

Hi there!

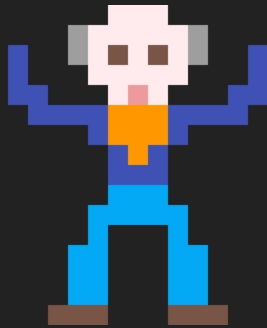
This class is:

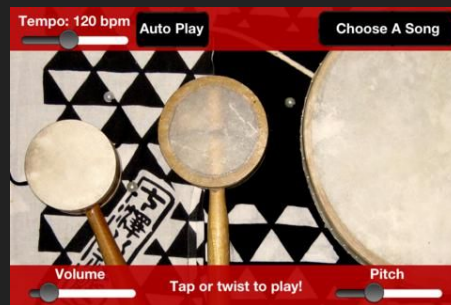
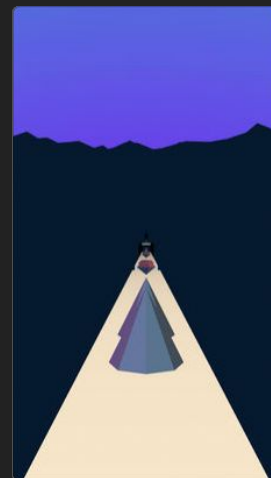
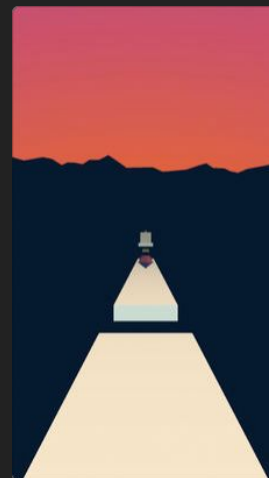
CS 121

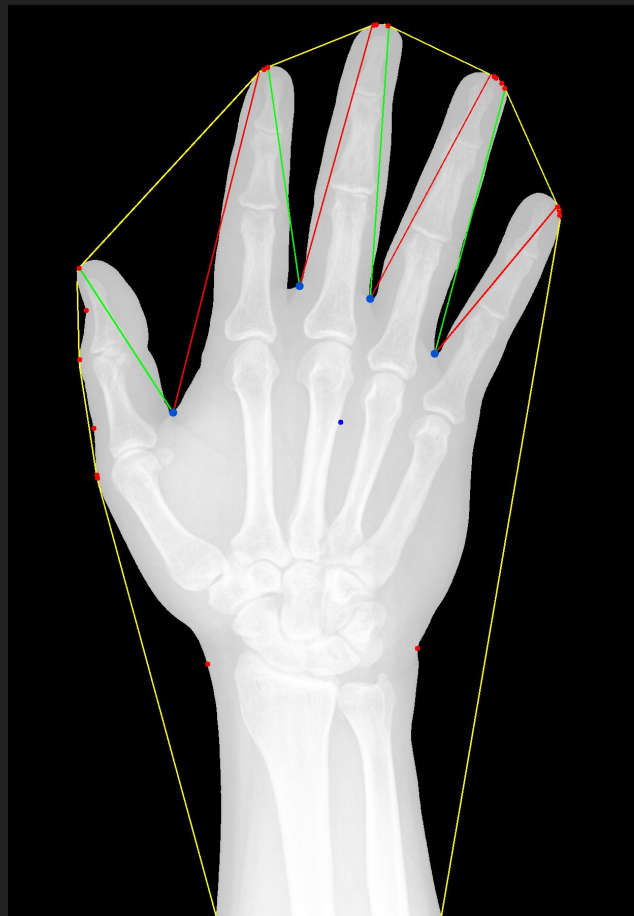
Computer Programming 1

# Prof. Carmine T. Guida

cguida@pace.edu







# Course Format

Online Synchronous

# zyBooks

This course uses an online textbook by zyBooks.

All reading, exercises and assignments are submitted through zyBooks.

Participation activities are required

Challenge questions are optional.

Labs are required

(I'll show you)

# Syllabus

*"It's in the Syllabus!"*

- Every Professor ever.

(let's check it out together)

What's due and when?

# The Schedule!

(let's check it out together)



## Wisdom from the previous semester:

When using zyBooks actually read and understand it because it is a great tool for when you are confused with a certain topic and zyBooks goes into a tremendous amount of detail.

Learn how programs are implemented or used in real life.

Make sure to study everything and not just what's in the review.

Always be engaged because each concept plays a significant role in a future concept.

Don't wait till the night before to do assignments.

You have to do the work or you will be lost.

Students should try to do some outside projects on their own.

If you are new at programming, try practicing each concept piece by piece. Don't let everything build up.

Ask questions (it's cliché but works).

# Computers are Everywhere!



## What's a Computer?

A device that performs a sequence of instructions or logical operations typically to process input into output.

Special-purpose computers:

- Traffic light controllers

- Microwave ovens

- Systems inside of a car

General-purpose computers:

- Desktops

- Laptops

- Smartphones

## Hardware and Software

Hardware: Chips, wires, keyboard, monitor, disks, etc.

Software: Operating systems, programs and their data.

Hardware and Software need each other.

# Hardware Tour

## Hardware Tour

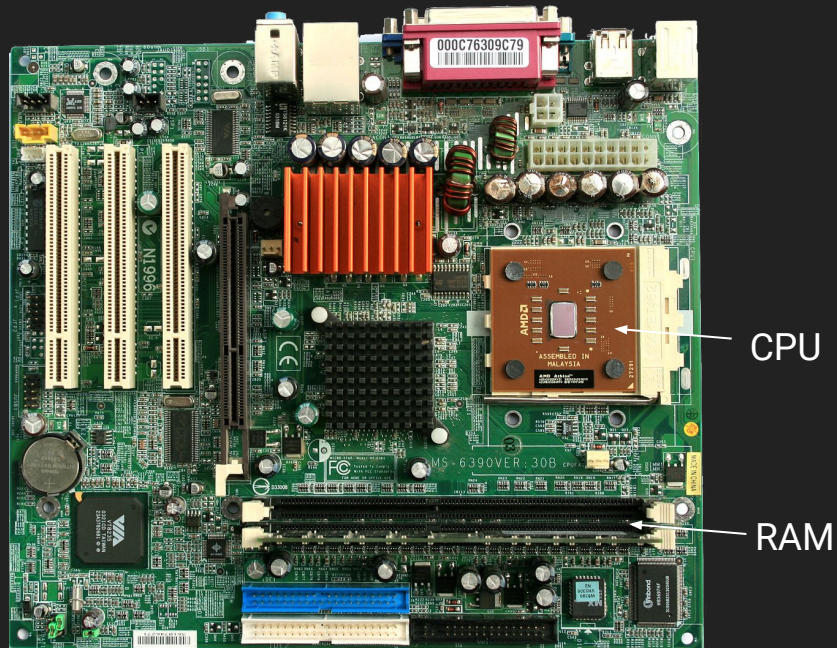
**CPU:** Central Processing Unit. Chip which executes instructions.

**Main Memory:** Primary storage area for active programs and data. Often called RAM (Random Access Memory). *Ephemeral* (good GRE word).

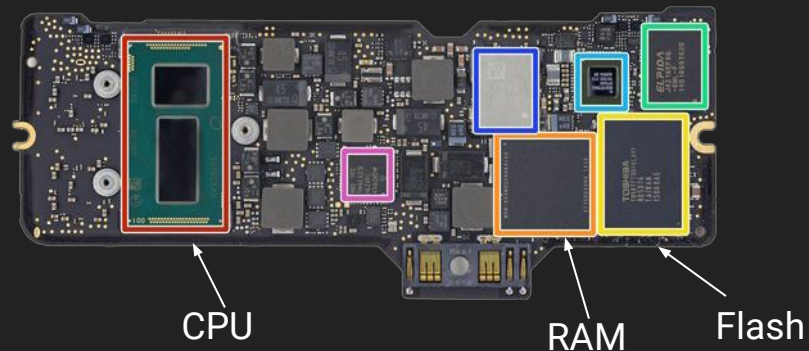
**Secondary Memory:** Long term storage such as Hard Drives, USB Flash Drives, etc.

**Other Components:** GPU (Graphics Processing Unit), Clock, Cache

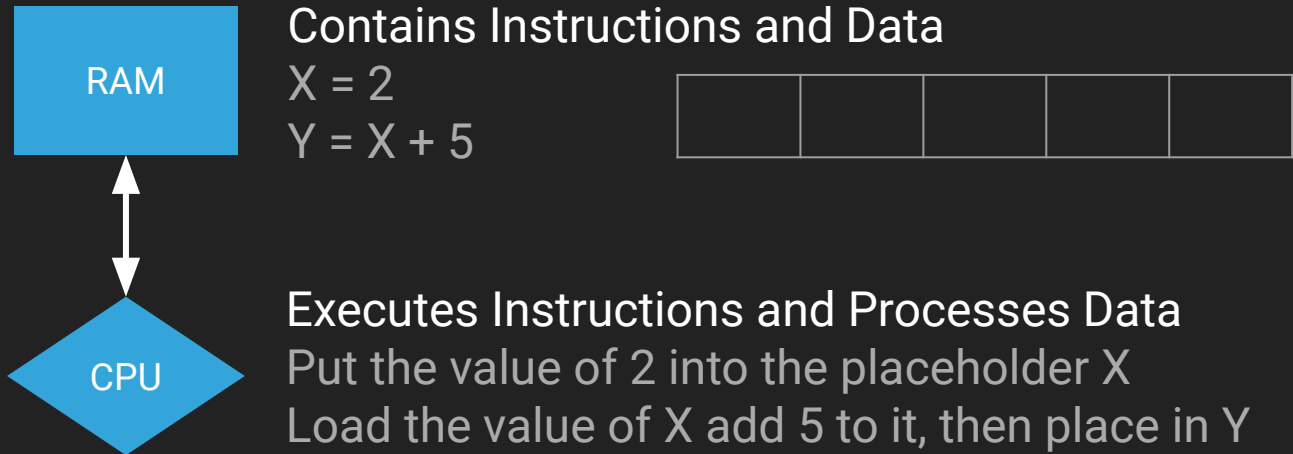
## Desktop



## Macbook



## Processor and Memory





## Input and Output

I/O devices facilitate user interaction.

I/O Devices: Keyboard, Mouse, Touch Screen, Game Controllers, Monitors, Printers, Network Adapters, Microphones, Speakers.

Note some devices are both Input and Output.

Java:

```
System.out.println("Hello, World!");  
Scanner scan = new Scanner(System.in);
```

# Software

## Software Categories

### Operating System

Windows, macOS, Linux

Manages Resources

Security, Permissions, etc.

### Application

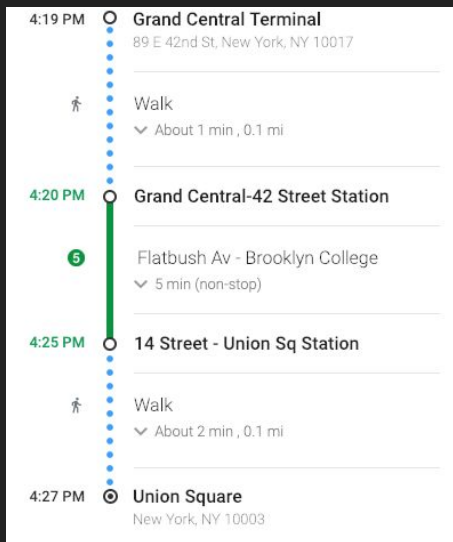
Any kind of software

MS Word, Chrome, Skype, Angry Birds

How do I get to  
The Empire State Building??

# Programming

A program is a set of instructions typically executed in sequence.  
Programming languages specify a vocabulary and grammar rules (syntax).



Preheat oven to 375°. In a small skillet, heat oil over medium-high heat. Add onion; cook and stir 3-4 minutes or until tender. Add garlic; cook 1 minute longer. Cool slightly.

In a large bowl, combine bread crumbs, cheese, eggs, seasonings and onion mixture. Add turkey and beef; mix lightly but thoroughly. Shape into 1-1/2-in. balls.

Place meatballs on a rack coated with cooking spray in a 15x10x1-in. baking pan. Bake 18-22 minutes or until lightly browned and cooked through. If desired, serve with pasta and pasta sauce.

## Java

A High-Level Language (not machine specific).

Created by Sun Microsystems, Inc. in 1995

Java syntax is similar to previous languages such as C and C++

## Why Java (and other) Programming Languages?

Writing in Assembly Language or Machine Code or is hard.

`print("Hello World");`



```
.model small
.data
    msg db 10d,13d,"Hello World$"

.code

    mov ax,@data
    mov ds,ax

    lea dx,msg
    mov ah,09h
    int 21h

    mov ah,4ch
    int 21h

end
```

```
48 83 ec 08
bf c4 05 40 00
e8 cc fe ff ff
b8 00 00 00 00
48 83 c4 08
c3
66 90
```

## Anatomy of a Java Program

Class declaration

(matches name of file)

Method

(main is program entry point)

Variable Declaration

(this is also an assignment)

Print Statements

(these use a library called "System")

Comments

(meaningful documentation about your program)

```
// Comment about the class
public class Hello
{
    // Comment about the method
    public static void main(String[] args)
    {
        // Comment about the variable
        int x = 5;

        // Comment about what we are trying to do here

        System.out.println("Hello, World!");
        System.out.println("X = " + x);
    }
}
```

Compilers ignore comments! They are to help you with reading/understanding your program.

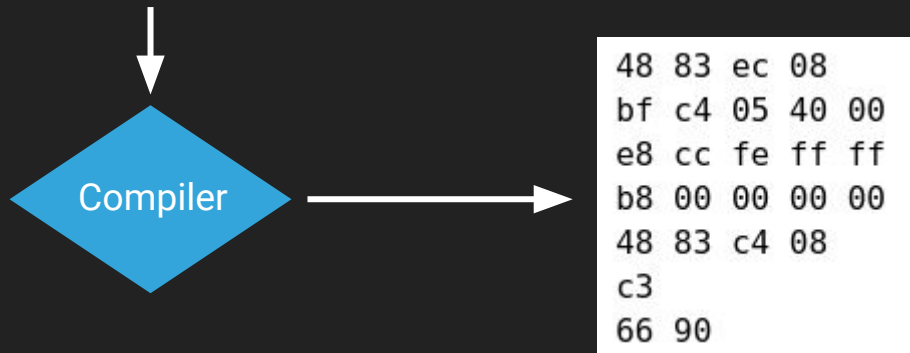


# Compiler

## Compiler

Transforms a high level language into assembly/machine code.

```
public class Hello {  
    public static void main(String[] args)  
    {  
        System.out.println("Hello, World!");  
    }  
}
```



## Compiler

Validates your code following the **rules** of the programming language.

Does not validate spelling (of text) or your **intent**:

The **best** part about programming:  
Computers do exactly what you tell them to do.

The **worst** part about programming:  
Computers do exactly what you tell them to do.

## Write Once, Run Anywhere

The Java compiler turns Java programs into **bytecode** (intermediate language).

The bytecode is run inside a **JVM** (Java Virtual Machine).

There are JVMs for Windows, Mac, Linux, etc.

Only the compiled bytecode is needed (not the original source code).

# Errors

## Errors

**Compile-time** : Syntax errors found by the compiler.

**Run-time** : Occurs during program execution such as trying to divide by zero or opening a file that doesn't exist.

**Logic error** : A program runs and produces output, but the output is incorrect.

# Tools of the Trade

## Tools of the Trade

**Text Editor:** Used to write your lines of code.

**Compiler:** Changes code into machine code (or bytecode).

**Debugger:** Typically used during development to trace a program to find errors.



## Integrated Development Environment (IDE)

An IDE combines a Text Editor, Compiler, Debugger (and more) into one package.

There are several IDEs available such as Visual Studio and XCode.

For this course we will be using an IDE known as Eclipse.

# Installing Eclipse

Google for `jdk 8` - the first link should be:  
“Java SE Development Kit 8 - Downloads - Oracle”  
Choose the one for your operating system.  
Click **accept**. You might need to create an account.

Wait for the JDK to finish installing before going to the next step!

Install eclipse



Link: <https://www.eclipse.org/downloads/packages/>

Look for: “Eclipse IDE for Java Developers” (use this one do not get the Enterprise one).

# Repl.it (browser)

Go to: <https://repl.it>

Works in your web browser (including Chromebooks).

You do not have to create an account, you can scroll to the bottom and look for Java.

If you create an account, you can **save your work**.

This is more convenient than installing an IDE however it is **much slower**.

# Let's Code!

Don't Forget!

Check the syllabus / schedule for reading assignments and due dates!

# Variables

## Variables

A variables is a **named** identifier that holds a value.

data type                      variable name

                                 ↙                      ↘

`int x;`

The diagram illustrates the components of a variable declaration. The text 'data type' has an arrow pointing to the word 'int' in the code 'int x;'. The text 'variable name' has an arrow pointing to the variable 'x' in the same code. The semicolon is also present at the end of the declaration.

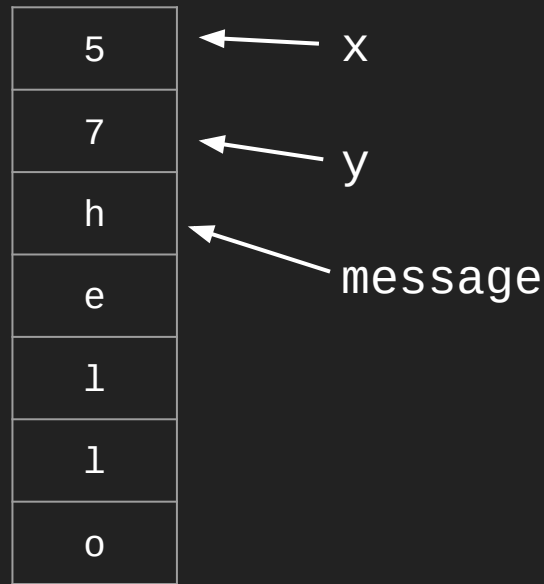
## Variables and Memory

A variable is a reference to a location in memory.

```
int x = 5;
```

```
int y = 7;
```

```
String message = "hello";
```



# Naming Variables



## Naming Style


Examples in naming variables

```
int count;
```

```
int numZombies;
```

```
int daysPerWeek;
```

```
int x, y, z;
```

 Notice the usage of camelCase.

 Multiple declarations on one line.

## Use Expressive Names!

While there are times where names such as *i*, *n*, *x*, *y*, *z* are used, your variable names should indicate what they are used for.

No!

```
int nz;
```

```
int t;
```

Yes!

```
int numZombies;
```

```
int total;
```

## Reserved Words

These words are used by the Java language and can not be used as names for your variables.

abstract  
assert  
boolean  
break  
byte  
case  
catch  
char  
class  
const  
continue  
default  
do  
double  
else  
enum  
extends

final  
finally  
float  
for  
goto  
if  
implements  
import  
instanceof  
int  
interface  
long  
native  
new  
package  
private

protected  
public  
return  
short  
static  
strictfp  
super  
switch  
synchronized  
this  
throw  
throws  
transient  
try  
void  
volatile

## Rules for Variable Names

Can be a sequence of letters (a - z, A - Z), digits (0 - 9)

Can also use \_ and \$ however these are not used as often.

Start with a letter, underscore or dollar sign. (Can not start with a number).

Reserved words can not be used.

## Which of following are valid identifiers?

grade

quizGrade

Network-Connection

frame2

3rdTestScore

MAXIMUM

MIN\_CAPACITY

student#

Shelves1&2

## Which of following are valid identifiers?

grade	Valid
quizGrade	Valid
Network-Connection	Invalid – cannot contain the '-' character
frame2	Valid
3rdTestScore	Invalid – cannot begin with a digit
MAXIMUM	Valid
MIN_CAPACITY	Valid
student#	Invalid – cannot contain the '#' character
Shelves1&2	Invalid – cannot contain the '&' character

## Case Sensitive

Java is case sensitive!

Total, total, TOTAL are different identifiers.

Programmers often have a convention to upper vs. lower case.

Title case for class names: HelloWorld

Upper case for constants: PI, SECONDS\_PER\_HOUR

Lower/camelCase for variables: zombie1, zombie2, numZombies

# Assigning Values



## Assigning a Variable

An assignment statement puts a value into a variable.

Declaration statement      `int x;`

Assignment statement      `x = 5;`

These can be combined on to one line.

```
int x = 5;
```

## Assigning a Variable

The left side is the variable. The right side can be an expression such as a number, another variable or a calculation (and more).

```
int x = 5;
```

```
int y = 0;
```

```
y = y + 1;
```

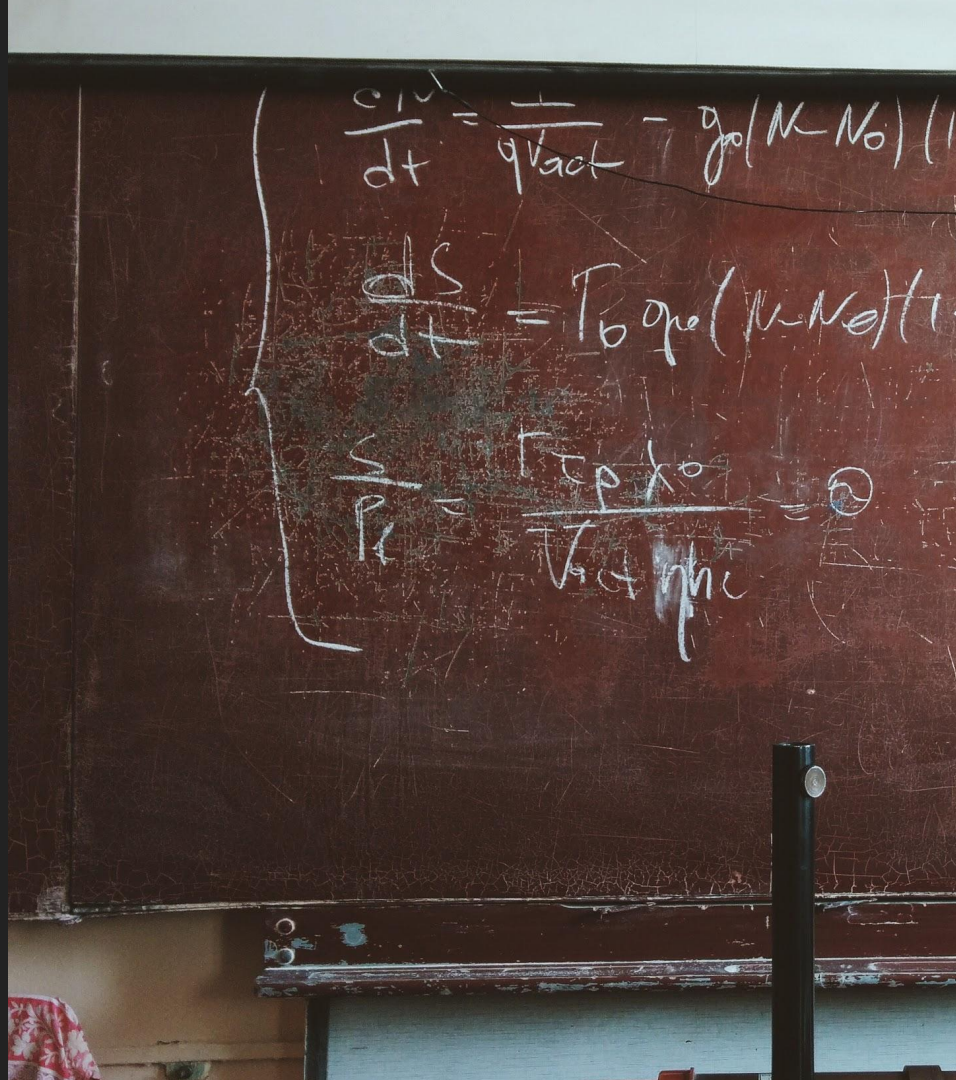
```
x = y + 1;
```

```
y = (5 * 20) + x;
```

# Arithmetic Operators

## Arithmetic Operators

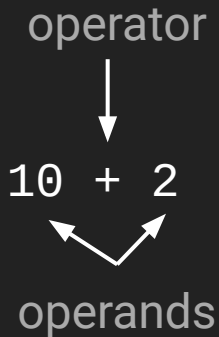
- + Addition
- Subtraction
- \* Multiplication
- / Division
- % Modulo (remainder)



## Operators and Operands

The operator is the arithmetic symbol.

The operands are the input values.



## Order of Operations

When evaluating expressions the following order is used (PEMDAS)

( )          Parentheses

unary -      When evaluating  $2 * -x$   $-x$  is first

\* / %        Multiplication, Division, Modulus

+ -          Addition, Subtraction

Left to Right

## Which of the following are valid?

$x = 5$

$y = x + 2$

$x = 2x$

$y = *2$

$z = (4 + 2) * 5$

$Z = 4 / 0$

## Which of the following are valid?

$x = 5$  Valid

$y = x + 2$  Valid

$x = 2x$  Invalid - needs to be  $2 * x$

$y = *2$  Invalid

$z = (4 + 2) * 5$  Valid

$Z = 4 / 0$  Invalid - Divide by 0 error.



# Let's Code!

Don't Forget!

Check the syllabus / schedule for reading assignments and due dates!