Week 1

Philip Caplan

CSCI 0145 - Introduction to Computing (Spring 2021)
A little about me... 

- Please call me Philip.
- I’m from Montreal, studied at McGill & MIT.
- I like to hike, practice yoga, and play drums (I’m really bad but I like making noise and hitting things).
- I have type I diabetes: I might need to test my blood sugar and/or eat candy.
- My dog Leila sometimes hangs out behind me during meetings.
I work on getting accurate simulations of physical phenomena.

involves computational geometry and computer graphics!

http://mdolab.engin.umich.edu

not my work but gives a good idea of the things I do:
https://www.youtube.com/watch?v=KSqT6fuHn7A
I’m also a big fan of giraffe art.
I’m also a big fan of giraffe art.
I’m also a big fan of giraffe art.
I’m also a big fan of giraffe art.
...so...I like using examples from computer graphics (I hope that's okay)

please complete **background survey** so I can learn about your interests!
https://forms.gle/KbQkWxmHXRf7rdjY8
A little about you...

In random breakout rooms (5 minutes):

- Introduce yourself! What are your non-academic interests?
- What is something you are excited to learn this semester?
- What is something you are looking forward to doing in 2021?
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
What is this course about?

**in a nutshell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
What is this course about?

in a nut shell: how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
- represent and store data (numbers, text and images),
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
- represent and store data (numbers, text and images),
- design simple circuits using basic logic gates,
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
- represent and store data (numbers, text and images),
- design simple circuits using basic logic gates,
- describe a simplified von Neumann architecture,
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
- represent and store data (numbers, text and images),
- design simple circuits using basic logic gates,
- describe a simplified von Neumann architecture,
- design and implement simple programs in an assembly level language.
What is this course about?

**in a nut shell:** how to solve problems with a computer

By the end of this course, you will be able to:

- solve problems with procedural statements, functions, basic data structures, and objects,
- design, implement, document, and debug medium-sized programs in Python,
- analyze and implement iterative and recursive solutions to computational problems,
- analyze asymptotic complexity of algorithms,
- represent and store data (numbers, text and images),
- design simple circuits using basic logic gates,
- describe a simplified von Neumann architecture,
- design and implement simple programs in an assembly level language.

I really want you to come out of this class with the skills to solve problems efficiently.
How will we achieve these learning goals?

Resources:
- readings (notes posted on website),
- videos posted to Panopto,
- labs on Wednesdays and Fridays.

Assessments:
- 15% labs (including pre-lab quizzes), in groups of 3-4, effort-based
- 40% homeworks (about 10), individual
- 15% midterm exam (April 7th)
- 25% final exam (exam week)
- 5% participation
Our team will help you achieve these goals

- **Instructor**: Philip Caplan
- **ASIs**: Catherine Miller (+labs) & Robert Lichenstein
- **Tutors**: Elva Osorio (+labs), Franklin Cardona (+labs), Charlotte Gray (+labs) Farhan Islam, Sabrina Templeton, Emma Doubman, Manny Fors
- **Graders**: Griffin Pizzano, Maggie Swanson, Nestor Orozco-Llamas

[go.middlebury.edu/cstutors](go.middlebury.edu/cstutors)
Weekly schedule

Monday
- no class!
- I will have office hours from 9am-11am
- readings posted to course website
- video posted to Panopto
- complete homework from previous week by 11:59pm

Wednesday
- read course notes and watch video by today
- complete pre-lab quiz on Canvas before lab (class) session
- work on labs with classmates!

Friday
- work on labs with classmates!
- submit labs for the week (effort-based) by 11:59pm (submit on repl.it as well as Canvas)

bookmark course calendar: https://philipclaude.gitlab.io/cs145s21/calendar.html
Wait, how will we do group work? Everyone is remote! → repl.it
Let’s have a discussion about group work.

- Why do you think it’s important to practice working with someone else?
- How do you approach "sticky" situations when working with someone?
  - What if you don’t understand something?
  - What if you sense your partner doesn’t understand something?
  - What if you feel the division of work is not equal?
Let’s have a discussion about group work.

- Why do you think it’s important to practice working with someone else?
- How do you approach "sticky" situations when working with someone?
  - What if you don’t understand something?
  - What if you sense your partner doesn’t understand something?
  - What if you feel the division of work is not equal?

When submitting your labs at the end of each week, there will be a question about how your group work went.
How should we communicate with you and each other?

- Personal matter? **email**
- Otherwise? **slack**
- Our team can help you with conceptual and clarification questions, but we cannot search for bugs.

**join our slack workspace!**

https://join.slack.com/t/middlebury-cs145-s21/shared_invite/zt-msoh5s5d-lEsjfujxpTDHNb5xIUUmHg

- Post questions in **general**, **labs**, **homeworks**, etc. channels.
- Send private messages to each other.
- Please do not send us (me, ASIs, tutors) private messages on slack about the material.
- Participation grade may be boosted based on regular, thoughtful responses to your classmates’ questions in slack channels,
- Be kind and respectful to your classmates.
Where are all the resources?

syllabus
https://philipclaude.gitlab.io/cs145s21/index.html

- **calendar:** https://philipclaude.gitlab.io/cs145s21/calendar.html (also expandable within each reading),
- **readings:** https://philipclaude.gitlab.io/cs145s21/notes/week01.html
- **labs:** https://middlebury.instructure.com/courses/8091 (Canvas)
- **homeworks:** https://middlebury.instructure.com/courses/8091 (Canvas)
- **reading quizzes:** https://middlebury.instructure.com/courses/8091 (Canvas)
- **videos:** https://midd.hosted.panopto.com/Panopto/Pages/Sessions/List.aspx#folderID=%22d4560c21-bfb5-47b1-b19f-acd600ca7b68%22 (Panopto)
- **repl’s:** https://repl.it/team/cs145s21
Algorithms & bugs
What is an algorithm?

An algorithm is a sequence of steps used to solve a problem.
Algorithms can be described using pseudocode.

Ingredients:

- **natural language**, e.g. English
- **mathematical notation** (more on this in CSCI 200)
- **control structures**:
  - **repetition**: repeat, for, do, while,
  - **conditional**: if, then, else if, else,
  - **abstraction**: function, procedure, class.

Once you have designed your algorithm, then you can **implement it with actual code**.
Following an algorithm is like following a recipe.

Instructions

1. To make the starter: Mix everything together to make a soft dough. Cover and let rest at room temperature for about 14 hours; overnight works well. The starter should have expanded and become bubbly.

2. To make the dough: Mix and knead everything together — by hand, mixer or bread machine set on the dough cycle — to make a soft, somewhat smooth dough; it should be cohesive, but the surface may still be a bit rough. If you’re using a stand mixer, knead for about 4 minutes on medium-low speed (speed 2 on a KitchenAid); the finished dough should stick a bit at the bottom of the bowl.

3. Place the dough in a lightly greased medium-sized bowl, cover the bowl, and let the dough rise for 90 minutes, gently deflating it, folding the edges into the center, and turning it over after 45 minutes.

4. Turn the dough out onto a lightly greased work surface. Gently deflate it, and divide it into three equal pieces.
Following an algorithm is like following a recipe.

Instructions

1. To make the starter: Mix everything together to make a soft dough. Cover and let rest at room temperature for about 14 hours; overnight works well. The starter should have expanded and become bubbly.

2. To make the dough: Mix and knead everything together — by hand, mixer or bread machine set on the dough cycle — to make a soft, somewhat smooth dough; it should be cohesive, but the surface may still be a bit rough. If you’re using a stand mixer, knead for about 4 minutes on medium-low speed (speed 2 on a KitchenAid); the finished dough should stick a bit at the bottom of the bowl.

3. Place the dough in a lightly greased medium-sized bowl, cover the bowl, and let the dough rise for 90 minutes, gently deflating it, folding the edges into the center, and turning it over after 45 minutes.

4. Turn the dough out onto a lightly greased work surface. Gently deflate it, and divide it into three equal pieces.

Once you have designed your algorithm, then you can implement it with actual code
What is an algorithm?

An algorithm is an unambiguous sequence of steps used to solve a problem.
Computers don’t make mistakes! Unexpected behavior is called a "bug."

distance = 10 feet
Computers don’t make mistakes! Unexpected behavior is called a "bug."

distance = 7 feet
Computers don’t make mistakes! Unexpected behavior is called a "bug."

\[ \text{distance} = 4 \text{ feet} \]
Computers don’t make mistakes! Unexpected behavior is called a "bug."

distance = 1 feet
What to do?

for Friday:
- read the notes: https://philipclaude.gitlab.io/cs145s21/notes/week01.html
- complete the background survey: https://forms.gle/KbQkWxmHXRF7rdjY8
- come ready to work on lab in groups: when you’re done the lab you’re free to go!
- please claim your accommodations by sending me your LOA (if necessary).

for next week:
- homework 1 due on Monday: https://middlebury.instructure.com/courses/8091/assignments/113417
- read lecture notes & watch video (posted on Monday)
- complete pre-lab quiz before Wednesday lab
- labs on Wednesday and Friday