Once you're seated, please respond to the poll at **pollev.com/compunc**

If you are not registered for PollEverywhere, please go ahead and do so before class begins:

https://poll.unc.edu/

Lecture 01

Take on Me

Practice: Primitive Data Types, Variables Introducing: Comments and Statements

Not enrolled in COMP110 yet? Please wait until after class for questions.

If you were not here Tuesday, you cannot be enrolled this point.

Videos 4 - 8 (~35min) – Watch by Thursday 1/16

- These will help orient you with the software we are using this semester and introduce the next two control flow statements: conditionals and loops.

Problem Set 0 – A Card for Someone Special

- Due Friday - 1/17 by 11:59pm
- Posted on the Calendar
- The purpose of this problem set is to ensure you have everything setup correctly.
- This problem is very short and simple and ***is not*** indicative of how challenging problem sets will ultimately be.

Your Software *Might Not* Work Today

- That's totally *OK!*

- If yours is not working, don't stress, just follow along with me!

- Come to the Open House tomorrow in Sitterson 008 from 12-6pm.
  - Want to be a computer science major and have questions?
    This is a great opportunity to talk to students who started out in 110 and are now at various points in their CS degree ranging from sophomores to seniors!

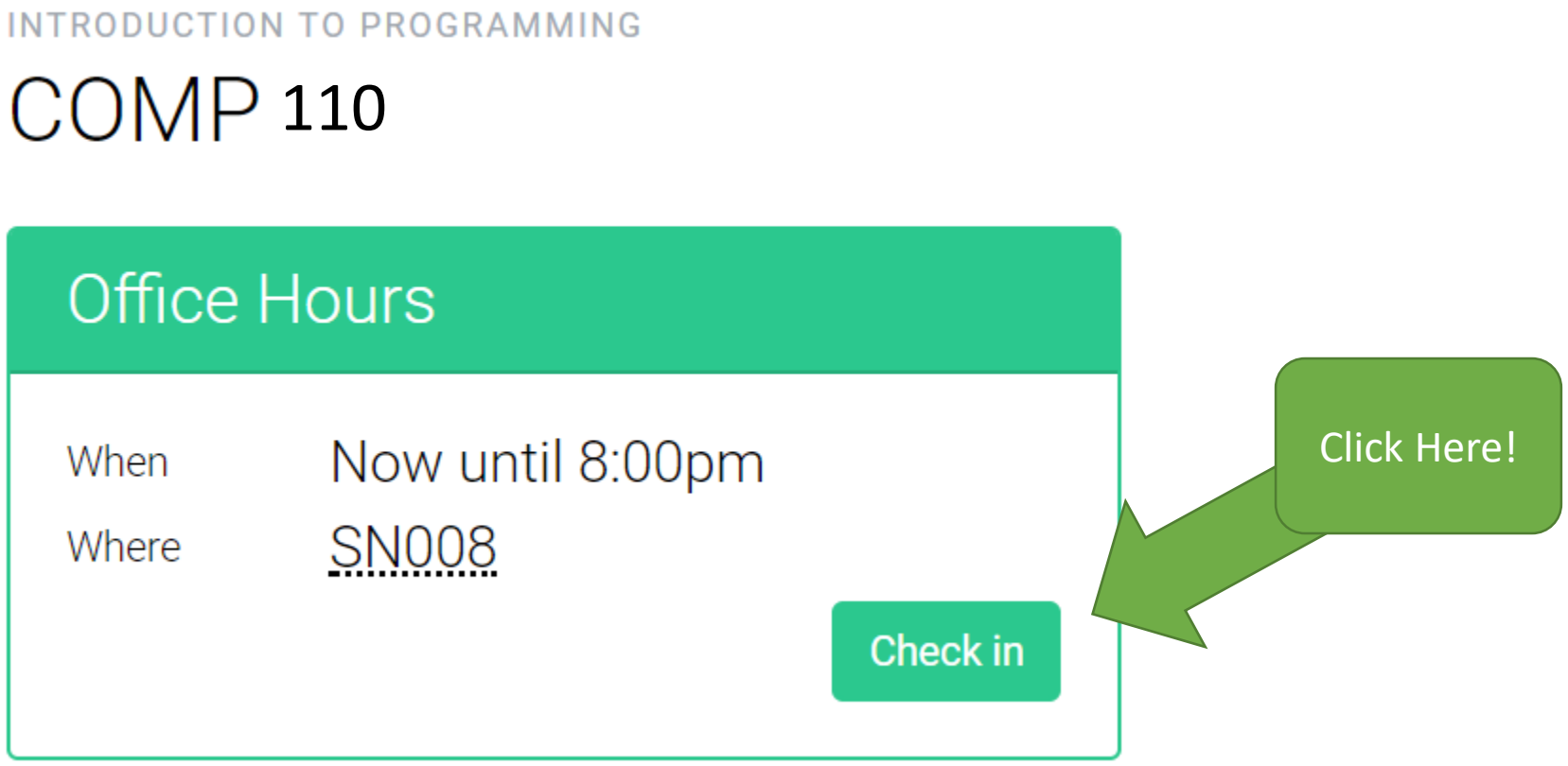
Registering for Course.Care

- If you have not registered for Course.Care yet, there's a link from the My110 Login page you should follow.

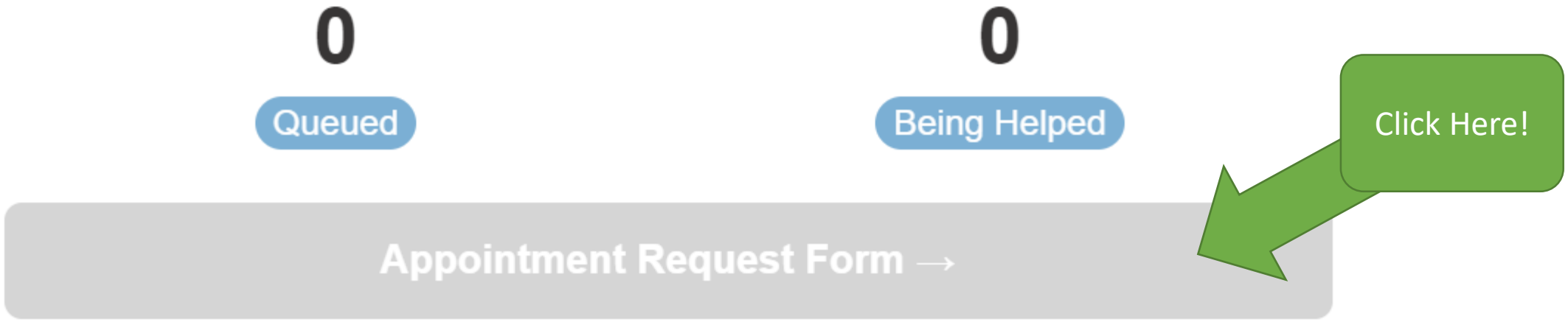
- Otherwise, the code for COMP110 on Course.Care is: **C1A606**

Office Hours Check-in Process

Click on "Get Help" on the course home page

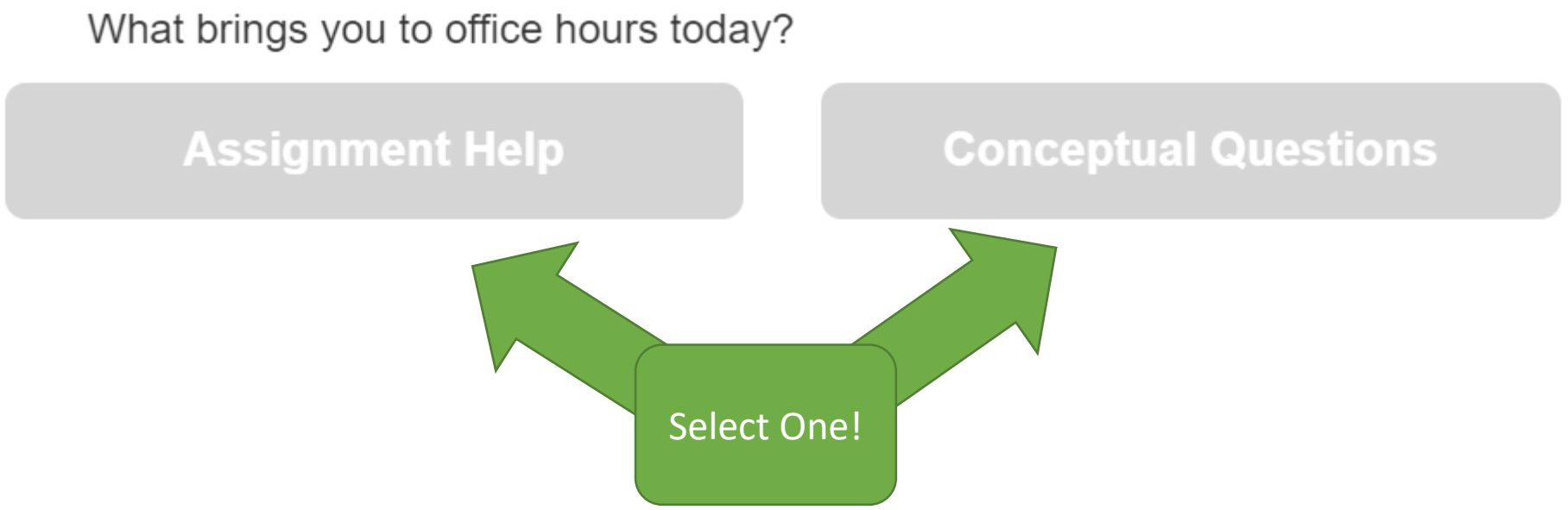


Office Hours Check-in Process

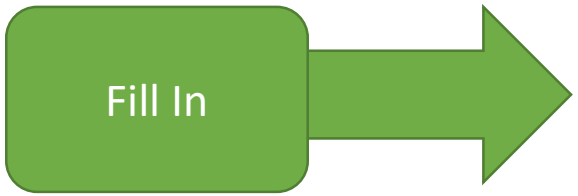


You can see how many people are currently waiting to be helped and currently being helped ahead of you.

Office Hours Check-in Process



Office Hours Check-in Process



IMPORTANT: You must demonstrate **effort and thought** in these fields. If you do not, the TAs are instructed to **cancel** your request so you can try again.

1. What section of the assignment do you need help with?

2. Describe in English what are you trying to express in code:

3. What concepts do you need to use to solve this problem?

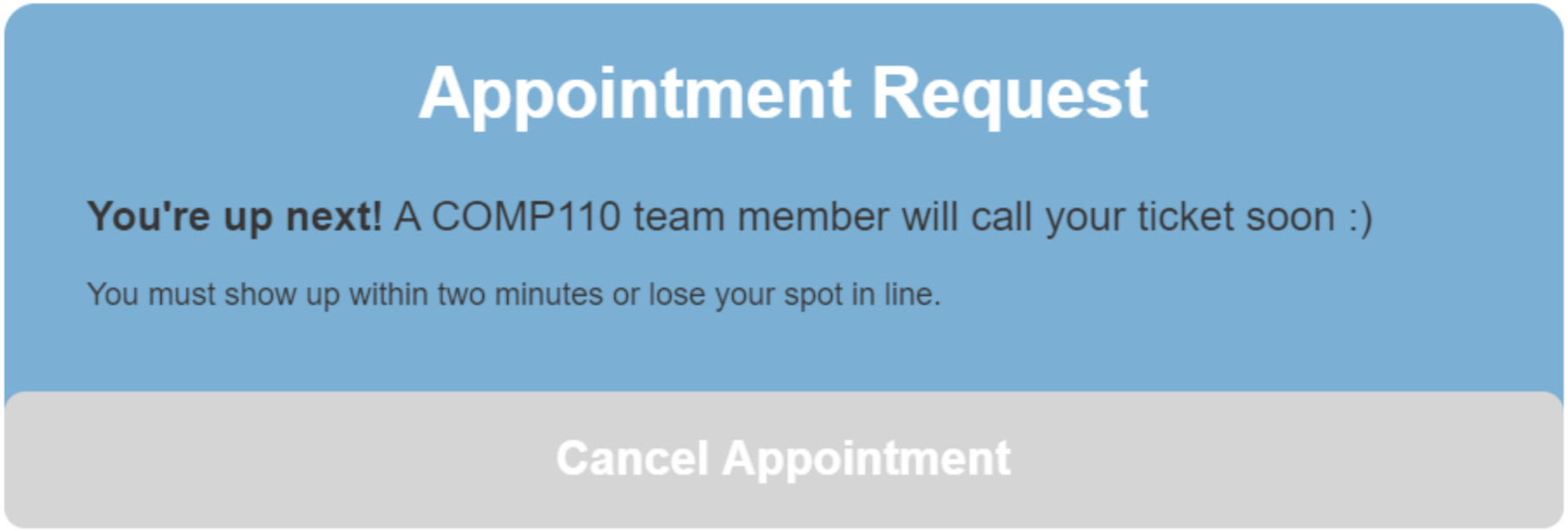
4. What have you tried? Why do you suspect it didn't work?

Disclaimer: Your help request will be cancelled if you cannot provide meaningful responses to each question.

Cancel

Get Help →

Office Hours Check-in Process



Office Hours Check-in Process

Kris is ready for you!

Come on in to SN008! You must show up within two minutes or lose your spot in line.

Cancel Appointment

Office Hours are Open Today!

Upcoming

Office Hours	Today from 11am to 6pm	SN008
Office Hours	Tomorrow from 11am to 6pm	SN008
Office Hours	Thursday from 11am to 6pm	SN008
Office Hours	Friday from 11am to 5pm	SN008

Challenge Question #0 - pollev.com/compunc

- Solve for yourself with paper/pencil then talk with your neighbors to see if you came to the same answer.
  - Don't use an interactive programming REPL!

- What does the following expression evaluate to?

```
1 + 2 ** 3 % 3 + "4"
```

Primitive Data Types Highlights

- *Every **value***, or piece of data, has a specific ***type***

- Primitive Types (simplest data types)

    1. **boolean**
        - Only two possible values: true or false
        - We will explore its operators next week

    2. **number**
        - Integers: -1, 2, 3
        - Decimals: 3.14
        - With `number` type data you have arithmetic operators like `**`, `*`, `/`, `%`, `+`, `-`

    3. **string**
        - Textual data surrounded by Quotes
        - Example: "Hello, World"
        - Strings can be text that resembles a number: "3.14"
        - You can join a `string` with another `string`/`number`/`boolean` using concatenation `+`

REPL - Playing with Data Types

- In Google Chrome, navigate to: **bit.ly/repl-110**
  - If you see an error about a favicon, press the "no smoking" symbol to clear the output

- Open the REPL:
  - Windows: Control + Shift + J
  - Mac: Command + Option + J

- Let's try the previous expression as well as some variations of it:
  - `1 + 2 ** 3 % 3 + "4"`
  - `1 + 2 ** 3 % 3 + 4`
  - `"1" + 2 ** 3 % 3 + "4"`

Questions on Primitive Data Types or Operations?

Challenge Question #1 - pollev.com/compunc

- What is the value of the variable **x** after this code listing completes?

```
let x = 110;
let y = 101;
let temp = x;
x = y;
y = temp;
```

Challenge Question #2 - pollev.com/compunc

- What is the value of the variable **e** after this code listing completes?

```
let a = "1";
let b = "2";
let c = "3";
let d = b + c + b;
let e = a + d + a;
```

Variables Highlights

- Variables allow your programs to store, change, and lookup values in memory.
  - You will use variables for all sorts of purposes in programs, like storing information about a the person using your app, counting the number of times something happens, the results of computations, and more!

- **Declare** and **Initialize**:

```
let <variableName> = <initial value>;
```

- **Assign** new values:

```
<variableName> = <new value>;
```

- **Access** a variable's value by its name:

```
<variableName>
```

REPL - Playing with Variables

- In Google Chrome, navigate to: **bit.ly/repl-110**
  - If you see an error about a favicon, press the "no smoking" symbol to clear the output

- Open the REPL:
  - Windows: Control + Shift + J
  - Mac: Command + Option + J

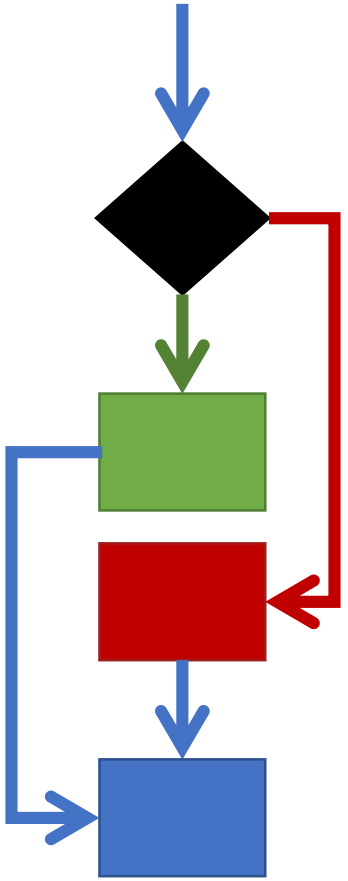
- Let's try the previous example:

```
let a = "1";
let b = "2";
let c = "3";
let d = b + c + b;
print(d)
let e = a + d + a;
print(e)
```

Questions on Variables?

Control Flow or "The Moves"

- The next unit will focus on **Control Flow**
- This refers to how the computer ***moves*** through processing your instructions
- We'll introduce these to you our own made up phrase: ***the moves***.



Think about a board game like **Monopoly**…

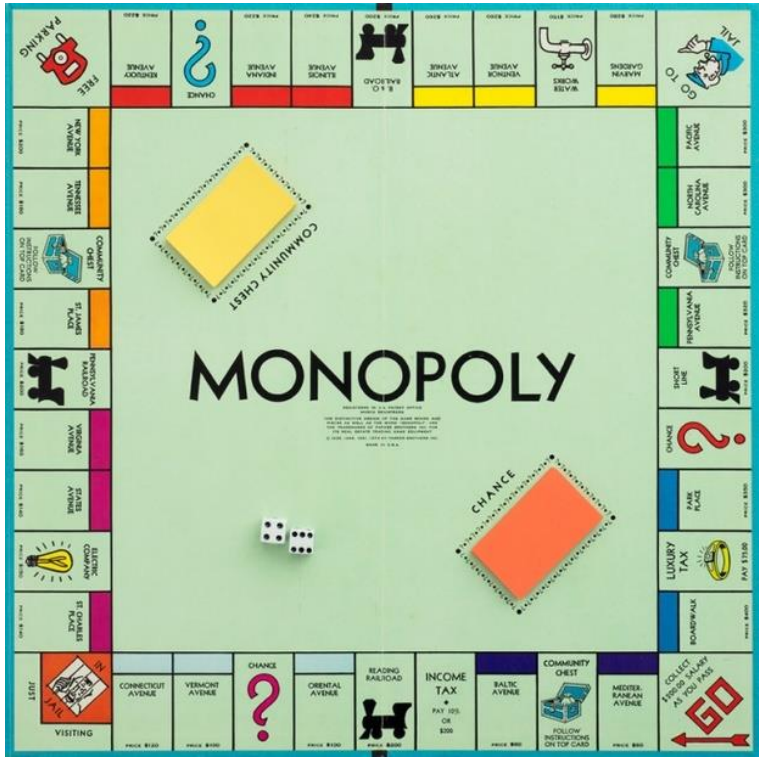
- Imagine you rolled a 1 every time.
  You ***move*** forward one space at a time.

- *Until* you reach a special square, i.e.:

  - **Chance Card Square**
    - A special move card can *jump* you around the board.

  - **Go to Jail Square**
    - Move all the way backwards to jail.

  - **How do you get out of jail?**
    - *<u>If</u>* you roll doubles, *<u>then</u>* you move forward by the number of spaces shown by the throw.



The Moves

- Like a board game, special moves exist in programs

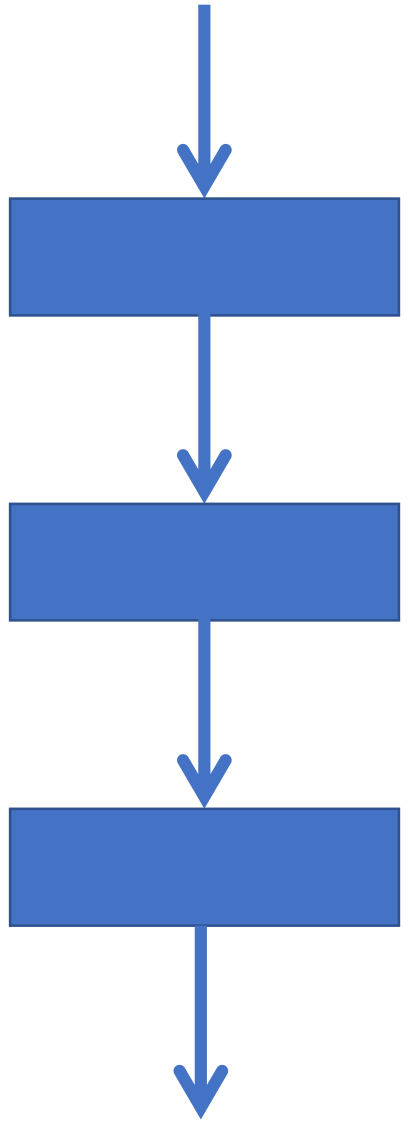
- In COMP110, there are ***3 moves*** you will learn in the upcoming unit:

  1. ~~Step Forward~~
  2. Conditionals (`if-then-else`)
  3. Loops (`while`)
  4. Function Calls

- In COMP401, you'll only learn ***1 other move:*** **"Exception"**

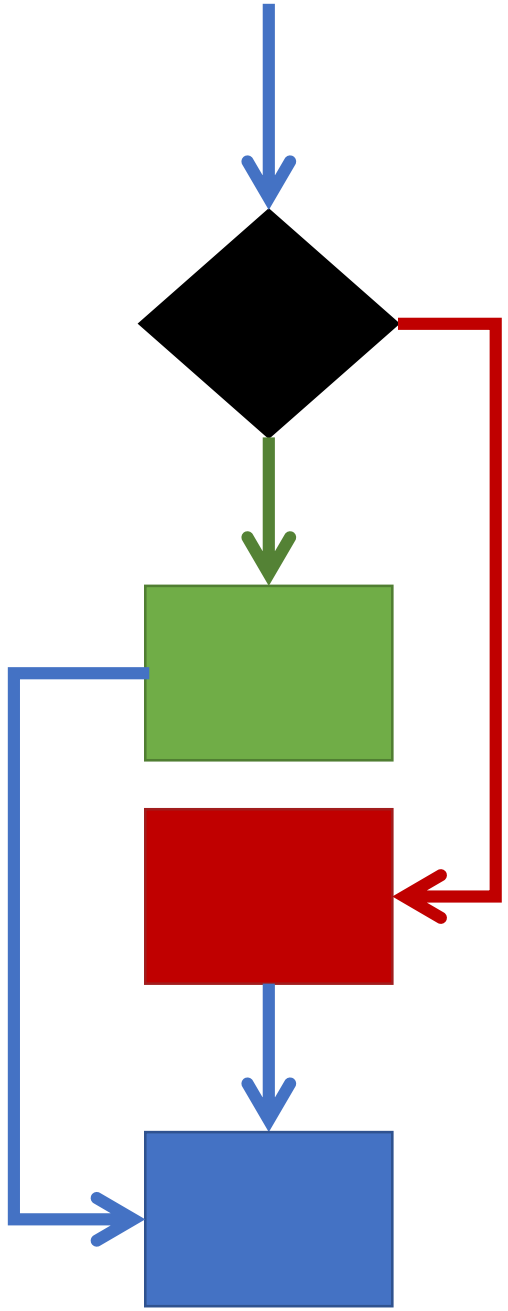
Move #1 – Step Forward

- You already know this one!
- The computer will process one line of code. Then it will process the next right after it. And the next…
- ***Until*** it encounters a *special* move or the end of the program.



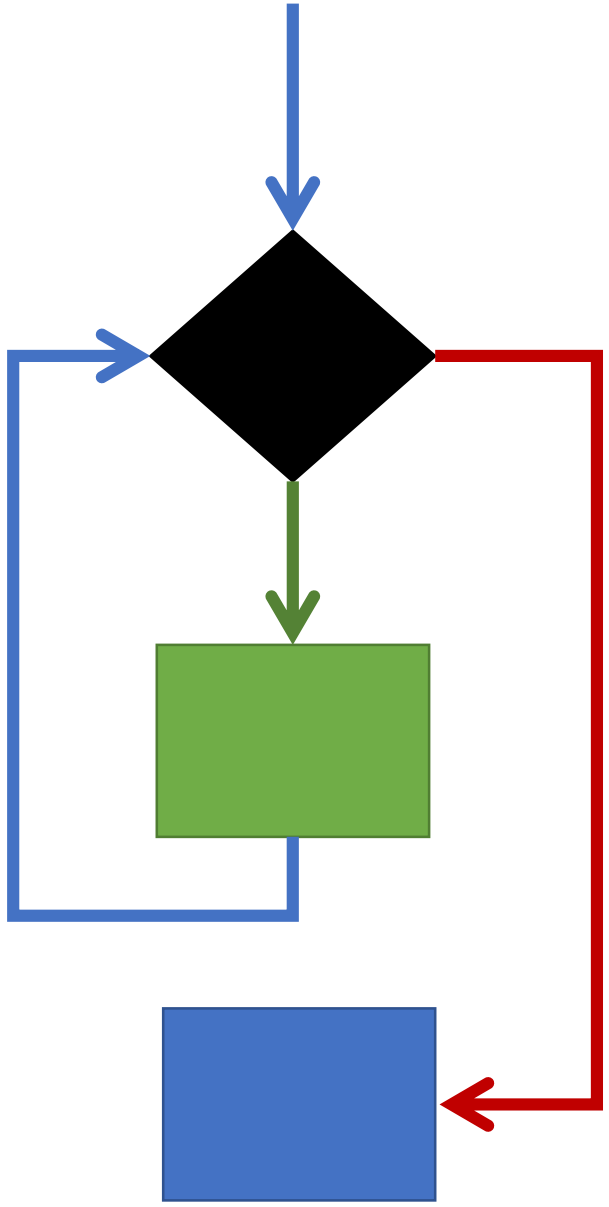
Move #2 – Conditionals

- Based on some test (**`boolean`**!)…
- **if** the test is `true`
  - ***then*** the computer will continue to the next statement.
- **else** it will jump to a specific statement further down in your program
  - more on the exact rules next lecture



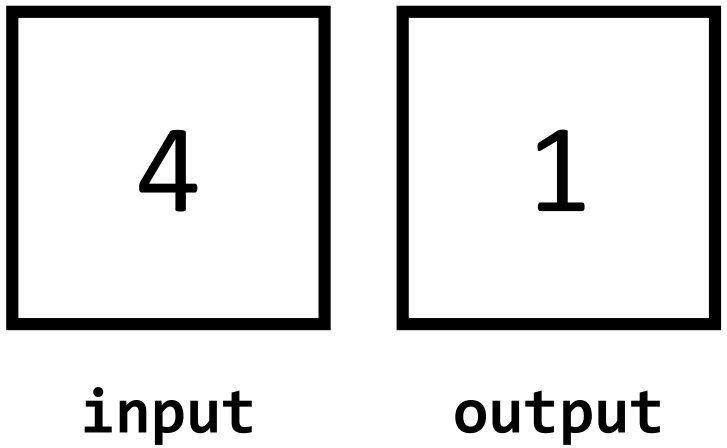
Move #3 – Loops

- Also based on some test (**`boolean`**!)…
- **if** the test is `true`
  - ***then*** the computer will continue to the next line.
  - at a specific point, the loop will ***jump back up*** to the test and run the test again.
- **else** it will jump to a specific line further down in your code
- You'll go loopy next week.



Practice Puzzle with a Conditional and Loops

- On a piece of paper, make two boxes.
  - If you're writing in pen, make them big enough to scratch out their contents with room to write more.
- Label one "input", write 4 in it.
- Label the other "output", write 1 in it.



Instructions

1. If **input is at least 2**,
   then continue to **INSTRUCTION 2**,
   otherwise, **JUMP** to **INSTRUCTION 5**

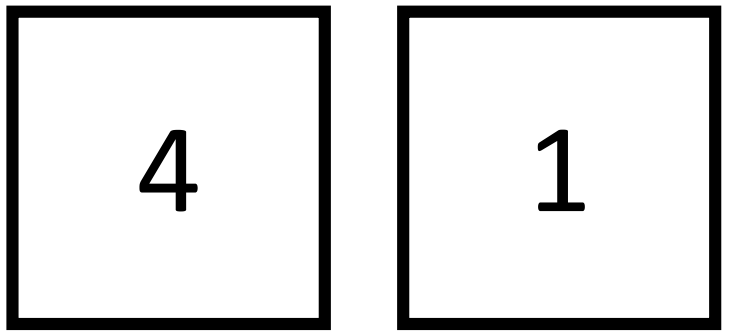
2. output = input * output;

3. input = input - 1;

4. JUMP to **INSTRUCTION 1**

5. The answer is stored in **output**.

6. EXIT



Instructions

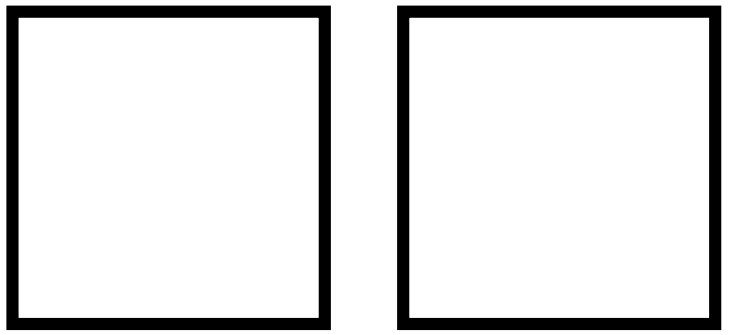
1. **IF `input` is at least 2**,
   THEN continue to **INSTRUCTION 2**,
   ELSE JUMP to **INSTRUCTION 5**

2. `output = input * output;`

3. `input = input - 1;`

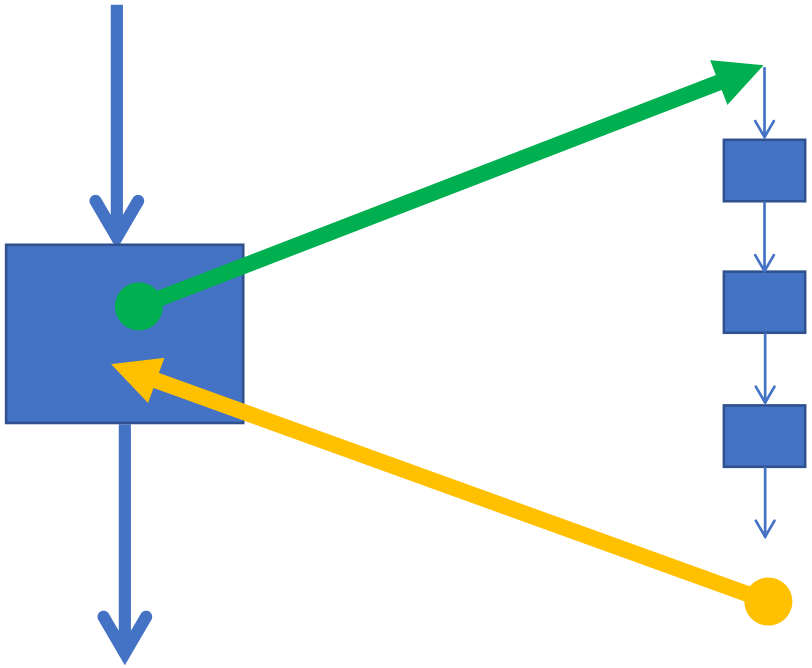
4. **JUMP** to **INSTRUCTION 1**

5. The answer is stored in **`output`**.
6. **EXIT**



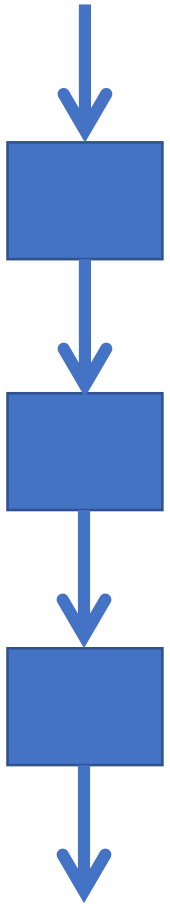
Move #4 – Function Calls

- The **function call** move is beautiful and magical. It's the power move.
- The computer *drops a bookmark* where the function call occurs and *jumps* into the function... *magic happens...*
- ...the computer then ***returns*** right to *the bookmark it dropped*, often with some data and the program continues on, business as usual.
- You've *already* used some function calls!
  - `print("hello, world");`

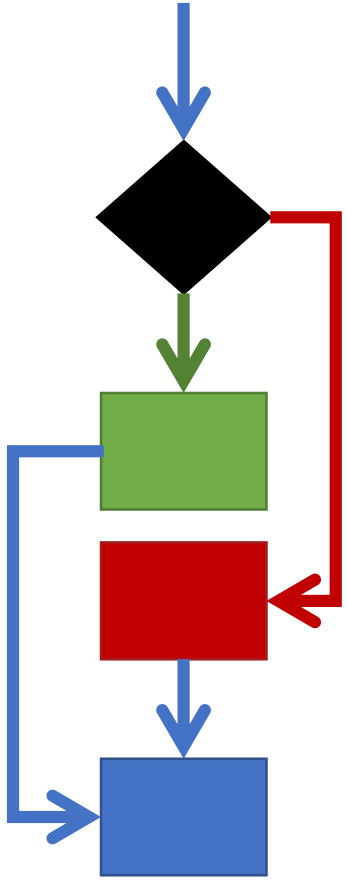


The Moves

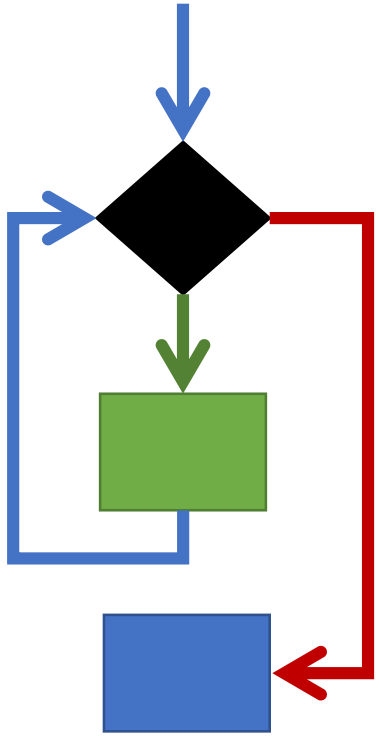
Next Line



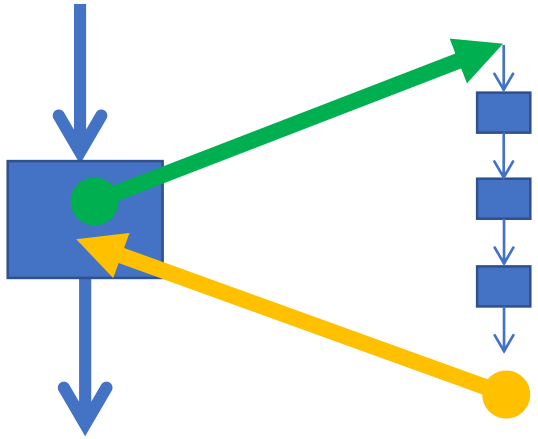
Conditionals



Loops



Calls (to Functions)



// Code Comments (1/2)

- We can add notes for ourselves in code using **comments**
- The computer ignores comments when running your program
- Single line comments begin with a //

```
// This is a single line comment
```

- Multi-line comments are surrounded by an opening **/*** and closing ***/**

```
/* This
   is
   a multi-line comment
*/
```

- Comments are also useful for ignoring code you've written without deleting it.

// Code Comments (2/2)

- In the early days of programming, we recommend writing comments liberally in your code to explaining what your program is doing in English.

- Comments are *free* so use them liberally.

- Comments are the easiest way to "take notes" when we are working on examples in lecture.

Statements (1/2)

- **Statements** are equivalent to an English sentence

- Statements *usually* end with a semi-colon ;
  - Like sentences end with a period!

- Each statement is an instruction you are giving to the computer.

```
print("hello, world");
```


That's a **statement**!

"Print 'hello, world' to the screen."

Statements (2/2)

- In a stored program, the computer will not carry out our instructions until we **run** the code
  - When you *write* programs, it is like you are writing a recipe down
  - When you *run* a programs, the computer is like the chef *following* your recipe
- Before the first statement runs, your program is a barren, empty world
  - *Your* code builds up its own little world piece-by-piece

```
print("hello, world");
```


That's a **statement**!

"Print 'hello, world' to the screen."