launch_code

How to Succeed at LaunchCode
Chapter 4: Data and Variables

WOMEN+ ▸ UNIT 1 ▸ CLASS 1 ▸ MARCH 22, 2023

Class Agenda

Announcements

How to Succeed at LaunchCode

Lecture: Data and Variables

Intro to Graded Assignment #1

Meet Your TAs

Studio: Chapter 5 - Goals & Growth Mindset

Announcements

Graded Assignment Deadlines

Part 1 of GA#1 is due **4/5**
Remember on graded assignments you can show and discuss your code ONLY with your TAs.

Part 2 of GA#1 is due on **4/19**

Enrollment deadline is 4/26
Graded assignment #1 must be fully complete and correct for you to continue in Women+.

How to Succeed at LaunchCode

launch_code

How to Succeed at LaunchCode

Getting the Most out of Women+

Course Tools & Resources

Expectations

Advice & Encouragement

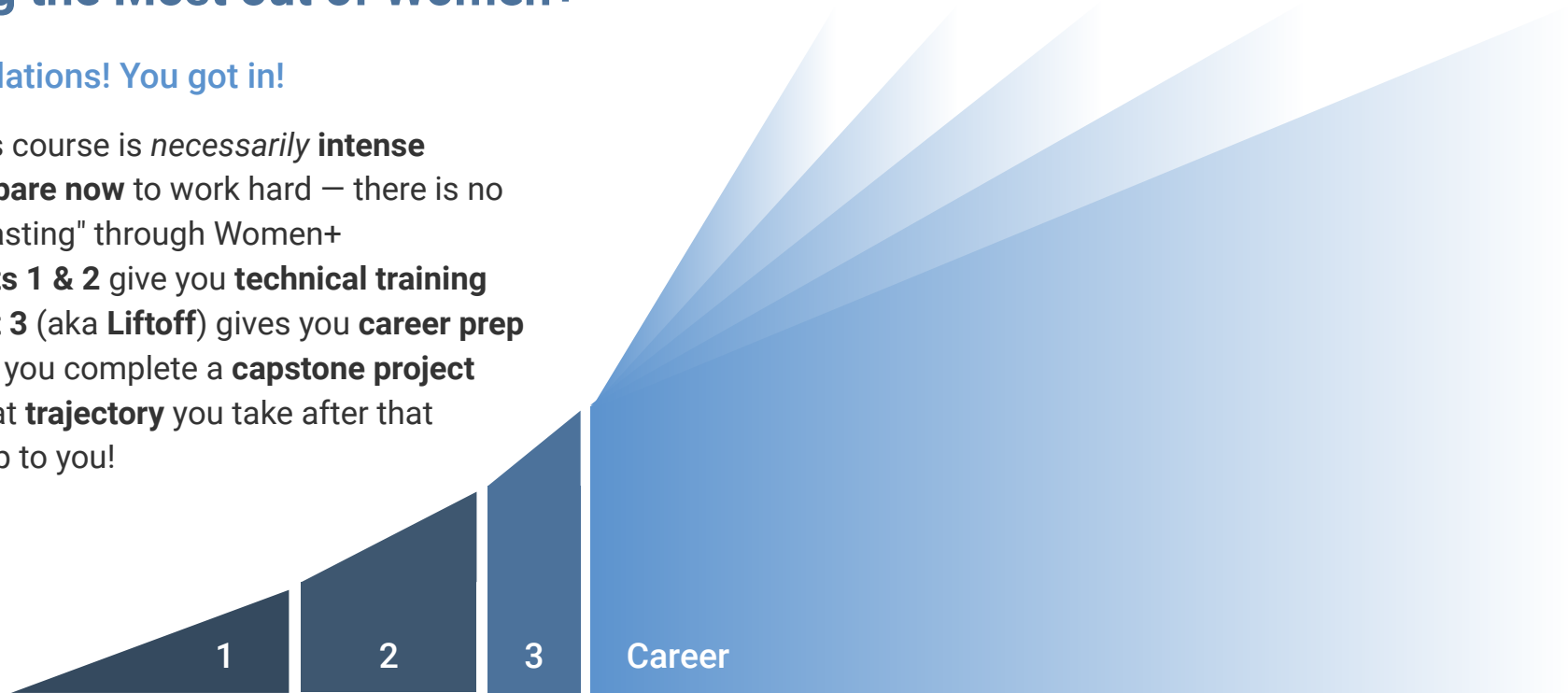
launch_code

How to Succeed at LaunchCode

Getting the Most out of Women+

Congratulations! You got in!

- This course is *necessarily* **intense**
- **Prepare now** to work hard — there is no "coasting" through Women+
- **Units 1 & 2** give you **technical training**
- **Unit 3** (aka **Liftoff**) gives you **career prep** and you complete a **capstone project**
- What **trajectory** you take after that is up to you!



How to Succeed at LaunchCode

Course Tools & Resources

Canvas

- Follow syllabus
- Complete quizzes
- Log class attendance
- Turn in prep exercises, studios, and graded assignments

Replit (& GitHub eventually)

- Write code
- Turn in links on Canvas

LC Course Book

- Assigned reading
- Studio instructions
- Graded assignment instructions
- Helpful resources

Google

Blogs, YouTube videos, documentation, forums... be resourceful!

Slack

- Announcements
- Lecture questions
- Group channels
- DM with TAs, peers
- Phone app
- Desktop app

Carrie's Practice Exercises

https://tinyurl.com/y3bn6st4

How to Succeed at LaunchCode

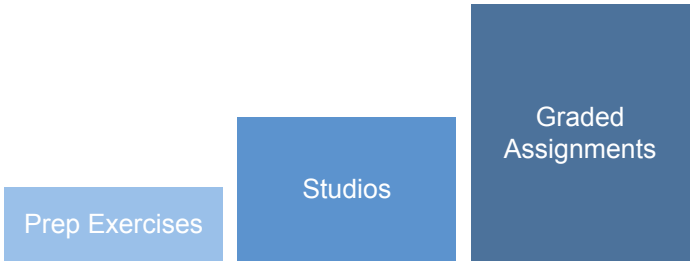
Expectations

In Class

- Stay **engaged** in lecture
- Use the Slack channel **#questions**
- If your instructor asks **questions**, feel free to speak up and answer!
- Otherwise, keep yourself **muted** when others are speaking
- **Collaborate** during studios — you learn **faster** & **better** when **working together** to solve the problem

Outside of Class

- Check **Slack** regularly
- Spend **~15 hours per week** on prep work & graded assignments
- Complete all prep work **ahead** of lecture (lecture moves quickly)
- **Turn in** your work for **TA feedback** and **communicate** with TAs
- Increasing levels of **difficulty**:



How to Succeed at LaunchCode

Advice & Encouragement

Figure Out What Works For You

- Be **strategic** about setting time aside
- Maximize your **home learning environment**
- Find **additional resources** for learning - videos, blogs, tutorials, practice problems, etc.

Be Kind to Yourself

- It will get **hard** — embrace the discomfort
- Everything you will feel is **normal**
- Keep your **end goals** in mind
- **ENJOY** the process!

Have an Extended Support System

- **Talk** to friends & family
- **Protect** your relationships
- Be **honest** with them about what you need from each other

Be There for Each Other

- LaunchCode is a **community**
- **No one** is alone
- Build each other up
- Learn from each other

- Decide TODAY to commit yourself to making time for this course, and you will succeed!

launch_code

Data and Variables

THE BASICS OF STORING INFORMATION

launch_code

Data and Variables

- Data Types
- Printing to the Console
- Detecting Types & Converting Data
- The Variable
- The Constant
- What's in a Name?
- Evaluating Expressions
- Operators
- Getting User Input

launch_code

Data and Variables

Data Types

String

- Words, letters, digits, special characters
- Must be in **single** or **double** quotes
- Quotes within quotes (as long as they're not the same kind)
- Can be very **short** or very **long**
- Can be **empty** (but still a string!)

```
"Welcome to LaunchCode!"

'Learning JavaScript is fun.'

'She said, "Let's do it!"'
```

```
19
20  /* THE STRING */
21
22  // TODO: Change the values for each string
23  // Remember that quotes within a quote require using single quotes
24  let emptyString = ""; // no characters
25  let space = " "; // space only
26  let zeeOrZed = "z"; // single letter
27  let farmBoy = "Westley"; // single word
28  let manInBlack = "Dread Pirate Roberts"; // multiple words
29  let memorableQuote = 'Inigo said, "Hello. My name is Inigo Montoya.
    You killed my father. Prepare to die."'; // quotes inside quotes
30  let rottenTomatoesScore = "97"; // numbers, but as a string
31
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Data Types

Number

- **Positive** or **negative**
- **Integers** (whole numbers)
- **Floats** (floating points/decimals)
- **Doubles** (longer decimals)
- In JavaScript, they're all the same type — `number`!
- NO quotes — otherwise it's a `String`
- NO commas

```
42

3.14

6.02214076

93000000

-10
```

```
33  /* THE NUMBER */
34
35  // TODO: Change the values for each number
36  let numberOfOutlaws = 3; // integer
37  let mostlyDead = 0.985; // decimal
38  let energyLevelAfterRevival = -86; // negative
39
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Printing to the Console

What is the Console?

- A **console** is a **command line interface (CLI)** where you can execute commands
- You can **interact** with it directly, AND
- You can **print** to the console from your code
- Use `console.log(yourContent)` in your code to print something to the **console**

```
console.log("JavaScript") ⇒ "JavaScript"

console.log(365) ⇒ 365
```

Data and Variables

Printing to the Console

Special Characters

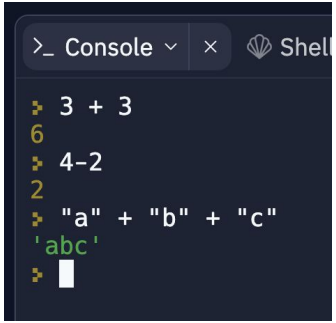
- An **escape character** is a special character that begins with a **backslash**, `\`
- They can be used as part of a **string**
- The **newline** character `\n` will add in a **line break**
- The **tab** character `\t` will add in a few characters of **blank space**

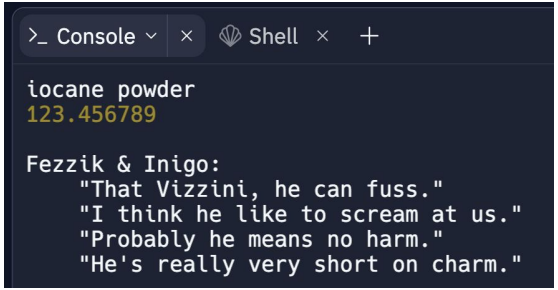
```
console.log("A\nB\nC") ⇒ A
                         B
                         C
```

```
console.log("A\tB\tC") ⇒ A    B    C
```

```
40
41  /**** PRINTING TO THE CONSOLE ****/
42
43  // Experiment with using the console directly.
44
45  // TODO: print a string, then print a number
46  console.log("iocane powder");
47  console.log(123.456789);
48  // TODO: Hit the Run button (big green arrow) to see the output
49
```

```
50
51  /* FORMATTING */
52
53  // TODO: Add newline and tag characters to format an indented list
54  console.log('\nFezzik & Inigo:\n\t"That Vizzini, he can fuss." \n\t"I
    think he like to scream at us." \n\t"Probably he means no harm."
    \n\t"He\'s really very short on charm." \n');
55
```





Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▸ Printing to the Console

Data and Variables

Detecting Types & Converting Data

Checking the Type of Data

- Use the **keyword** `typeof` to find out what **data type** something is
- Combined with `console.log()` you will see the **result** in the **console** when you **run** the program

keyword data

```
console.log(typeof "St. Louis, MO") ⇒ "string"

console.log(typeof 20) ⇒ "number"
```

```
58
59  /* DETECTING TYPES */
60
61  // TODO: Print the type of a string of characters to the console
62  console.log(typeof "R.O.U.S.");
63
64  // TODO: Print the type of a negative number to the console
65  console.log(typeof -1);
66
```

```
string
number
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▸ Detecting Types

Data and Variables

Detecting Types & Converting Data

Changing the Type of Your Variable

In some cases, a **data type** can be changed

- The **function** `String()` will change data to the `String` type
- The **function** `Number()` will attempt to change data to the `number` type
- If a string **cannot** be converted to a number, the value will be `NaN` — "not a number"

```
String(3.14) ⇒ "3.14"

Number("3.14") ⇒ 3.14

Number("pi") ⇒ NaN
```

```
69
70  /* CONVERTING DATA */
71
72  // TODO: Convert 4 to a string and print the result, then verify the type
73  console.log(String(4));
74  console.log(typeof String(4));
75
76  // TODO: Convert "10000" to a number and print the result, then check the type
77  console.log(Number("10000"));
78  console.log(typeof Number("10000"));
79
80  // TODO: Try to convert "Humperdinck" to a number and print the result, then check the type
81  console.log(Number("Humperdinck"));
82  console.log(typeof Number("Humperdinck")); // turns out NaN has a type!
83
```

```
4
string
10000
number
NaN
number
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▸ Type Conversion

Data and Variables

The Variable

What is a Variable?

A **variable** stores **data** so you can refer to it later

- Declared with the **keyword** `let`
- The **variable name** should be meaningful
- Use **camelCase** as the **naming convention**
- **Value** is **assigned** with the **equals** operator
- In older examples of code you might see the keyword `var` used instead — that's outmoded

keyword variable name

```
let myClass = "WebDev";
```

assignment value

Data and Variables

The Variable

Declaration vs Initialization

- You **declare** a variable when you use the **keyword** `let` and give it a **name**
- You **initialize** a variable when you **assign** a **value**
- It is possible to **declare without initializing** — value can be assigned later
- Use the keyword `let` only **once** when declaring

keyword

```
let newLanguage;
```

variable name

```
newLanguage = "JavaScript";
```

assignment value

Data and Variables

The Variable

Reassigning Variable Values

- Once you have **initialized** a variable by giving it its first value, you can **reassign** it later
- Remember: you **don't** use the `let` keyword again

```
console.log(newLanguage); ⇒ "JavaScript"

newLanguage = "Python"

console.log(newLanguage); ⇒ "Python"

newLanguage = "C#"

console.log(newLanguage); ⇒ "C#"
```

```
/**** THE VARIABLE ****/

// TODO: Declare & initialize a string variable, then print it
let vizziniPhrase = "Inconceivable!";
console.log(vizziniPhrase);

// TODO: Declare & initialize a number variable, then print it
let numberOfTimesSpoken = 3;
console.log(numberOfTimesSpoken);

// TODO: Give the newest variable a new value, then print it
numberOfTimesSpoken = 5;
console.log(numberOfTimesSpoken);

// TODO: Declare another variable and then print before initializing:
let inigoResponse;
console.log(inigoResponse);

// TODO: Initialize the variable, then print it again:
inigoResponse = "You keep using that word. I do not think it means what you think it means.";
console.log(inigoResponse);
```

```
Inconceivable!
3
5
undefined
You keep using that word. I do not think
it means what you think it means.
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▸ Declaring & Initializing Variables

Data and Variables

The Constant

- Stores information, like any variable
- Value is **protected** and cannot be changed
- Declared with the keyword `const`
- Sometimes the convention **SCREAMING_SNAKE_CASE** is used instead of **camelCase**.
- Useful in situations where you want to ensure the value is **never altered**

keyword variable name

```
const EULERS_NUMBER = 2.71828
```

assignment value

```
105
106  /**** THE CONSTANT ****/
107
108  // TODO: Declare and initialize a constant
109  // Use the allcaps underscore naming convention
110  const MOVIE_TITLE = "The Princess Bride";
111
112  // TODO: Try to assign a new value to the constant.
113  // What happens when you run the program?
114  MOVIE_TITLE = "Some Other Title";
115
```

```
TypeError: Assignment to constant variable.
    at Object.<anonymous> (/home/runner/Class01-LectureExamples-Solution/index.js:114:13)
    at Module._compile (node:internal/modules/cjs/loader:1159:14)
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▸ The Constant

Data and Variables

What's in a Name?

The Importance of Good Variable Naming

- Be **descriptive**
- **Readability** is more important than **length**
- Avoid **confusion** with other variables
- Make it **obvious**...
  - what **specific information** it holds
  - what **data type** it is
- Don't use **keywords** like `const` as variable names — they are **reserved** in JavaScript

~~`icflvr`~~ `iceCreamFlavor`

~~`scoops`~~ `numberOfScoops`

~~`type`~~ `coneType`

```
119
120  /**** WHAT'S IN A NAME? *****/
121
122  // These are not great choices for variable names.
123  let book = 'The Princess Bride: S. Morgenstern\'s Classic Tale of True Love
     and High Adventure, The "Good Parts" Version';
124  let type = "Adventure";
125  let kind = "hardcover";
126  let books = 10;
127
128  // TODO: Declare new variables for the values above
129  // They should be descriptive, specific and obvious about data type
130  let bookTitle = 'The Princess Bride: S. Morgenstern\'s Classic Tale of True
     Love and High Adventure, The "Good Parts" Version';
131  let bookGenre = "Adventure";
132  let bookFormat = "hardcover";
133  let numCopiesAvailable = 10;
134
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

EXAMPLE ▶ Choosing Good Variable Names

Data and Variables

Evaluating Expressions

What Is an Expression?

An **expression** lets you use code to **solve a problem**

- **Combines** values, variables, operators, and more
- Can be as **simple** as getting the value of a variable
- Can be **complex** with multiple operations

The expression is **evaluated**

The resulting value is **returned**

```
someString

1 + 2

(3 * someNumber) + 14

"my name is " + myName
```

Data and Variables

Evaluating Expressions

Using an Expression

- The expression can be **evaluated in place**, OR
- You can **store** the **returned value** in a **variable** to use elsewhere
- Storing it in a variable is a good idea when you need to use that value **more than once**

```
console.log(6 * 8);


let product = 6 * 8;

let total = 10 + product;

console.log(product);

console.log(total);
```

Data and Variables

Operators

Basic Arithmetic...

- `+` Addition
- `-` Subtraction
- `*` Multiplication
- `/` Division

Exponentiation

- `**` "to the power of"
- `**2` "squared"
- `**3` "cubed"

Increment & Decrement

- `++` Increment (increase by 1)
- `--` Decrement (decrease by 1)

```
132
133  /**** EVALUATING EXPRESSIONS WITH OPERATORS ****/
134
135  /* Arithmetic */
136
137  let a = 7;
138  let b = 2;
139  // TODO: Print the difference between a and b using the arithmetic
     operator for subtraction
140  console.log(a - b);
141
142  let m = 7
143  // TODO: Print m to the 4th power using the exponentiation operator
144  console.log(m**4);
145
```

```
5
2401
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

```
147  /* INCREMENT & DECREMENT */
148
149  // Incrementing
150  let x = 5;
151
152  // TODO: Print x++, then print x, then print --x
153  // Try to guess the values that will result with each console.log()
154  console.log(x++); // returns existing value, then increments
155  console.log(x); // new value
156  console.log(++x); // increments, then returns new value
157
158  // TODO: Increment x one more time without printing it, then print x on the next line
159  x++;
160  console.log(x);
```

```
5
6
7
8
```

```
162  // Decrementing
163  let y = 19;
164
165  // TODO: Print y--, then print y, then print y--
166  // Try to guess the values that will result with each console.log()
167  console.log(y--); // returns existing value, then decrements
168  console.log(y); // new value
169  console.log(--y); // decrements, then returns new value
170
171  // TODO: Decrement y one more time without printing it, then print y on the next line
172  y--;
173  console.log(y);
174
```

```
19
18
17
16
```

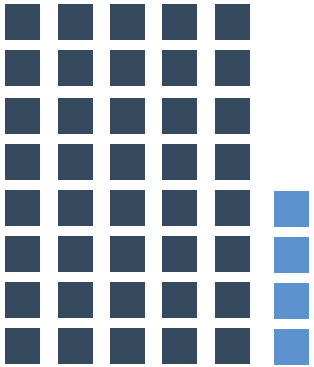
Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Operators

Modulo

- `%` is the **modulo** operator, or "**mod**"
- The **modulus** is the number you're dividing by
- Produces the **remainder** after dividing!



`44 % 8` ⇒ `4`

`%` is the *modulo*
`8` is the *modulus*

5 perfect groups of 8

4 left over

Practical Application

A common use of the modulo operator is for determining if a number is **even** or **odd**:

- If `someNum % 2` is `0`, `someNum` must be **even**
- If `someNum % 2` is `1`, `someNum` must be **odd**

```
175
176  /* MODULO */
177
178  // TODO: Print the remainder from 100 with a modulus of 7
179  // 7 * 14 = 98, so we expect the value to be 2
180  console.log(100 % 7);
181
182  // TODO: Print the remainder from 6 with a modulus of 2
183  // 2 * 3 = 6, so we expect the value to be 0
184  console.log(6 % 2);
185
```

```
2
0
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Operators

Order of Operations

Remember **PEMDAS**:

- **P**arentheses
- **E**xponents
- **M**ultiplication
- **D**ivision
- **A**ddition
- **S**ubtraction

Even with strings instead of numbers, your program needs to know **in which order** it should **evaluate** each **expression**

```
8 * (2 + 2**2) - 36 / (2 * 7 - 2**3)
8 * (2 + 2**2) - 36 / (2 * 7 - 2**3)
8 * (2 + 2**2) - 36 / (2 * 7 - 2**3)
    8 * (2 + 4) - 36 / (2 * 7 - 8)
        8 * 6 - 36 / (14 - 8)
            8 * 6 - 36 / 6
                48 - 6
                  42
```

```
186
187  /* ORDER OF OPERATIONS */
188
189  // TODO: un-comment the line below and run the program to verify the result is 42
190  console.log(8 * (2 + 2**2) - 36 / (14 - 2**3));
191
```

```
42
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Operators

Building Strings

- Combine strings with a **plus sign** +
- This is called **concatenation**
- If both **strings** and **numbers** are included in the expression, as soon as JavaScript encounters a string, it starts treating everything as strings after that (it evaluates from left to right)

```
console.log("Y" + "E" + "S") ⇒  "YES"

console.log(1 + 2 + "3" + 4) ⇒  "334"
```

```
197
198  /* STRING BUILDING */
199
200  let titleAfterMawage = "Princess";
201  let originalName = "Buttercup";
202
203  // TODO: Print the title and name together
204  console.log(titleAfterMawage + " " + originalName);
205
206  // TODO: What happens if you have a mix of strings and numbers? Try it out.
207  let numberOfShips = 4;
208  console.log("You write " + numberOfShips + " copies of a letter. I'll send
     my " + numberOfShips + " fastest ships, one in each direction.");
209
```

```
Princess Buttercup
You write 4 copies of a letter. I'll send my
4 fastest ships, one in each direction.
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Operators

Compound Assignment

These four operators are **shorthand** for assigning a **new value** to an **existing variable:**

- `+=` **Add** and update value — `x += 4` means `x = x + 4`
- `-=` **Subtract** and update value — `x -= 4` means `x = x - 4`
- `*=` **Multiply** and update value — `x *= 4` means `x = x * 4`
- `/=` **Divide** and update value — `x /= 4` means `x = x / 4`

```
192
193  /* COMPOUND ASSIGNMENT */
194
195  let numberOfBoos = 3;
196
197  // TODO: After Buttercup asks the Ancient Woman why she is booing
     her, she booed 5 more times. Increase the value of the variable,
     then print it.
198  numberOfBoos += 5;
199  console.log(numberOfBoos);
200
```

```
8
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Data and Variables

Getting User Input

Interacting with a User in the Console

- **Import** the `readline-sync` library (just **once** at the **top** of the file)
- Assign it to a **constant**, `input`, so you can make use of the library
- `input.question()` does **two** things:
  - **Prints** the string between the parentheses to the **console**
  - **Returns** their response (which you should **store** in a variable to use later in your code)

constant representing library — import library

```
const input = require('readline-sync');

let name = input.question("Enter your name: ");
```

variable to store user response — print question and return user response

```
/**** GETTING USER INPUT ****/

// TIP: During development, you play the role of user so you can test your own code

// TODO: Import the readline-sync library as the constant input
// Normally this would go at the top of the file
const input = require('readline-sync');

// TODO: Print a greeting to the user
console.log("\nWelcome to the Princess Bride Fan Club!");

// TODO: Ask the user for a name and store it in a variable
let favoriteCharacter = input.question("\nWho is your favorite character?\n\t");

// TODO: Print a response to the user that includes their input
console.log("\n" + favoriteCharacter + "? That's my favorite character too!");

// TODO: Ask the user for another name and store it in a variable
let sixFingeredMan = input.question("\nPop quiz! Who is the Six-fingered Man?\n\t");

// TODO: Print a response to the user that includes their input
console.log("\nThat's correct! " + sixFingeredMan + " was the man Inigo was looking for to avenge his father's death.");

// TODO: Go back and clean up the output using special characters so it is easier to read in the console
```

```
Welcome to the Princess Bride Fan Club!

Who is your favorite character?
    Miracle Max

Miracle Max? That's my favorite character too!

Pop quiz! Who is the Six-fingered Man?
    Count Rugen

That's correct! Count Rugen was the man Inigo was
 looking for to avenge his father's death.
```

Starter code: https://replit.com/@CarolineRose/Class01-LectureExamples-Starter-Code
Solution: https://replit.com/@CarolineRose/Class01-LectureExamples-Solution

Intro to Graded Assignment #1

launch_code

Intro to Graded Assignment #1

Graded Assignment #1 - Candidate Testing

Part 1 (due 4/5)

- Get the candidate's **name**
- Ask **one question**
- Get their **response**
- Tell them if their response was the **correct answer** or not

You just need to know how to store information in **variables**, get **user input**, and add **logic** with **conditionals**

Part 2 (due 4/19)

- **Modify** your original quiz to go through the question, response, and user feedback process for **all 5 questions**

Use **conditionals**, **arrays**, and **loops**!

Part 3 (due 4/26)

- Add calculations to **grade** the **quiz**
- Make sure everything prints to the console in a **specific format**

Reminders

- Share code **only with TAs** — no classmates or outside help
- Completed assignment due **4/26** at **5:00 PM** to stay enrolled

What's Next

Class 2 - Conditionals & Debugging - 3/29

- Due before class
  - Prep work (reading, quiz, exercises) - chapters 5 & 6
- Lecture
- Studio - Data & Variables (chapter 4)
- Review

Class 3 - Strings & Arrays - 4/5

- Due before class
  - Prep work (reading, quiz, exercises) - chapters 7 & 8
  - **Part 1** of **Graded Assignment #1**
- Lecture
- Studio
- Review

Studio

launch_code

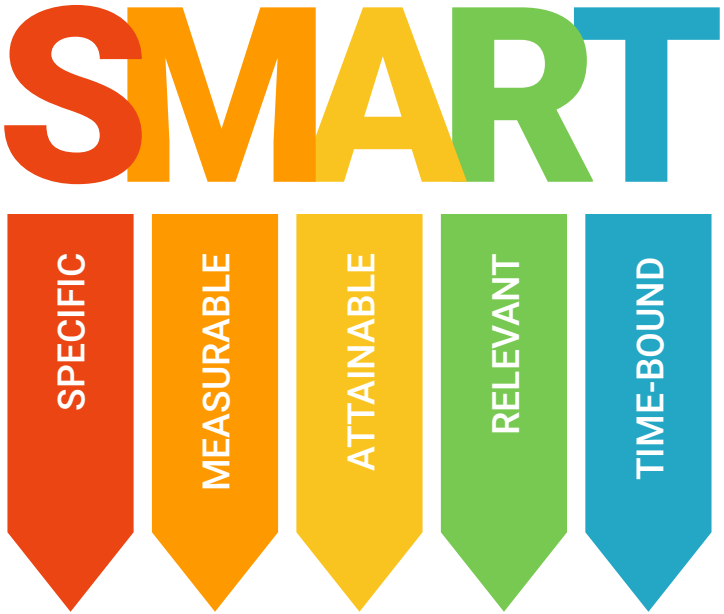
Studio

Tonight's Studio - Chapter 5

SMART Goals & Having a Growth Mindset

- Get to know each other!
- What are your goals for this class?
- What are your goals for your career?
- Think about what motivates & inspires you

No studio review since it's a non-coding studio



Studio

Meet Your TAs

Check the **People** page on **Canvas** — Look yourself up and find your TAs' names in the **Section** column

Track	TA Pair	Room Assignment
Java	Alex E and Ethan	Glass
Java	Ashley and Michael	Westbrooks
Java	Jaimee and Zora	Rivers A/B/C
Java	January and Richmond *(Madison subbing for January tonight!)*	Lewis
Java	Taylor and Ted *(Jordan subbing for Taylor tonight!)*	Brutus A/B
C#	Alex D and Anna	Lecture Hall
C#	Jamey and DaShaun	Lecture Hall