Var- spot in memory if a name, to store info,
Instaliton, giving a name to space in memory,
Assignment, putting some info into that space in memory

Different types of variable, called data types called types,
numbers: / Integers any numbers
String: are Words or letters for english
Boolean: logical operator true vs false
Char: is a string that is one letter
Any letters without “” will make the computer tink

```
operators:
   +  Add
   -  Subtract
* Multiply
/ Divide
% Reminder
++  Adds one
-- Subtracts one
+= Adds a amount to the previous var
 -= Subtracts a amount to the previous var
```

Logical operators;
Turns it into a boolean
=== equal
= makes it that

```
=== (equals)
!== does not equals
>=
```

>

```
console.log(typeof x)
```

+ Adds variables together

```
/*


initialize a variable
assign a variable a value
```

```
types:
string
number
boolean
char

arithmetic operators:
+
-
*
/
%
?
++
--
+=
-=

logical operators:
=== (equals)
!== does not equals
>=

*/


"Still there? Lost you on the room?"

var x = 3;
var y = "Henry"
var z = true;
var answer = "Hello "

if(x > -1){
 answer = answer + " True"
 if(z === true){
   answer += "Mine";
```

```
 }
}

if (y !== "Henry"){
 answer += "A"
}
```

```
var x = 3;
var y = "This"
console.log(typeof 5)
console.log(typeof x)
console.log(typeof "This");
console.log(typeof x === -2);
console.log(typeof y);
console.log(typeof x + y)
/*
number
number
string
boolean
string
string

*/
```

```
var x = 3;

myArray.push(5);
```

```
var x = "Hello";
var y = " World";
var yourArray = [x,33,y];
```

```
var newArray;
newArray =[1,0,2];

newArray.unshift(2);
//2,1,0,2
newArray.pop();
//2,1,0
newArray.pop();
//2,1
newArray.push(1);
//2,1,1



console.log( (newArray[0] + newArray[2]) === 3)

newArray[0] = "Hello"



console.log((yourArray[0] + yourArray[2]))

var x = "Hello ";
// var y = 5;

//Zero indexing: in all coding languages, counting starts from 0, not 1

//primitive data types
//data structures: containers for arrangements of multiple primitive data types, ways of tying data together in a way that makes the control flow of your program make more sense

//Arrays
//lists of data, data
//initialized with square brackets
//can index into with square brackets
```

```
//has functions...

// push, which adds to an array
// pop, removes the last element
// shift, removes an element from the front
// unshift, add an element to the front

// type: array = object
//
// pass by value - the value of a variable, or a reference to a position
in an object, are defined by values, rather than by references to any
other context in which they are defined
// pass by reference - reference the spot in memory, objects and array
//
var newArray;
newArray = [1,0,2];

newArray.unshift(2);
//2,1,0,2
newArray.pop();
//2,1,0
newArray.pop();
//2,1
newArray.push(1);
//2,1,1



var additionalArray = [4,5,6];
additionalArray.unshift(397374632864236);
// newArray.unshift("Hello");

newArray[0] = "Hello"


console.log ( (newArray[0] + additionalArray[1] === "Hello4")    )
```

```
while(x < 15){

 x += 1;
 console.log(x)
}

for(var i = x; i < 10; i++){
 }

var num_stars = 10;
var starter = "*"



for(i = 1; i < num_stars; i++){
 starter = starter + "*";
}
=
```

Var myArr = [1,2,3];

```
for(i = 0; i < myArr.length; i++){
console.log(myArr[i])
 }



var x = 2;
var power = 5;

while(0 < power){

}
```

```
var myFirstDictionary = {
 "key":5,
 "secondkey":"Hello World"
};


var exampleArray = [];


/array indexing -> use square brackets[index of the element you want to
return]


console.log(Array[1]);

//array indexing -> use square brackets[index of the element you want to
return]
var hello = "hi";
var myDictionary = {
"hi":"World"
 };
console.log( myDictionary[hello] === "World");


A function is a reusable piece of code, that usually takes in something,
does something to it, and returns something else


 //name -> the name of the function, you'll use this to reference it later
//arguments - > what goes into the function, these are user defined
// logic, what to do when the function is called
// return statement, which says to return

//multiplyArray
//take an array and an integer, and return the array multiplied by that
integer
```

```
//[1,2,3] - > [2,4,6]
var array = [1,2,3];
function multiplyArray(arg1){
 var x = arg1.length;
 for(i = 0; i < x; i++){
   arg1[i] = arg1[i] * 2;
 }
 return arg1;
};

console.log( multiplyArray(array) ) ;

// var x = 15;

// for(var i = 0; i < 11; i ++){
//   console.log(x + i);
// }
//important data structures:
//array: a list of elements
//object/dictionary: links key values pairs
// indexing into object:
//    1 -> firstDictionary["first"]
//    2 ->


// a function is a reusable piece of code, that usually takes in
something, does something to it, and returns something else

//name -> the name of the function, you'll use this to reference it later
//arguments - > what goes into the function, these are user defined
// logic, what to do when the function is called
// return statement, which says to return



//multiplyArray
```

```
//take an array and an integer, and return the array multiplied by that
integer
//[1,2,3] - > [2,4,6]
var array = [1,2,3];
function multiplyArray(arg1){
 var x = arg1.length;
 for(i = 0; i < x; i++){
   arg1[i] = arg1[i] * 2;
 }
 return arg1;
};

// functional programming
//a programming style that deals with the manipulation of functions,
usually mathematical functions

var myArr = [2,3,2];

function myAdd(arg1,arg2) {
 return arg1 + arg2
}

function myMultiply(arg1){
 var x = arg1.length;
 var y = 1;
 for(i = 0; i < x; i++){
   y = arg1[i] * y;
 }
 return y;
}
//anonymous function with count variable name
var myMultiply = function(arg1){
 var x = arg1.length;
 var y = 1;
 for(i = 0; i < x; i++){
   y = arg1[i] * y;
 }
```

```
 return y;
}

Datatype function
//


function doToArray(myArr, myFnction){
 for(i = 0; i < myArr.length; i++){
   myArr[i] = myFnction(myArr[i])
 }
 return myArr;
}


myArr.forEach(function(x){
 console.log(x)
})



var myArr = [1,2,3,4,5]



console.log(myArr);

function greaterThanTwo(x)
{
 return x > 2
}

function filter(myArr,filter_function){
7
  myArr.forEach (function(myElem){
 if(filter_function(myElem)){
   newArray.push(myElem);
```

```
 }
 })
 return newArray;
}


console.log(filter(myArr,greaterThanTwo))


function addTwo(x){
 return x + 2;
}

function subtractThree(x){
 return x - 3;
}

function doubleMap(myArr,fn1,fn2){
var newArray = [];

myArr.forEach(function(x){
 newArray.push( fn1(fn2(x)) );
})

return newnewArray;
}


console.log(
doubleMap([1,2,3],addTwo,subtractThree)
)



var myObject = {
 "1";"Hello",
 "2":"World"
}
```

```
Object.keys(myObject).forEach(function(key){
 console.log( myObject[key]  )
})

function objectMap(myObj,fn1){
 //returns new object which has the same key value pairs as the original
myObj, just every value is the value paired to that key in myObj with fn1
applied

}
//() =  run function
// no () = referring to the function



user = {
 lizardNames:["Tod","Brian"],
 printLizardFood:function(name){
   if(name === "Tod"){
     console.log("5 lbs of food needed");
     return
   }
   console.log("2 lbs of food needed (Brian's smaller)")

 }
}



function how_many_lizards(user){
console.log(user.lizardNames);

["Tod","Brian"].forEach (function(elem){
 user.printLizardFood(elem);
});
```

```
// forEach(fn1) - > applies that function to everything in the array, one
by one, starting at the beginning, ending at the end

var orc = {
   "Strength":5,
   "Damage": 10,
   "HP":30,
}

var dragon = {
   "Strength":25,
   "Damage": 55,
   "HP":220,
}

//class -> a template for a type of object that will exist in the program,
where the rest of the program can depend on template

class Enemy {
 }

function do_damage(enemy,player){
 player.hp = player.hp - enemy.Damage;
}


class Enemy {
 attack(){

 }
}



how_many_lizards(user);
```

Account

```
function sumAll(myArr){
var x = 0;
myArr.forEach (function(Elem){
   x += Elem;
 })
return x;
}


//polymorphism
//OOP - object oriented programming
// a style of programming based on creating customizable data types in
order to group data together into units in a way that makes sense

//class
//Special type user
User.name
Class = named object
var x = {"key":'value'};


var user = {};

var my_user = {
 dogs:["Benji","Spot","Grover"],
 breeds:["Akita","Golden Retriever","Brown Lab"]
}


function how_many_dogs(user){
console.log(user.dogs);
console.log(user.breeds);

}
```

```
how_many_dogs(my_user);


User.restaurants_visited.forEach(function(restaurant){
 console.log(restaurant)
})



user = {
 "printcatfood":function(){
   console.log("You need five pounds of cat food");
 },
"cats":["Archie","Dot"]
}



function how_many_cats(user){
console.log(user.cats);
user.printcatfood();
}

how_many_cats(user);

//recursion\

      // using functions in functions to loop them
//recursive
//a base case -> the simplest case or 'the lowest case' of the function.
You provide a value that the function should return for the base case.
//Elegant way to write a loop in functions.
// cover other bigger cases by building off of the base case

4
```

```
console.log(raiseTo(2))


function powersOfTwo(raiseTo){
 if(raiseTo === 1){
  return 2;
 }else {
   return 2 * powersOfTwo(raiseTo - 1);
 }
}


console.log(powersOfTwo(2))

//variable scopes
//Way to organize code to not repeat variables.
//scope - a range within which variable definitions remain relevant
//global scope - > everything has access to variables within the global
scope
var x = 2;


function plusTwo(num){
  return num + x;
}

console.log(plusTwo(2));




//function scope -> variables defined within a function only exist within
that function

function plusTwo(num){
 var x = 2;
 return num + x;
```

```
}


console.log(x)
```
Undefined

// variable conclusions - when variables are assigned different values in a group project.

```
// functional programming
//a programming style that deals with the manipulation of functions,
usually mathematical functions\
//OOP - object oriented programming
// a style of programming based on creating customizable data types in
order to group data together into units in a way that makes sense

Class


class Tab {
 constructor(title){
   this.title = title;
 }
}

new Tab("Google");

Tab.title;


//[1,2] -> 3
function sumArray(MyArr){



}
/[1,2] -> 3

function sumArray(MyArr){
```

```
var x = MyArr.length;
//2
var y = 0;
if(x === 0){
//find the base case
return y;
} else {
//get rid of the y
return y + MyArr[x] + sumArray(x - 1);
//turn this into a array
}


console.log(sumArray([1,2]));



//Hi this is me....
//[1,2] -> 3
function sumArray(MyArr){
var x = MyArr.length;
//2
var y = 0;
if(x === 0){
//find the base case
return y;
} else {
//
return y = MyArr[MrArr.length - 1] + sumArray(MyArr.pop());
x
//turn the x into a array but how into a array
}
console.log(sumArray([1,2]));

//x = 2
//y= 0
//x doesn't equal 0 so
```

```
//y = 2 +

var orc = {
   "Strength":5,
   "Damage": 10,
   "HP":30,
}

var dragon = {
   "Strength":25,
   "Damage": 55,
   "HP":220,
}

//class -> a template for a type of object that will exist in the program,
where the rest of the program can depend on template

class Enemy {
 constructor(Strength, Damage, HP){
   this.Strength = Strength;
   this.Damage = Damage;
   this.HP = HP;

 }

 attack(){

 }
}

var WitchKing = new Enemy(80,200,550);

WitchKing.Strength === 80;

var Enemy = {
 attack:function(){
```

```
 }
}
Enemy.attack

function do_damage(enemy,player){
 player.hp = player.hp - enemy.Damage;
}
<canvas id="myCanvas" width="200" height="100">

</canvas>
```

Terminal - -bash

c:/
Is main harddrive

/ folders

Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\coolk>
C:\Users\coolk>ce
'ce' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\coolk>CD
C:\Users\coolk

C:\Users\coolk>cd
C:\Users\coolk

C:\Users\coolk>cd ..

C:\Users>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users>dir

```
 Volume in drive C is OS
 Volume Serial Number is 14EE-6EDD

 Directory of C:\Users

10/28/2019  11:13 PM    <DIR>          .
10/28/2019  11:13 PM    <DIR>          ..
05/05/2020  03:51 PM    <DIR>          coolk
10/28/2019  11:16 PM    <DIR>          defaultuser0
10/29/2019  03:09 AM    <DIR>          Public
               0 File(s)              0 bytes
               5 Dir(s)  12,689,240,064 bytes free

C:\Users>ce coolk
'ce' is not recognized as an internal or external command,
operable program or batch file.

C:\Users>cd coolk

C:\Users\coolk>

Welcome to Node.js v12.16.3.
C:\Users\coolk\Desktop>nodemation.

C:\Users\coolk\Desktop>node example.js
3

C:\Users\coolk\Desktop>
[90m    at internal/main/run_main_module.js:18:47[39m

C:\Users\coolk\Desktop>node example.js henry toll
henrytoll

C:\Users\coolk\Desktop>node example.js 3 6
36

C:\Users\coolk\Desktop>node example.js 3 6
C:\Users\coolk\Desktop\example.js:2
var x = parseInt process.argv[2] + parseInt process.argv[3];
```

```
            ^^^^^^^

SyntaxError: Unexpected identifier
[90m    at wrapSafe (internal/modules/cjs/loader.js:1047:16)[39m
[90m    at Module._compile (internal/modules/cjs/loader.js:1097:27)[39m
[90m    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1153:10)[39m
[90m    at Module.load (internal/modules/cjs/loader.js:977:32)[39m
[90m    at Function.Module._load (internal/modules/cjs/loader.js:877:14)[39m
[90m    at Function.executeUserEntryPoint [as runMain]
(internal/modules/run_main.js:74:12)[39m
[90m    at internal/main/run_main_module.js:18:47[39m

C:\Users\coolk\Desktop>node example.js 3 6
C:\Users\coolk\Desktop\example.js:2
var x = parseInt(process.argv[2]) + parseInt(process.argv[3];)
                                             ^

SyntaxError: missing ) after argument list
[90m    at wrapSafe (internal/modules/cjs/loader.js:1047:16)[39m
[90m    at Module._compile (internal/modules/cjs/loader.js:1097:27)[39m
[90m    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1153:10)[39m
[90m    at Module.load (internal/modules/cjs/loader.js:977:32)[39m
[90m    at Function.Module._load (internal/modules/cjs/loader.js:877:14)[39m
[90m    at Function.executeUserEntryPoint [as runMain]
(internal/modules/run_main.js:74:12)[39m
[90m    at internal/main/run_main_module.js:18:47[39m

C:\Users\coolk\Desktop>node example.js 3 6
9

C:\Users\coolk\Desktop>c

login as: 22henryt
22henryt@park.lrei.org's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-71-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu May  7 15:21:34 EDT 2020
```

```
  System load:  0.0               Processes:           111
  Usage of /:   17.7% of 29.40GB   Users logged in:     1
  Memory usage: 28%                IP address for eth0: 162.243.26.34
  Swap usage:   0%                 IP address for tun0: 10.8.0.1

  Graph this data and manage this system at:
    https://landscape.canonical.com/

316 packages can be updated.
245 updates are security updates.

New release '16.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu May  7 15:21:34 2020 from p-74-209-19-88.dsl1.rtr.chat.fpma.frpt.net
22henryt@park:~$ cd website
22henryt@park:~/website$ ls -l
total 16
-rw-r--r-- 1 22henryt cli  48 Apr 17  2019 hello.js
-rw-r--r-- 1 22henryt cli 172 May  7 15:23 index.html
-rw-r--r-- 1 22henryt cli  36 Apr 17  2019 index.html.save
-rw-r--r-- 1 22henryt cli  41 Apr 17  2019 test.js
22henryt@park:~/website$ nano index.html
22henryt@park:~/website$ ^C
22henryt@park:~/website$
```

```
function test(ar1,ar2,ar3,ar4){
var object = {"CO1":"ar1", "CO2":"ar2","CO3":"ar3","CO4":"ar4"};

return [object.CO1, object.CO2, object.CO3, object.CO4];
}
console.log(test(2,3,4,5));

function test(ar1,ar2,ar3,ar4){
var object = {"CO1":"ar1", "CO2":"ar2","CO3":"ar3","CO4":"ar4"};
```

```
return [object.CO1, object.CO2, object.CO3, object.CO4];
}



function secondstoMinutes(numSeconds){
 return numSeconds / 60;
}

function areaofASquare(side_length){
 return side_Length^2;
}

function sumLessThan100(num1,num2){
 x = num1 + num2;
 if(x >= 100){
   return false
 } else{
   return true
 }
}

function returnNumSeconds(numHours,numMinutes){
 numMinutes += numHours * 60;
 return numMinutes * 60;
}

function AreTheseArraysTheSame(arr1,arr2){
 if(arr1 === arr2){
 return true
 } else{
 return false
 }

function AreTheseArraysTheSame(arr1,arr2){
```

```
 if(arr1 === arr2){
   return true
 }

 return false
 }
```

```
function AreTheseArraysTheSame(arr1,arr2){
 var x = [1,2,3];
 x = y;
Vs
 return x === y;
}
function AreTheseArraysEquivalent(arr1,arr2){
 if(arr1.length === arr2.length){
 for(i = 0; i <  arg1.length; i++){
   if(arr1[i] !== arr2[i]){
     return false;
   }
 }
 return true
}

[5,32,1]
function sort(myArr){
//return the sorted version of myArr
}
//try not to use a second array
[1,5,32]


//[5,32,1]
function sort(myArr){
 var newArr = {};
```

```
    for(i = 0; i <  myArr.length; i++){
     var c = i + 1
   if(myArr[i] > myArr[c]){
     newArr.unshift(myArr[i]);
     }
 return

//return the sorted version of myArr
}
}
//try not to use a second array
//[1,5,32]
//return [1,5,32]
   if(myArr[i] < myArr[x]){
//[5,32,1]
function sort(myArr){
 var newArr = {};
   for(i = 0; i <  myArr.length; i++){
     var c = i + 1
   if(myArr[i] > myArr[c]){
     newArr.unshift(myArr[i]);
     }
 return

//return the sorted version of myArr
}
}
//try not to use a second array
//[1,5,32]
//return [1,5,32]
console.log(sort([5,1,3]))

//[5,32,1]
function sort(myArr){
 var newArr = {};
   for(i = 0; i <  myArr.length; i++){
     for(x = 0; i <  myArr.length; i++)
```

```
   if(myArr[i] < myArr[x]){
   //how do I change
     newArr.unshift(myArr[i]);
     }

 return


test.key1// value1
test["key1"] // also value1




var test = {"key1":"value1","key2":"value2"}

test.c// value1
test["cat"] // also value1

//myArr = [2,3,4]
function reduce(myArr, accumulator_function){
var x = 0;
if(myArr.length === 0){
return myArr[0]
} else{
function (for(i = 0; i < myArr.length; i++){
 accumaltor_function(myArr[i] );

  accumulator_function(for(i));
}

}

function add(x,y){return x + y;}
```

```
function multiply(x,y){return x * y};

function accumulator(x,y){
 return x;
}

reduce([1,2,3], add) === 6;


function addDictionary(myDict){
 var x = "";
 var arr = Object.keys(myDict);
 for(i = 0; i < arr.length; i++){
 x += myDict[arr[i];


 }
return x;

}
console.log(addDictionary(test))
console.log(addDictionary(test) === "value1value2")




[1,2,3,4,5]


var test = {"key1":"value1","key2":"value2"}

test.c// value1
test["cat"] // also value1

//myArr = [2,3,4]
```

```
function reduce(myArr, accumulator_function){
var x = 0;
if(myArr.length === 0){
return myArr[0]
} else{
function (myArr (for(i = 0; i < myArr.length; i++){
 accumaltor_function(myArr[i] );

  accumulator_function(for(i));
}

function reduce(myArr, accumulator_function){
 if(myArr.length === 0){
 return myArr[0]
} else{
var y = 0;
 for(i = 0; i < myArr.length; i += 2, {
 var x = i + 1
 y += accumulator_function(myArr[i], myArr[x])
      }
      }
}

function reduce(myArr, accumulator_function){
if(myArr.length === 0){
return myArr[0]
} else{
var y = 0;
  for(i = 0; i < myArr.length; i ++);{
   y = accumulator_function(myArr[i], y);
   }

 }
 return y;
}
```

```
function reduce(myArr, accumulator_function){
if(myArr.length === 0){
return myArr[0]
} else{
var y = myArr[0];
  for(i = 1; i < myArr.length; i ++);{
   y = accumulator_function(myArr[i], y);
   }

 }
 return y;
}

[1,2,3], add
```

Multi array

```
///single dimensional array
//[1,2,3]
// var myArray = [[1,2,3],[4,5,6],[7,8,9]];

myArray[0][1] === 2

myArray[1][2] === 6
```

```
function flatten(myMultiArray){
var newArray = [];
 for(i = 0; i < myMultiArray.length; i++){
   for(b = 0; b < myMultiArray[i].length; b++){
    newArray.push(myMultiArray[i][b]);
   }
 }
return newArray;
}
```

```
flatten([[1,2,3],[4,5,6]])

function slice(arr,index){
var newArray = [];
for(i = 0; i < arr.length; i++){
 if(i !== index){
 newArry.push(i);
   }
 }
}

function slice2(arr,index){
// [1,2,3,4,5],2
 for(i = 0; i < arr.length; i++){
   arr.pop();
   if(i !== index){
 arr.push(i);
   }
 }
}

var arr = [1,2,3,4];

slice2(arr,3);

console.log(arr);



function slice2(arr,index){
// [1,2,3,4,5],2
var temp = arr[0];
for(var i = 1; i < temp.length; i++){
  if(index >= i){
    arr[i] =  arr[i+1];
  }
}
```

```
arr.pop([arr.length - 1]);
}

https://lodash.com/docs/4.17.15#uniq

var myArr = [1,2,3];
myArr.indexOf(1) = 0;

-1 =
```

unque

```
[2,1,2] -> [2,1]
[1,2,2,2,3,4] - >[1,2,3,4]

function

function check(myArr,checky){
for(var i = 0; i < myArr.length; i++){
 if(myArr[i] === checky){
   return true
   }
 }
 return false
}

function unque(arr){
var newArr = [];
 for(var i = 0; i < arr.length; i++){
   if(check(arr,i) === false){
   newArr.shift(i);
   }
 }
 return newArr;
}
console.log(unque(2,3,1,2))
```

Encapsulation separate coding functions instead of one big.

```
function unque(arr){
 var newArr = [];
console.log(arr.length)
 for(var i = 0; i < arr.length; i++){
   if(check(newArr,arr[i]) === false){
   newArr.unshift(arr[i]);
   console.log(arr[i]);
   }
 }
return newArr;
}

function check(myArr,checky){
console.log("here")
for(var i = 0; i < myArr.length; i++){
 console.log("HERE")
 console.log(myArr[i]);
 if(myArr[i] === checky){
   return true
   }
 }
 return false
}


function count(arr){
 var obj = {1:0};
 for(i = 0; i < arr.length; i++){
   if(obj[1] === undefined){
     obj[arr[i]] = 0;
   } else {
     obj[arr[i]] = obj[arr[i]] + 1;
   }

 }

 return arr;
```

```
}
```

Time efficiency

```
// [1,1,2,2,2,2,3] - > {1:2,2:4,3:1};

fu

function count(arr){
new final = {};
new newArr = [];
for(i = 0; i < arr.length; i++){

 if(c(newArr,arr[i]) === false){
 }
}
}



function count2(arr,checker){
var count = 0;
 for(i = 0; i < arr.length; i++){

 }

var obj = {};

// ["Hello","Hello","World","HEY","HEY","HEY"]

// {Hello:3, World:1, HEY:3}


// [1,1,2,2,2,2,3] - > [1,1,2,2,3]
```

```
function firstX(arr,amountOfDuplicatesAllowed){
 }

function firstTwo(arr){
new obj = {};
 for(i = 0; i < arr.length; i++){
   if(obj[arr[i]] === undefined){
     obj[arr[i]] = 1;
   } else {
   if(obj[arr[i]] > 2){

   }
   }
 }
}



// ["Hello","Hello","World","HEY","HEY","HEY"]

// {Hello:3, World:1, HEY:3}



function firstTwo(arr){
var obj = {};
var jArray = [];



for(i = 0; i < arr.length; i++){
   if(obj[arr[i]] === undefined){
    obj[arr[i]] = 1;
   // console.log(obj);
  } else {
    obj[arr[i]] += 1;
  }
```

```
 // console.log(obj);
}
Object.keys(obj).forEach (function(Elem){
   if(obj[Elem] >= 2){
     jArray.push(Elem);
     jArray.push(Elem);
     }  else {
   jArray.push(Elem);
   }
})
return jArray;
}


console.log(firstTwo([1,1,1,2,2,2,3,3,3]))


function zerototen(x){
Math.floor((Math.random() * 5))  + 1
}

console.log(zerototen(5))



function zerototen(x){
Math.floor((Math.random() * 5))  + 1
}

function kindaShuffle(arr){
var newArr = [];
for(i = 0; i < arr.length; i++){
c = arr[i];
c = Math.floor((Math.random() * c ));
console.log(c);
if(c >= 2.5){
newArr.unshift(arr[i]);
```

```
}else{
newArr.push(arr[i]);
}
c = 0;
}
return newArr
}

function realShuffle(arr){
newArray = [];
for(i = 0; i < arr.length; i++){
c = Math.floor((Math.random() * arr.length));
newArray.unshift(arr[]);
}
return newArray;
}



console.log(realShuffle([100,2,3,4,5]))


function number (){
c = Math.floor((Math.random() * 10));
return c
}

function realShuffle(arr){
var newArray = [];
var c = 0;
for(i = 0; i < arr.length; i++){
c = number(1);
if(c >= 5){
newArray.unshift(arr[i]);
}else {
newArray.push(arr[i]);
}
```

```
}
return newArray;
}
console.log(number(1))
console.log(realShuffle([3,2,4]));



function greaterThanTwo(arg){
return arg > 2;
}
// var arr = [0,1,2,3,4,5,6]
// separate(arr,greaterThanTwo) -> [[0,1],[2,3,4,5,6]]
function separate(arr,filter_function){
var array1 = [];
var array2 = [];
for(i = 0; i < arr.length; i++){
 if(filter_function(arr[i]) === true){
 array1.unshift(arr[i]);
 } else{
 array2.unshift (arr[i]);
 }
}
return [array1,array2];
}
console.log(separate([1,3,5], greaterThanTwo));

funct

function sample(arr){
c = Math.floor((Math.random() * array.length));
return arr[c]
}
```

. = [“string”]
[var]

```
var obj = {};
var obj = {"key":2}
obj.key
obj.key = 4;


obj = {"key":{"k":"j"},"other_key":"other_value"};
function clone(obj){
 var newObject = {};
 var c = 0;
 Object.keys(obj).forEach(function(Elem){

   if(typeof obj[Elem] === "object"){
       newObject[Elem] = clone(obj[Elem]);
   } else {
     newObject[Elem] = obj[Elem];
   }

 })
return newObject;
}


function reverse(arr){
var newArr = [];
arr.forEach (function(Elem){
 newArr.unshift(Elem);
})
return newArr
}

function addTwo(x){return x + 2}

function multiplyByThree(x){return x * 3;}

console.log(metaFunction(addTwo,multiplyByThree,2));
```

```
function metaFunction(fn1,fn2,argument1){
return fn2(fn1(argument1));
}

console.log (doAll([addTwo,addTwo,multiplyByThree],2));
function doAll(fnArray,argument1){
x = argument1;
fnArray.forEach (function(Elem){
x = Elem(x);
})
return x;
}



reverse([1,2,3])

var obj = {};
var obj = {"key":2}
obj.key
obj.key = 4;

var str = "k"
var x = {"k":"v"}
x[str] === v



Object.keys (function)
Object.”key”



function showAll(objectOfArrays){
var x = Object.keys(objectOfArrays)
 for(i = 0; i < x.length; i++){
   Object.keys (objectOfArrays).forEach (function(Elem){
```

```
      console.log(objectOfArray[Elem]);
    });
  }
}



var x = {"k":[1,2,3],"marvel":[4,5,6]}


function objToArr(obj)
{
var newArray = [];
Object.keys(obj). forEach (function (Elem){
console.log(Elem)
console.log(obj[Elem])
newArray.push(obj[Elem])
})
return newArray
}
console.log(objToArr(x));

var arr = [1,2,3,4];
arr.splice(1,2)
//2,3




var arr = [1,2,3,4];
arr.splice(1,2)
//2,3

function removeEveryX(arr,num){
 for(var i = 0;i<arr.length;i++){
   if(i%num === 0){
        arr.splice(i,1)
```

```
    }
  }
}
```

```
function removeEveryX(arr,num){
var c = 0;
for (var i = 0; i < arr.length; i++){
 console.log(c);
 console.log(i);
 for(c = num; c > 0 ; c--){
    if(arr[i] = arr[c * arr.length]){
     console.log("spliced")
     console.log(i)
     arr.splice(i,1);
   }
 }
}
return arr;
}

var v = [1,2,3,4,5,6,7];
removeEveryX(v,3);

//[1,2,4,5,7]
```

Class

```
var User = { Height:90, eyeColor:"Blue", Hair:"Brown"};
var User2 = { Height:2, eyeColor:"Pink", Hair:"Brown"};
var User3 = { Height:102, eyeColor:"Brown", Hair:"Black"};

function printHairColor(user){
return console.log("This user's haircolor is " + user.Hair);
}
```

```
class User
{
 constructor(height,eyeColor,hair){
   this.Height = height;
   this.EyeColor = eyeColor;
   this.Hair = hair;
 }

 add(x,y){
   return x  + y;
 }
}

var User1 = new User("50 inches","brown","brown");

printHairColor(User)



class Dog
{
constructor(Breed,Bark){
 this.Breed = Breed;
 this.Bark = Bark;
}
}

var x = new Dog("Golden", "woof");
console.log(x)



class user
{
 constructor(Id, Password, FavoriteMovies){
   this.Id = Id;
   this.Password = Password;
```

```
    this.FavoriteMovies = FavoriteMovies;
  }

  findId(){
    console.log(this.Id);
  }
   findPassword(){
    console.log(this.Password)
  }
   findFavoriteMovies(){
    for(var i = 0; i < this.FavoriteMovies.length; i++){
    console.log(this.FavoriteMovies[i] + " ")
    }
  }

findFavorites(arr){
 for(var i = 0; i < this.FavoriteMovies.length; i++){
 var x = 0;
 var c = arr.indexOf(this.FavoriteMovies[i]);
 c -= arr.length;
 x = c + 1;
 }
 return x
 }
}



var user1 = new user("bobK", "1234",["Star Wars 7","Star Wars 1","Star
Wars 2"]);
var user2 = new user("kellyS", "drowssap",['The Room','The Birds']);
user1.findId()
user2.findId()
user1.findFavoriteMovies()
user1.findFavorites(["Star Wars 1","Titanic","Jurassic Park"])
```

```
//class creation Korean Horror films and add some fields and methods



class guitarOwner
{
constructor(Fendor, Gibson, Ibanez){
  this.Fendor = Fendor;
  this.Gibson = Gibson;
  this.Ibanez = Ibanez;
}
 findGibson(){
  console.log(this.Gibson)
}
findAllGuitars(){
  console.log(this.Fendor,this.Gibson,this.Ibanez)
}
}
var Tom = new guitarOwner("JazzMaster", "Flying V","JS2480");
var Bill = new guitarOwner("Stratcastor", "Les Paul",'THBB10');
Tom.findGibson();
Bill.findAllGuitars();
//class creation Korean Horror films and add some fields and methods



class Movie
{
constructor( Title, Genre, Description, Rating){
  this.Title = Title;
  this.Genre = Genre;
  this.Description = Description;
  this.Rating = Rating;
}
}
```

```
var It = new Movie("It", "Horror", "A Clown harassing some kids" ,
"7/10");

class NetflixUser
{
 constructor(Id, Password, FavoriteMovies){
   this.Id = Id;
   this.Password = Password;
   this.FavoriteMovies = FavoriteMovies;
   this.MovieDictonary = {};
 }
  addtoFavoriteMovies(newMovie){
   this.FavoriteMovies.push(newMovie);
 }

 howManyTimes(kMovie){
 if(this.MovieDictonary[kMovie.Title] === undefined){
   this.MovieDictonary[kMovie.Title] = 1;
   }else (
   this.MovieDictonary[kMovie.Title] += 1
   )
 }
}

var Jerry = new NetflixUser("Jerry3", "dorwssap", ["It","Kill Bill"]);
Jerry.howManyTimes(It);
console.log(Jerry.MovieDictonary);
Jerry.howManyTimes(It);
console.log(Jerry.MovieDictonary);




class Movie
{
constructor( Title, Genre, Description, Rating){
  this.Title = Title;
```

```
  this.Genre = Genre;
  this.Description = Description;
  this.Rating = Rating;
}
}

var It = new Movie("It", "Horror", "A Clown harassing some kids" , "96");

class NetflixUser
{
 constructor(Id, Password, FavoriteMovies,AllMovies){
   this.Id = Id;
   this.Password = Password;
   this.FavoriteMovies = FavoriteMovies;
   this.MovieDictonary = {};
   this.AllMovies;
 }
  addtoFavoriteMovies(newMovie){
   this.FavoriteMovies.push(newMovie);
 }

 howManyTimes(kMovie){
 if(this.MovieDictonary[kMovie.Title] === undefined){
   this.MovieDictonary[kMovie.Title] = 1;
   }else (
   this.MovieDictonary[kMovie.Title] += 1
   )
  }
 likingMovies(){
   var trueLike = {};
   this.MovieDictonary.forEach(function(Elem){
   var GenreApprecitation = howManyTimes(Elem) * howManyTimesGenre(Elem);
   trueLike.Elem = Elem.Rating * GenreAppreciation;
   })
   return truelike = {};
 }
FavoriteFavoriteMovies(){
```

```
    var FavoriteFavoriteMovies = {};
    this.LikingMovies().forEach(function(Elem){
    if(trueLike.Elem >= 90){ //findway to get a more accurate cut off
    FavoriteFavoriteMovies.Elem
    }
    })
  }
  reccomendedMovies(){
    Object.keys(this.FavoriteFavoriteMovies()).forEach(function(){

    })
  FavoriteFavoriteMovies(this) forEach(function(Elem){
  If(true)
  })
  }

}

var Jerry = new NetflixUser("Jerry3", "dorwssap", ["It","Kill Bill"]);
Jerry.howManyTimes(It);
console.log(Jerry.MovieDictonary);
Jerry.howManyTimes(It);
console.log(Jerry.MovieDictonary);


8



class Sports{
  constructor(name,score){
    this.name = name;
    this.score = score;
  }


}
```

```
var andy =
[{name:"bball",score:2},{name:"tennis",score:5},{name:"soccer",score:5}]

var ben =
[{name:"bball",score:1},{name:"tennis",score:2},{name:"soccer",score:7}]
//helps two friends decide what sport to play
function returnDualFavorite(person1SportScores ,person2SportScores){

}
```

```
class Sports{
 constructor(name,score){
   this.name = name;
   this.score = score;
 }


}

var andy =
[{name:"bball",score:2},{name:"tennis",score:5},{name:"soccer",score:5}]

var ben =
[{name:"bball",score:1},{name:"tennis",score:2},{name:"soccer",score:7}]
//helps two friends decide what sport to play
function returnDualFavorite(person1SportScores ,person2SportScores){
var running count =
all sports

}
```

```
class Sports{
constructor(name,score){
  this.name = name;
  this.score = score;
}
}
var andy = [{"bball":2},{"tennis":5},{"soccer":5}]
var ben = [{"bball":32},{"tennis":19},{"soccer":7}]
//helps two friends decide what sport to play


function returnDualFavorite(person1SportScores ,person2SportScores){
var runningCount = 0;
var returnOne = "None";
var obj = {};

for(var i = 0; i < person1SportScores.length; i++){
Object.keys(person1SportScores[i]).forEach(function(key){
 obj[key] = person1SportScores[i][key]
})
}



for(var b = 0; b < person2SportScores.length; b++){
 Object.keys(person2SportScores[b]).forEach(function(key2){
   if(obj[key2] !== "undefined"){
     obj[key2] += person2SportScores[b][key2];
     if(obj[key2] > runningCount){
       runningCount = obj[key2]
      // console.log(runningCount);
       returnOne = key2;

     }
   }
 })
}
```

```
return returnOne;
}

returnDualFavorite(andy,ben);
```

Java versus

```
class Car
{
 int carSpeed;
int carRating;
String carBrand;

public Car(int inputtedCarSpeed, int inputtedcarRating, String
inputtedcarBrand){
 carSpeed = inputtedCarSpeed;
 carRating = inputtedcarRating;
 carBrand = inputtedcarBrand;
}

public void printCarBrand(){
 System.out.println(carBrand);
}
```

```
}
class Main {
```

```
 public static void main(String[] args) {
 Car myCar = new Car(120,80,"Tesla");
 myCar.printCarBrand();
 }
}
//System.out.println
//class main = where the world starts
//public vs private = Scoop
//static =
//int = interger
//String = letters
//Boolean = booleans
//for loop the same
//if are the same
//== equals
//functions in java needs to have a return type and methods
//void doesn't return anything
```

cat class

make some more System.out.println methods