add 1 to the year value.
- Compute the interest as $\text{balance} * 0.05$ (i.e., 5 percent interest).
- Add the interest to the balance.

Report the final year value as the answer.

More formal, keeping in mind the rules of representations:

```
year = 0
balance = 10000
while balance is less than 20000:
    year = year + 1
    interest = balance * 0.05
    balance = balance + interest
output(year)
```



Bank example

More formal, keeping in mind the rules of representations:

```
year = 0
balance = 10000
while balance is less than 20000:
    year = year + 1
    interest = balance * 0.05
    balance = balance + interest
output(year)
```

The exact wording is not important. What is important is that pseudocode describes a sequence of steps that is

- Unambiguous
- Executable
- Terminating



Bank example

More formal, keeping in mind the rules of representations:

```
year = 0
balance = 10000
while balance is less than 20000:
    year = year + 1
    interest = balance * 0.05
    balance = balance + interest
output(year)
```

Suppose the numbers (\$20,000, 5 percent, \$10000) were user selectable. Are there values for which the algorithm you developed **would not terminate**? If so, change the algorithm to make sure it always terminates.



Bank example

More formal, keeping in mind the rules of representations:

```
year = 0
balance = 10000
while balance is less than 20000:
    year = year + 1
    interest = balance * 0.05
    balance = balance + interest
output(year)
```

1. Final balance < initial balance
 - It will not enter the while loop
 - But it will terminate. Output will be 0 years (final balance already)
2. Negative interest rate
 - Balance will never reach target. Not terminating: **infinite loop**
3. Other situations?



Bank example

Modify to account for user error:

```
year = 0
balance = value entered by the user
if balance < 0
    ask user to enter again
rate = value entered by user
if rate < 0
    ask user to enter again
target = value entered by the user
if target < balance
    output ("target already achieved. Enter another target value")
while balance is less than target:
    year = year + 1
    interest = balance * rate
    balance = balance + interest
output(year)
```



Assignment #1

- Writing algorithms to solve problems
 - provide 4-6 steps
 - If a human would need more information to complete the task
 - Break down the steps into 4-6 sub-steps
- Do your best at pseudocode – ask TA / CA for feedback in recitation.



Picobot

- Any New Questions?

