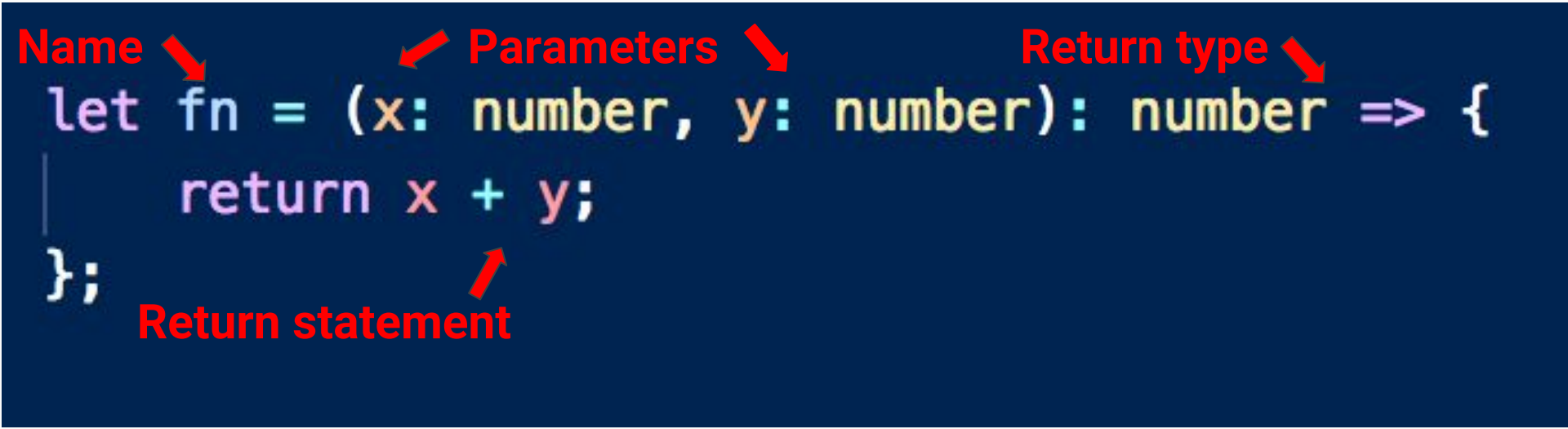
# COMP 101: Review Session 4

With your boys Tony and Mason

# Announcements

- Tony isn't sick anymore
- Quiz 4 tomorrow
  - Equal parts functions and classes
  - Important topics could include function calls, return statements, parameters/arguments, objects, properties, constructors
  - From this week- won't need to call `csvToArray()` , but may need to work with an array of objects
- Registration coming up- let us know if you're thinking about taking another COMP class!

# Anatomy of a Function



# `return` statements- Conceptual Review

- If a function has a return type other than `void`, we need a return statement that returns a value/object of the correct type
- When the computer hits that return statement, it immediately stops executing the function and sends a given value back to the calling function
- Only one return statement will be evaluated in a function call

```
export let main = async () => {
    let result = fn(3, 5);
    print(result);
};
```

```
let fn = (x: number, y: number): number => {
    return x + y;
};
```

# Boolean Questions

T - Functions can have more than one return statement.
F - Not all functions with a return type have a return statement.
F - We should not make a return statement inside a for loop.
T - Two variables in the same function can have the same name.
T - All variables have a type and a scope.
T - Parameters are local variables for the function.
F - We cannot define a variable inside a function of the same name.
T - Functions help us avoid repetitive code.

# Scope

Scope lets us know where a variable can be used. Just two rules:

1. The scope of a variable is defined by the innermost curly braces that surround its declaration
2. Two variables cannot have the same name if their scopes overlap

What will be printed when we run the code to the right?

```
let b = 50;
print(b);
if (b > 30) {
    let b = 100;
    print(b);
}
print(b);
```

50
100
50

# return statements- Spot the Problem

```
let a = (): number => {
    return 3 + "4";
};
```

Returns string “34”, but return type is number

```
let b = (): number => {
    return 2;
    print("Hello");
};
```

“Hello” will not be printed- stop executing after return

```
let c = (): number => {
    return 1;
    return 2;
};
```

Second return is unreachable, since we always stop executing after first return

```
let d = (): number => {
    print(4);
};
```

Non-void functions must have a return statement that matches the return type

# Hands-On #1

What is printed? “110101”

`sum()` will always return the first number in the array, since there is a return statement inside the loop.

`awe(3, “01”)` → “110101”

```
export let main = async () => {
    print(awe(sum([3, 5, 7, 9]), "01"));
};

let sum = (n: number[]): number => {
    let total = 0;
    for (let i = 0; i < n.length; i++) {
        return total + n[i];
    }
    return total;
};

let awe = (q: number, w: string): string => {
    let res = "11";
    while (q < 6) {
        res += w;
        q += w.length;
    }
    return res;
};
```

# Hands-On #2

1) What is printed?

   1, 4

2) Call the b function so that it will return 100.

   Example: `b(0,10)`

3) How many parameters does a have? How many arguments does b have?

   2, 4

```
let a = (n2: number, n1: number): number => {
    return n1 - n2;
};

let b = (n1: number, n2: number): number => {
    return a(n1, n2) * -a(n2, n1);
};

export let main = async () => {
    print(b(3, 2));
    print(b(4, 6));
};
```

Check in on course.care!

C413C

# Classes and Objects

- A **class** is a blueprint or template that can define properties and capabilities for multiple objects.
- An **object** is a specific “instance” of the class that we can interact with in our program.
- If we think of classes as a “type”, like string or number, then objects of that class are variables of that type

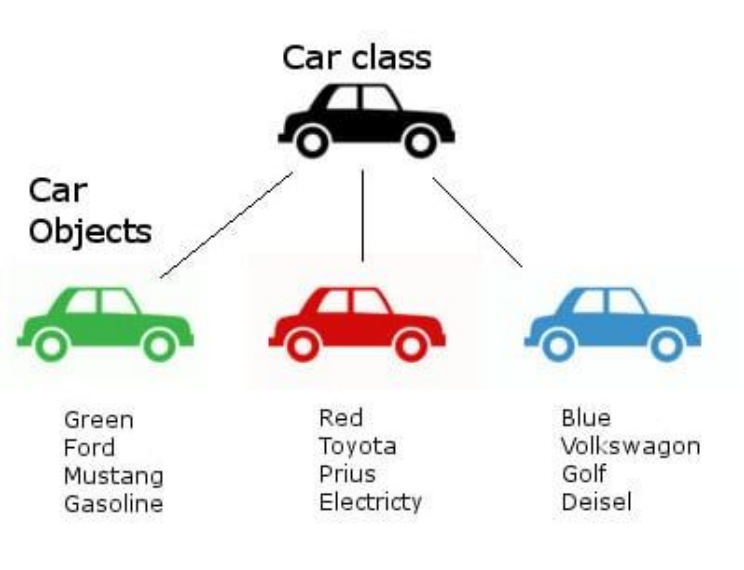



Image Source: https://breatheco.de/en/lesson/object-oriented-programming/

# Anatomy of a Class

```
class <Name> {

    <propertyName>: <propertyType>;
    <propertyName>: <propertyType>;

    constructor (<parameters>) {
        // do something
    }

}
```

**Properties** are pieces of data associated with each object. Every object of the class will have the same properties and default values.

A **constructor** is used when we make a new object to give properties initial values. You can think of a constructor as a special kind of “function” that returns an object of the class.

# Syntax for Working with Objects

```
class Book{
    name: string = "";
    pageCount: number = 0;
}
```

```
let twilight = new Book();

twilight.name = "Twilight";
twilight.pageCount = 498;
```

- The “new” keyword is always used when constructing a new object. If the class has a constructor with any parameters, we will need to provide values inside the parentheses.
- We use the dot operator to access an object’s properties or execute its methods.

# Hands-On #3: Code Writing

- Define a class called `UNCBathroom` with the following specifications:
  - Each `UNCBathroom` has a string property called `name`, and two number properties called `location` and `cleanliness`
  - The constructor should take in parameters and set values for these properties
- Define a function called `score()` that takes in a `UNCBathroom` object and returns the average of `location` and `cleanliness`
- In the `main` function, create a `UNCBathroom` object representing your favorite/least favorite bathroom and print its score

# Hands-On #3

```
class UNCBathroom {
    name: string = "";
    location: number = 0;
    cleanliness: number = 0;

    constructor(name: string, location: number, cleanliness: number) {
        this.name = name;
        this.location = location;
        this.cleanliness = cleanliness;
    }
}

let score = (bathroom: UNCBathroom): number => {
    return (bathroom.location + bathroom.cleanliness) / 2;
};

let main = async () => {
    let davis = new UNCBathroom("Davis Library", 6, -1);
    print(score(davis));
};
```