

# Unit 01: Control Flow Practice

Lecture 05

Go to [poll.unc.edu](https://poll.unc.edu)

Sign-in via this website then go to [pollev.com/compunc](https://pollev.com/compunc)

VSCode: Open Project -> View Terminal -> `npm run pull` -> `npm start`

# Announcements

- Personal TA Teams Assigned – Your first point of contact for questions.
  - Technical questions or questions that would require an attachment or copying and pasting code will be redirected to Office Hours
  - Go to:
    1. My110
    2. My Team
    3. Message My Team

# Office Hours Process Reminder

- Office Hours Process

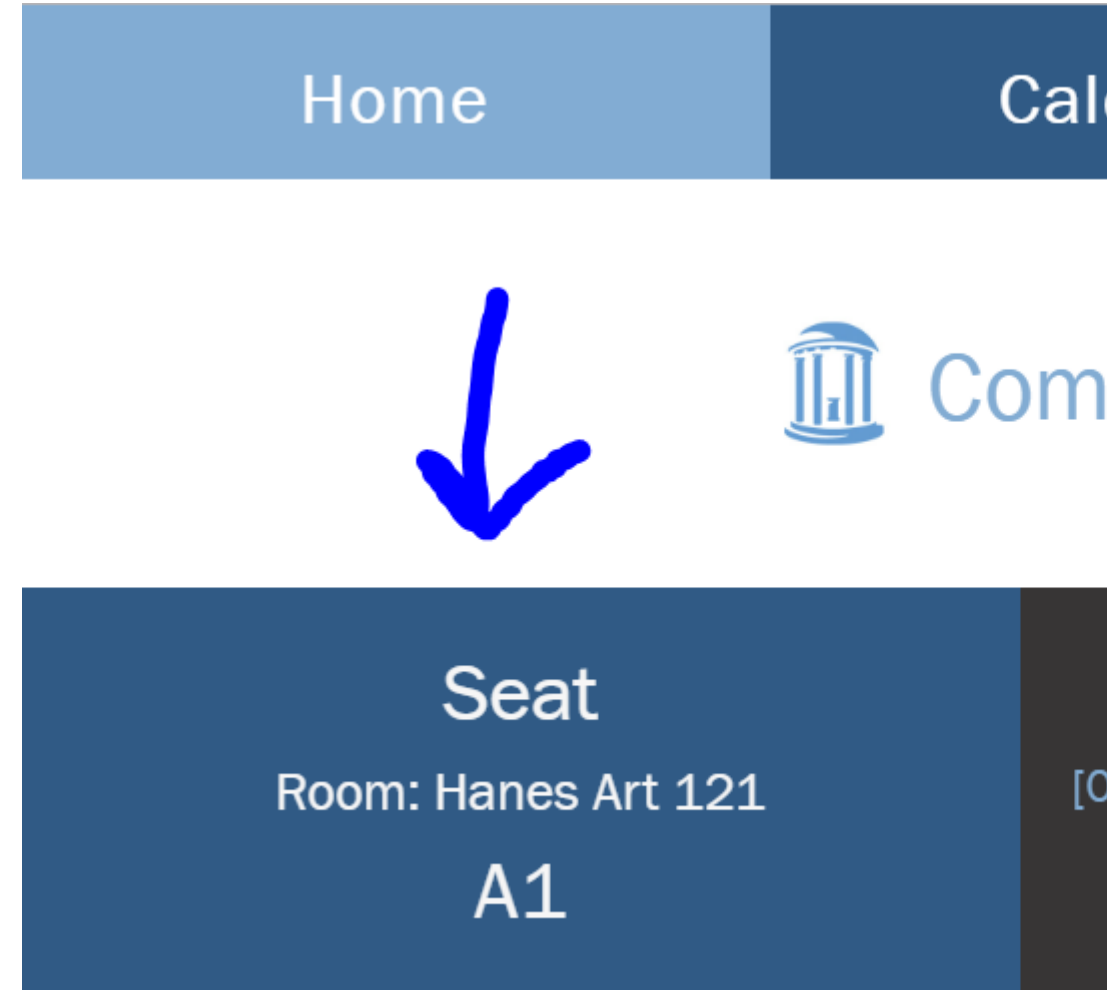
1. Wait in the lobby, not in the office hours room.
2. Submit your request for help on Course.Care
3. Keep an eye on your ticket and come in when it's called. If you do not come in within 2 minutes of it being called, we will assume you are cancelling.
4. An individual appointment is limited to 15 minutes / one specific question.
5. After an appointment, return to the lobby.

# Worksheet Submission Process

- Plan to be ready to submit 6 hours before the deadline.
  - The first time you submit a handwritten assignment can take up to an hour to get the app correctly installed, scanned, emailed to yourself, double checked on your computer, and submitted.
- Attempting to email a worksheet will not count as a submission. To submit your worksheet you must upload through your Gradescope account.
- To receive full credit, be sure you:
  1. Print and hand write your worksheet responses.
  2. Scan (on phone) your worksheet with all pages in the correct ORDER and vertical ORIENTATION.
  3. Open your PDF on your computer to double check correctness before submitting.
- Failure to properly submit will result in point deductions.
- For detailed instructions on how to scan from your phone read here:  
<https://comp110.com/topics/getting-started/gradescope-submissions>

# Preparing for Quiz 1

- **Know your assigned seat.** After you login to My110 on COMP110.com, go back to the Home page to see your assigned (shown right).
  - Attempting to sit in a seat other than the one assigned will result in a score of 0 for that quiz.
  - We will randomize seats through the semester.
- Practice Material:
  - Submitted Worksheet
  - Practice Worksheets
  - Video Slides
  - Practice Problems in Class
  - Relevant Topics Pages on Comp110.com
- **Review Session – 5pm Monday in FB009**



**Challenge Questions:** There are three variable initialization statements below. In each blank (three subsequent poller questions), write a valid function call to an appropriate function defined below. You may use any literal values you'd like as arguments.

PollEv.com/compunc

```
export let main = async () => {  
  let x: number = _____;  
  let y: boolean = _____;  
  let z: string = _____;  
};  
  
let a = (): boolean => {  
  return random(0, 1) === 0;  
};  
  
let b = (s: string): number => {  
  return s.length;  
};  
  
let c = (n: number, s: string): string => {  
  return s + ":" + n;  
};
```

```

3  let main = async () => {
4      let x = b(a(2));
5      print(x);
6  };
7
8  let a = (x: number): number => {
9      print("a");
10     let y = b(2 * x);
11     return y;
12 };
13
14 let b = (x: number): number => {
15     print("b");
16     let y = 2 * x;
17     return y;
18 };
19
20 main();

```

**CQ 4. Given the code left, draw an environment diagram.** Check-in with the printed output when complete on [PolleEv.com/compunc](https://PolleEv.com/compunc)

```
3  let main = async () => {
4      let x = b(a(2));
5      print(x);
6  };
7
8  let a = (x: number): number => {
9      print("a");
10     let y = b(2 * x);
11     return y;
12 };
13
14 let b = (x: number): number => {
15     print("b");
16     let y = 2 * x;
17     return y;
18 };
19
20 main();
```



# Environment Diagrams (v1)

## Function Call

1. Evaluate function call argument values
2. Establish new frame on call stack
  - i. Add name of function
  - ii. Add RA (Return Address line #)
  - iii. Copy arguments to parameters in frame
3. Jump to first line of function definition

## Function Return Statement

1. Evaluate returned expression
  - Add RV (Return Value) in current stack frame
2. Jump back to function caller
  - i. Line is in RA (Return Address)
  - ii. The function call evaluates to last frame's RV

**Current Frame:** The most recently added frame that has not returned.  
(*No RV!*)

**Name Resolution:** Look for name in the current frame. (*For now.*)

**Variable Declaration:** Enter name and space for variable to current frame.

**Variable Assignment:** Find variable location via name resolution, copy assigned value to it.

**Variable Access:** Find variable location via name resolution, use value currently assigned to it.

# Increment Operator (++)

- Adding one to a variable is so common when looping there is a special operator for it...
- We often write: **`i = i + 1;`**
- We can instead write: **`i++;`**
- These two statements have the exact same impact of incrementing **`i`**'s value by **`1`**.

# Decrement Operator (--)

- Subtracting one from a variable is *also* so common, there is a special operator for it...
- We often write: **`i = i - 1;`**
- We can instead write: **`i--;`**
- These two statements have the exact same impact of incrementing **`i`**'s value by **`1`**.

# The TypeScript REPL

- For all lectures, worksheets, and problem sets, we grade based on the TypeScript language's rules.
- Your web browser's REPL follows the JavaScript language's rules.
  - JavaScript is TypeScript *without any data typing rules*
  - In JavaScript you can do **fully crazy** things like multiply numbers with booleans
    - What does it even mean? No point in knowing... it's never a good idea and is a bug or bad code.
  - We started in the JavaScript REPL because you didn't need VSCode installed to use it
- For a TypeScript REPL – open a new terminal in VSCode and run the command:  
**npm run repl**

```

3 export let main = async () => {
4     print(mystery(11, 4));
5     print(mystery(13, 4));
6 };
7
8 let mystery = (n: number, d: number): number => {
9     let tries = 0;
10    while (tries < 3) {
11        if (n % d === 0) {
12            return n;
13        } else {
14            n++;
15            tries++;
16        }
17    }
18    return -1;
19 };
20
21 main();

```

**CQ 5. Given the code left, draw an environment diagram.** Check-in with the printed output when complete on [PollEv.com/compunc](https://pollev.com/compunc)

```
3 export let main = async () => {
4     print(mystery(11, 4));
5     print(mystery(13, 4));
6 };
7
8 let mystery = (n: number, d: number): number => {
9     let tries = 0;
10    while (tries < 3) {
11        if (n % d === 0) {
12            return n;
13        } else {
14            n++;
15            tries++;
16        }
17    }
18    return -1;
19 };
20
21 main();
```

# Challenge Question #6: Code writing

- Write a function named **prod** that is given two numbers and returns the product of every number between those two numbers, inclusive of both numbers.
- For example: calling `prod(2, 4)` should return the result of  $2*3*4$ .
- Check-in on **[PollEv.com/compunc](https://pollev.com/compunc)** when you have a solution.

# Challenge Question #7 - pollev.com/compunc

- What is the result of calling: **cute(3)**

```
let cute = (force: number): string => {  
  let s = "";  
  let i = 1;  
  while (i < force) {  
    s = s + "h";  
    let h = 0;  
    while (h < i) {  
      s = s + "e";  
      h++;  
    }  
    i++;  
  }  
  return s;  
};
```



# Notes on Nested Loops

- **General Rule:** When the closing curly brace of a loop is encountered, the loop jumps back to the start of **its matching condition**.
- An inner loop will jump back up to the inner loop's condition and an outer loop will jump back up to the outer loop's condition.
- Thus, an inner loop must complete all of its **iterations** for every individual iteration of an outer loop.