

# Week/Topic 1: Welcome & First Program

*Goals: By the end of this topic, we will ...*

- *meet the course instructional staff & gain insights into what this course is all about*
- *meet classmates and make the first steps in co-creation of a community of learners*
- *contextualize this course in the broader field*
- *be introduced to some of the policies in the course & review the syllabus/course pack*
- *write our first program in Python and play*
- *expand on our definition of computation to create a model of a computer*
- *introduce pair programming as a construct*

Who are we?



**Alicia**  
Course Instructor  
(they/them)

Alicia is pronounced (IPA): [ ə.ɪ.ˌfə ]

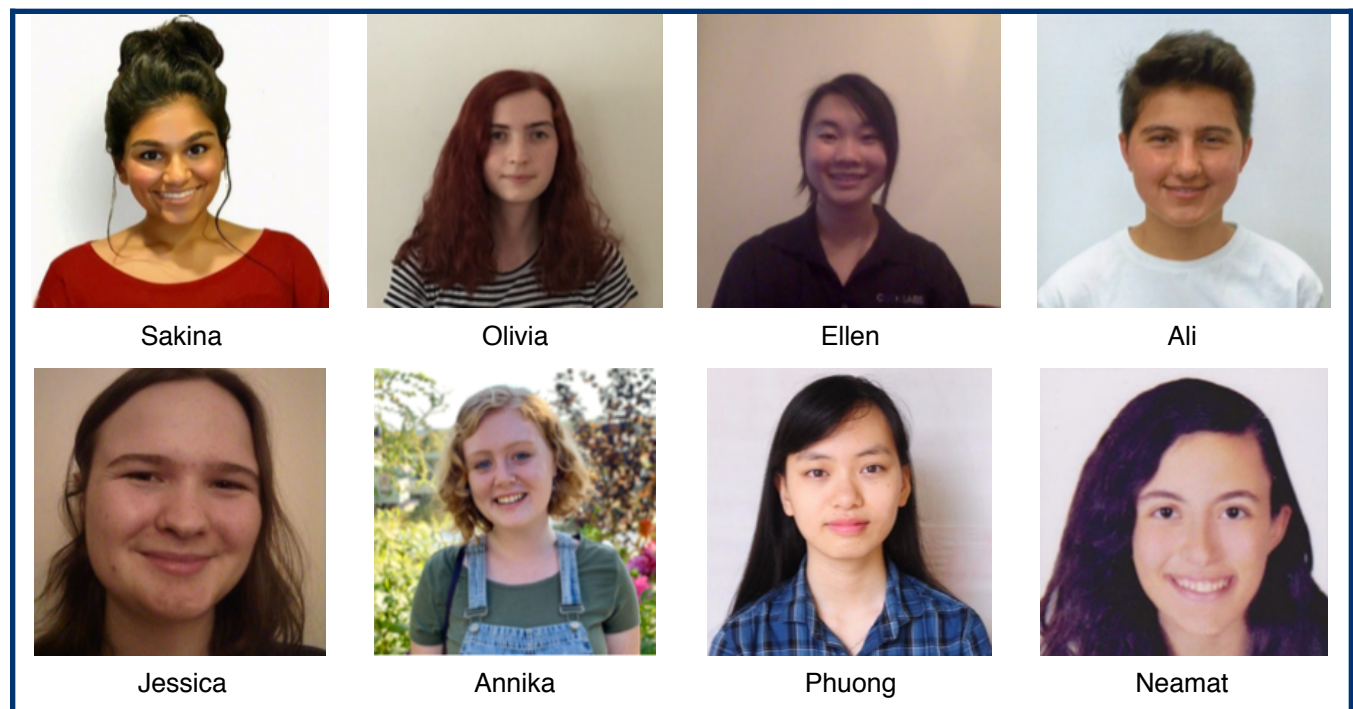


**Mariah**  
Pedagogical Partner  
(she/her)



**David**  
Lab Instructor  
(he/him)

*Teaching Assistants (TAs):*



*Reader/Grader: Hiba*

- What questions do you have?—

---

Who are you?

Opening questions...

1. What is one thing that you are really good at?
2. What brought you to this class?
3. What is computer science?

DO: Take a moment and think about these questions and then introduce yourself to the people around you and share your answers.

ALL: Discuss as class.

---

### Class Homework!

1. Read over the course webpage: <https://amgrubb.github.io/csc111/>
2. Fill out doodle poll: <https://doodle.com/poll/xwu8fsmnudn73umy>
3. Read over the entire syllabus, bring questions to class.
4. On Moodle: "Upload your photo... asap!"

---

What is Computer Science (in 2019):

- Essential tools in modern society
- Personal computers (desktops and laptops)
  - write papers, manage personal finances, ....
  - entertainment: games, video, audio, ....
- Business computers
  - day-to-day operations: payroll, billing, ...
  - customer service: web sites, customer data, ...
- Embedded computers
  - microchips used as controllers in cars, phones, wearables, ....
- Supercomputers
  - large "number crunchers" used in scientific research and other areas
  - parallel processing: from a few dozen to a few thousand CPU chips



*History lesson in a future class.*

## What Computers Can't Do...

### **Problem: Choose a college or university to attend?**

College experience? Multiple factors? Could be computer assisted? Who else would you trust with this decision? Hard problem!



### **Problem: What's happening in this picture?**

Dog, toy, halloween costume? Humans are really good at this?

Humans develop knowledge in experience and culture. Computers need to be trained?

Will computers of the future be able to understand this photo?



### Problem: Win a game of chess?

The rules of the game are simple, the computer examines *all possible moves* = practical limitation ( $10^{43}$  possible games -> supercomputer reviewing  $10^{12}$  boards/sec would need  $10^{21}$  years). Computers can learn patterns like grand masters, who do not consider all possible moves.



```
while (True):  
    print("looping...")
```

### Problem: Detect an infinite loop.

Known as “the halting problem”.

Impossible to detect if a program is stuck in an infinite loop.

---

### Patterns

Q: What do you notice?

Q: What tasks are computers really good at and what tasks are humans really good at?

A problem might not be solvable by humans or computers because:

- some attributes are not quantifiable (quality of life at a university)
- it is impractical (chess)
- it is impossible (halt checker)

Problems that humans can solve but computers can't are often described in terms of “intelligence”

- natural language processing, planning, design, ...
- an active area of CS research: artificial intelligence (AI)

### Definition: Computation

“a sequence of well-defined operations that lead from an initial starting point to a desired final outcome”



mathematical



logical

Computation is carried out by humans and machines.

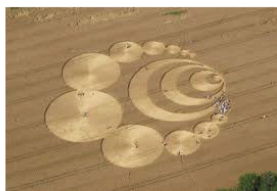
# Activity: Computational Thinking -> Creating Crop Circles



Pictures of crop circle tourism in England  
nationalgeographic.com



The Crop Circle Mystery: A Closer L...  
livescience.com



Crop circle - Wikipedia  
en.wikipedia.org



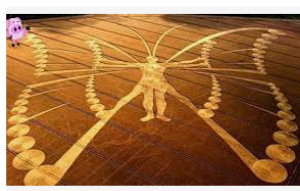
Crop circles demystified: how the ...  
telegraph.co.uk



crop circles  
lifedeathprizes.com



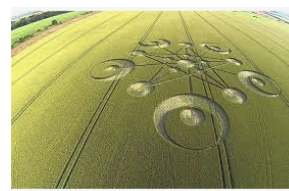
Webb: Crop circles plagued Illinois in...  
courierpress.com



10 Amazing Crop Circles That Have Left ...  
youtube.com



The science behind crop circles  
zmscience.com

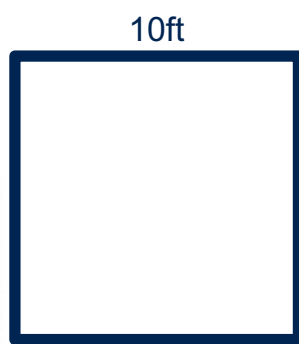


Wiltshire Crop Circles - Eden Saga ...  
eden-saga.com

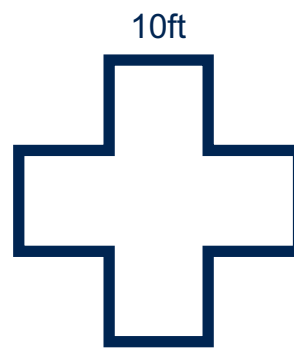


Crop Circle at Ansty, Nr Sa...  
cropcircleconnector.com

With a partner (or group of 3), write down the instructions to make your tractor complete each of the following tasks.



Task 1



Task 2

the 1<sup>st</sup>  
letter of  
your first  
name

Task 3

Discussion:

- What do you notice?
- Were there any letters you couldn't draw?
- Can you tell in advance which shapes are impossible?

With a well defined set of operations, **what can we build?** When do the **rules not apply?** (Meta-reasoning).

## Definition: Computer Science

“the study of computation”

- **Problems** that can be solved computationally
- **Languages** used to describe computational processes
- **Machines** that carry out those processes
- **Theoretical limits** of computation
- **Applied computation** that is computational solutions to problems in math, science, medicine, business, education, journalism, ...

---

Learning Computer Science...

You have already done computer science! 🤖 😊

Reprisal of Opening questions...

1. What is one thing that you are really good at?

(New) How did you get really good at this thing?

(New) What from that experience can you apply to this course?

~~2. What brought you to this class?~~

3. What is computer science?

(New) Were any of your preconceived notions about computer science incorrect?

(New) 4. How will you know if you are learning computer science?

Thoughts:

- Growth/learning mindset -> this will take practice
- Few terms, little/no memorization, little reading, all about the application of concepts and problem solving
- A new way of thinking (abstraction, computational and systems thinking)
- You will succeed if you do the “right” work and think about your thinking (metacognition)

---

Learning Promises:

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

- Create and document computer programs using correct Python syntax that can be readily understood and used by other programmers.
- Propose algorithms in order to analyze problems that use basic control flow constructs (e.g., if-then statements, loops, functions, lists, simple input-output).
- Demonstrate foundational development techniques, including top-down design, program documentation, modular design, and library usage.
- Understand the high-level internal operation of a computer, including the central processing unit, simple memory management, and the file system.
- Explain core computer science topics, such as complexity, object-oriented programming (OOP), sorting, and recursion.
- Help you develop the Essential Capacities for Smith Students.

**Programming and computational thinking are essential skills for all members of a computer using society, no matter what your ultimate career. It is foundational, like reading, writing, and arithmetic.**

Don't just take my word for it. Listen to **Barack Obama**, "Don't just buy a new video game; make one. Don't just download the latest app; help design it. Don't just play on your phone; program it. **No one's born a computer scientist**, but with a little hard work and some math and science, just about anyone can become one." and the rest of the clever people at <http://blog.teamtreehouse.com/trust-us-trust-them-learn-to-code>

**DO NOT believe you must be a certain kind of person to be good at computer science.**

---

Technology in the Classroom:

What do you find helpful about having a laptop/phone/tablet in classes?

What do you find unhelpful about you or others having a laptop/phone/tablet in classes?


DO: Take a moment and think about these questions and discuss them with the folks around you and share your answers.

ALL: Discuss as class....create technology policy.

---

Help me get to know you.

About me: I can't read *cursive letters*, please PRINT as neatly as you can. 🧐 Take your time.

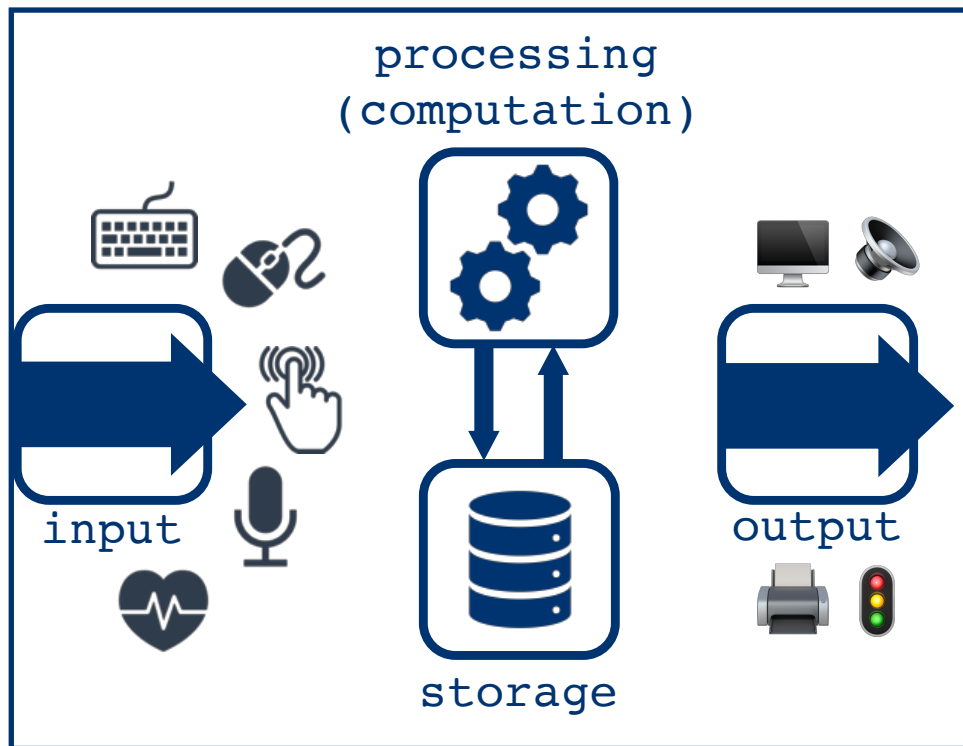
Preferred Name: _____	Pronoun(s): _____	
Full Name: _____	OneCard#: _____	
Major (Intended): _____	Class Year: _____	
Prior Experience: _____	House: _____	Lab: _____
What's one big thing you hope to get out of this course?		
What can I do to help you learn best?		
What's one big question that sometimes keeps you up at night? - OR - If you could build any software application what would it be and why?		

Take five minutes and discuss your answers with your neighbor.

Suggestion: Exchange emails and/or phone numbers...book a study date for next week.

## Model of System

Here is a model of a system.



What systems can you think of that are not computer systems...

input -> process -> output

...much of the world is organized this way (may no include storage).

Programming in this course...

We use:



Cool stuff written using Python:



So what about it?

Python is: (a) multi-paradigm, (b) interpreted language, (c) with dynamic typing, and (d) automatic memory management.

*More on (c) and (d) later.*

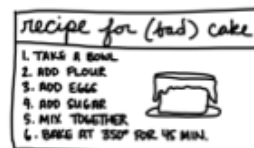
## Programming Paradigms



object-oriented



functional



imperative



declarative





	COMPILER	INTERPRETER
What it takes in:	an entire program	a single line of code (or a single instruction)
What it returns:	intermediate object code (e.g. classes)	<nothing>
Relative speed:	faster	slower
Memory usage:	uses more memory (↑ objects take space ↑)	uses less memory (no intermediate objects)
Work is done:	just once	every time the code is executed
Reports errors:	after checking the entire program	after each instruction is run

---

## Our First Python Program (Demo Time)

1. What it means to be an interpreted language.
2. Hello World
3. Numbers

DO: What do you notice about the code we wrote in the demo. What are the common elements? What type of information might we store?

We will discuss `print( )` later.

## Concept: Variables

Where do we store information? — Variables.

- In CS, a variable is a place to store a piece of data.
- In Python, variables are:
  - declared by giving them a name
  - assigned using the equals sign (assignment statement)

Declaring a variable.

```
num = 9
x = 4.5
y = 6.7
name = "Alicia"
dep = 'CS'
```

Assigning the variable.

## Assignment Statements

The general form of an assignment statement: *variable = expression*

What happens when python evaluates an assignments statement:

1. Evaluate the expression to the right of the = sign. This produces a memory address of the value that is produced by the expression.
  2. Store the memory address in the variable on the left of the = sign.
- Order matters, entering `19 = z` will result in a syntax error.

## Variable Names

The rules for legal Python names:

1. Names must start with a letter or `_`.
2. Names must contain only letters, digits, and `_`.
3. Cannot be controlled phrases (eg. `if`, `else`, `for`) see below.

For Python, in most situations, the convention is to use `pothole_case`. We recommend using the PEP 8 -- Style Guide for Python Code [<https://www.python.org/dev/peps/pep-0008/>].

Some textbooks (and your instructor) occasionally use `camelCase` so you should be comfortable with this as well.

Keywords in Python programming language

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

[<https://www.programiz.com/python-programming/keywords-identifier>]

## Concept: Variable Types

Python can store: algebraic expressions, True/False, numbers, integers, words, letters, decimal numbers, arrays of numbers or letters

A *type* is a set of values and operations that can be performed on those values.

### Numbers

- `int`: integer  
For example: 3, 4, 894, 0, -3, -18
- `float`: floating point number (an approximation to a real number)  
For example: 5.6, 7.342, 53452.0, 0.0, -89.34, -9.5

Floats and integers behave differently when subjected to arithmetic operations.

### Strings

A *string literal* is a sequence of characters.

- `str`: string  
For example "Hello"

Strings in Python start and end with a single quotes (') or double quotes (").

A string can be made up of letters, numbers, and special characters.

If a string begins with a single quote, it must end with a single quote. The same applies to double-quoted strings. You cannot mix the type of quotes.

For example:

```
>>> 'hello'
'hello'

>>> 'short- and long-term'
'short- and long-term'

>>> 'Welcome to CSC111'
'Welcome to CSC111'
>>> "What is 2 + 9?"
'What is 2 + 9?'
>>> "Will this work?"
      File "<stdin>", line 1
          "Will this work?"
              ^
SyntaxError: EOL while scanning string literal

>>> reminder_text = 'Please buy groceries after work'
>>> reminder_text
'Please buy groceries after work'
```

**Question:** How do I store a quotation mark inside a string if I'm already using quotation marks to denote the beginning and end of a string?

Use a backslash before the quote.

```
>>> "The student declared, \"I won't be late for my lab\""
'The student declared, "I won't be late for my lab"'
```

## Concepts: Calculations/Arithmetic and Precedence

In scientific computing, a primary use of Python is statistics and mathematical operations. Computers allow you to perform mathematical operations much faster. Here are the basic operators.

### Arithmetic Operators

Operator	Operation	Expression	English description	Result
+	addition	11 + 56	11 plus 56	67
-	subtraction	23 - 52	23 minus 52	-29
*	multiplication	4 * 5	4 multiplied by 5	20
**	exponentiation	2 ** 5	2 to the power of 5	32
/	division	9 / 2	9 divided by 2	4.5
//	integer (floor) division	9 // 2	9 divided by 2	4
%	modulo (remainder)	9 % 2	9 mod 2	1

### Operator Precedence

You have seen this idea before in math. Do you remember PEMDAS / BEDMAS?

US	CAD	Operation	Operator	Order / Precedence
P	B	Brackets	( )	First / Highest
E	E	Exponents	**	
			- (negation)	
MD	DM	Multiplication and division	*, /, //, %	
AS	AS	Adding and subtracting	+ (addition) - (subtraction)	Last / Lowest

Operations of equal precedence are evaluated left to right.

## Advanced Concept (Optional): Augmented Assignment Operators

We can use the same variable on both sides of an assignment statement as shown below:

```
>>> number = 3
>>> number
3
>>> number = 2 * number
>>> number
6
>>> number = number * number
>>> number
36

>>> score = 50
>>> score
50
>>> score = score + 20
>>> score
70
```

Python first evaluates the expression on the right of the = sign to produce a number and then it makes the variable on the left of the = sign refer to that number. These assignment operations are so common that Python has produced a short form notation to make the statement more concise.

### Augmented Assignment Operators

Operator	Expression	Identical Expression
+=	x = 7 x += 2	x = 7 x = x + 2
-=	x = 7 x -= 2	x = 7 x = x - 2
*=	x = 7 x *= 2	x = 7 x = x * 2
/=	x = 7 x /= 2	x = 7 x = x / 2
//=	x = 7 x //= 2	x = 7 x = x // 2
%=	x = 7 x %= 2	x = 7 x = x % 2
**=	x = 7 x **= 2	x = 7 x = x ** 2

## Concept: Input/Output

### **print()**

- print() outputs information to the console ("the shell")
- Works on lots of different data types (strings, integers, floats, and many more!)
- When print() is called on ("passed") a variable, it outputs the contents

```
>>> x = 7
>>> print(x)
```

The function print is much more flexible than we have seen so far.

```
help(print)
```

### Print multiple strings.

```
>>> print("Smith", "College")
```

### Print strings and numerical variables.

```
>>> year = 2019 #recall assignment statements
>>> print("The year is:", year)
```

Or even print all variables if you'd like, as long as each is separated by a comma.

```
a = "I am learning to"
b = "program in python. Including how to print numbers, such as"
c = 3
print(a, b, c)
```

### **len()**

- len() takes in a string and gives back the string's length (number of characters, including spaces)
- Can be called on string literals ("stuff in quotes") or on variables whose contents are strings
- Unlike print(), len() **returns a value**

### Get the length of a string.

```
>>> len('programming is fun')
18
>>> len("12346")
5
```

### **input()**

- Python has a built-in input() function that allows us to ask the user to type in information
- Input always returns a string. If you want the user to enter something that will be treated as an integer or float, you will need to put it through another function to transform it to the desired type.
- The input() function takes in a value, which will be printed to the console as a prompt:

```
>>> input("Enter some text: ")
Enter some text:
```

In general, we will want to save what the user enters so we can do something with it. This means we need to assign the value returned by the `input()` function to some variable.

```
>>> x = input("Enter some text: ")
Enter some text:
```

or

```
num = input("What number? ")
num = float(num)
num = num + 1 # assign a new value of one greater to num
print("One more than that number is: ", num)
```

### **eval()**

The user's input is always returned as a string, even if they enter only numeric characters. If we want Python to interpret it as a number, we can use the `eval()` function.

```
>>> x = eval(input("Enter some text: "))
```

Then we can manipulate `x` using mathematical operations.

---

## Exercise

- Find a partner, and write a program that asks the user to `input()` two strings:
  - a word
  - a number
- Store the user input in appropriate variables
  - remember: `eval()` will return the numeric value of a string
- `print()` the word the user-specified number of times
  
- Want a challenge? Also ask the user to input a character (a single letter or symbol), and use that to print a banner around the repeated word.

### Possible Solution:

```
my_word = input("Enter a word:")
my_num = eval(input("Enter a number:"))
print(my_word * my_num)
```

---

## Refresher Exercise from Lab 0

Let's write a short program that prints the following:

```
#####  
# CSC 111 #  
#####
```

What do we need?

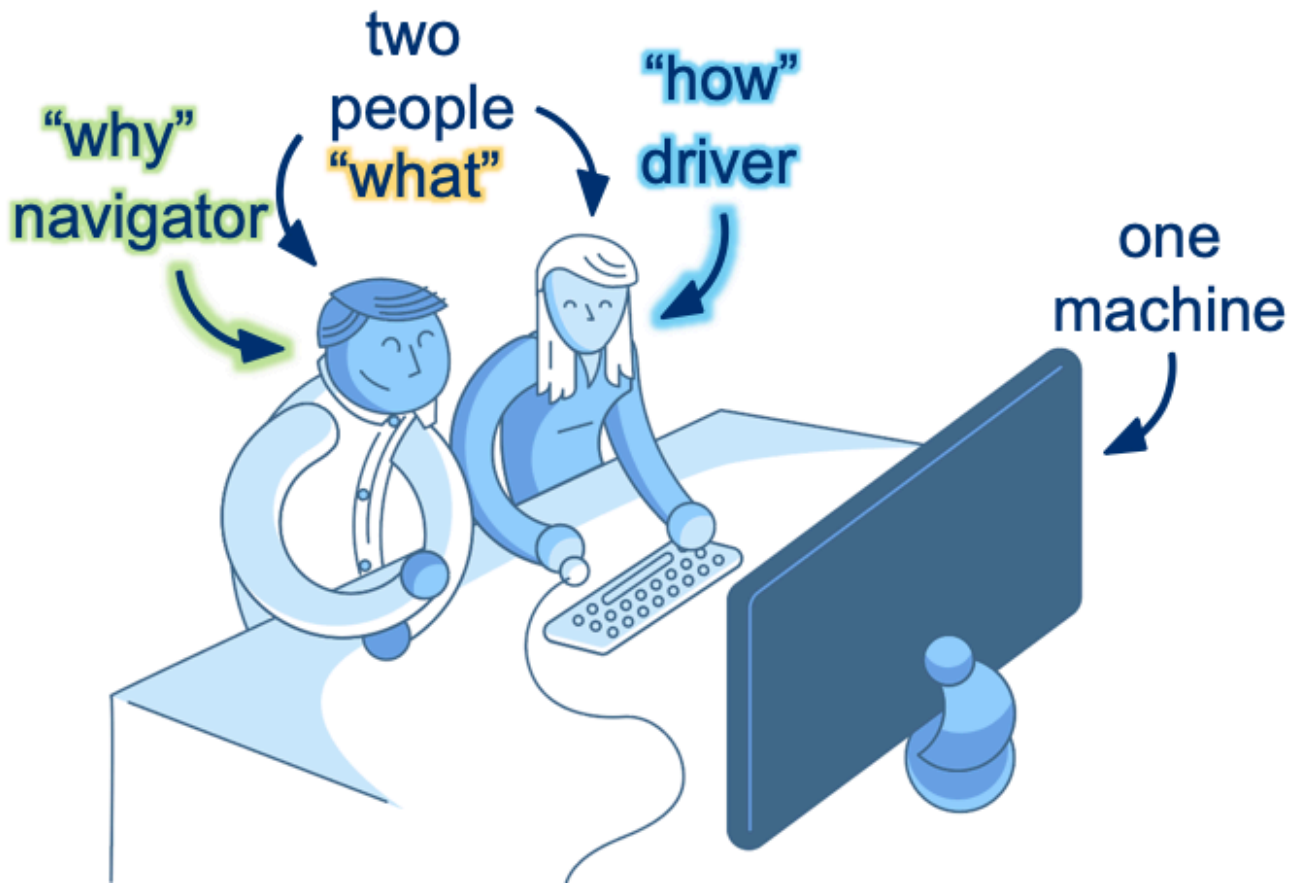
What is our input? What is our output? What computation is required?

**Possible Solution:**

```
thing_to_print = "CSC 111"  
size = len(thing_to_print)+4  
line = "#" * size  
print(line)  
print('#', thing_to_print, '#')  
print(line)
```



Concept: Pair Programming - Lab Preparation



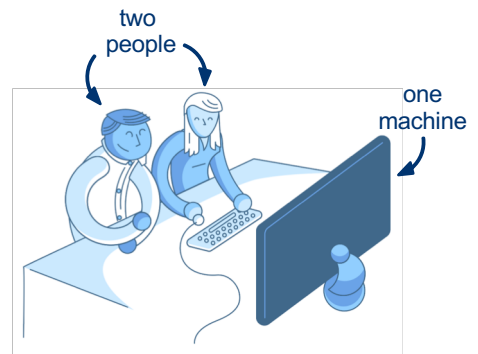
Homework: Watch [https://www.youtube.com/watch?v=u\\_eZ-ae2FY8](https://www.youtube.com/watch?v=u_eZ-ae2FY8)



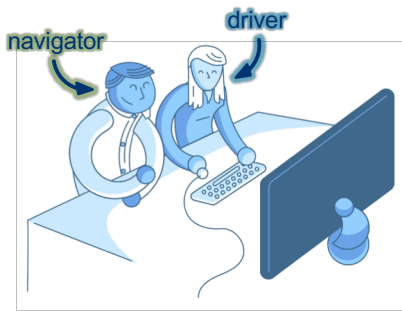
### A problematic (but common) model



### A better model: "pair programming"



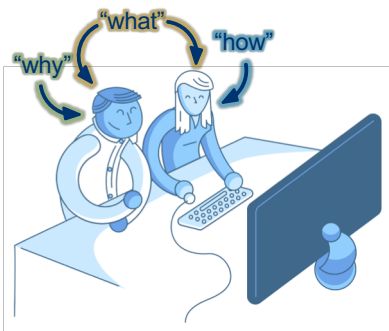
### Two complimentary roles



### A common analogy



### Navigator vs. driver: different focus



### Pairing

