

Meat & Potatoes: Objects & Functions (OOP)

Lesson 3: Objects, Functions

Brought to you by the University of Maryland Balloon
Payload Program

Recap

- Arrays
 - Still need to talk about 2D Arrays!
- Loops
 - For loops, while loops
- If statements
 - If, else if, else statements

Data Types

- Integer: 1, 2, 3, 4, 5, ...
- Float: 1.5, 20.2, 100.12354, 3.1415, ...
- Character: 'a', 'b', 'c', ...
- String: 'Hello World', 'abcdefg', ...
- Boolean: True, False

Making our own “Data Type Combinations” Using **Objects**

An ‘**object**’ is a **collection of information** that we give a **label**.

For example, we can make a **class** of **objects** called “**Dog**”. Each instance of a dog will have a name, age, and breed.



Making our own “Data Type Combinations” Using **Objects**

An ‘**object**’ is a collection of information that we give a **label**.

For example, we can make a **class** of **objects** called “**Dog**”. Each instance of a dog will have a name, age, and breed.

Then, we can create a “**Dog**”, for example, “**Dog1**”, with some **data** attached to it:

- Name: “Spot” *[string]*
- Age: 11 *[integer]*
- Breed: “Golden Retriever” *[string]*

So if someone asks us about parts of Dog1, all of this info is grouped together. It is separate from Dog2, or Dog3, etc.

Making our own “Data Type Combinations” Using **Objects**

We can think of an ‘object’ as a group of variables in a labeled bin. Another classic example is a customer on a website:

A **customer** has:

- Email
- Password
- Age
- Phone Number
- Address

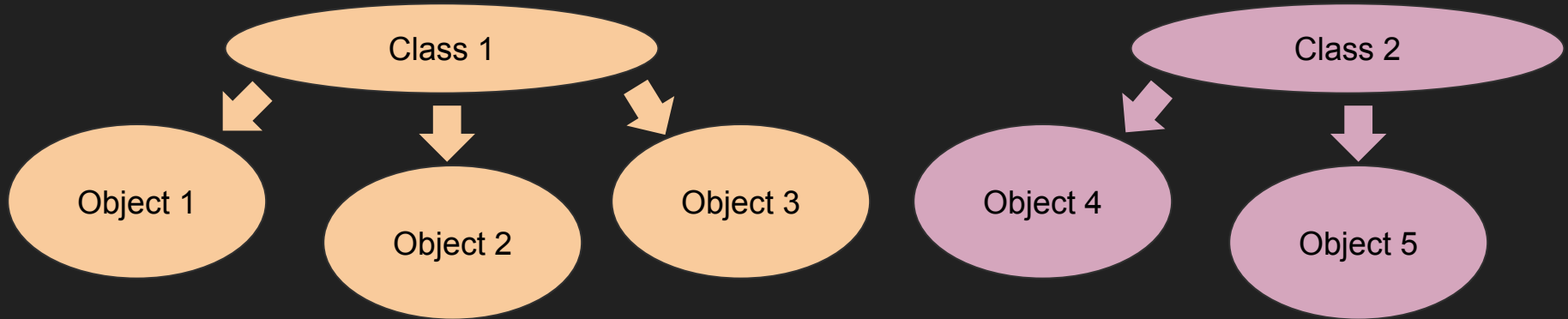
Customer 112345 has:

- Email: “customer1@gmail.com”
- Password: “password123”
- Age: 24
- Phone #: “3012469980”
- Address: “554 Normweller Road,
College Park, MD, 20872”

Objects vs. Classes

“Dog” is a **class**, like a blueprint, but an instance of that class, “Dog1” is an **object**

We can have many **Dogs**, but only one **Dog1** → **Dog1** is a **Dog**



Using Object & Their Data

We can ask for `Dog1.age`, `Dog1.name`, and `Dog1.breed` to get the information (variables) stored inside the object:

```
print(Dog1.age)
```

Creating an Object Syntax

Syntax:

```
class classname:
```

```
    def __init__(self, var1 = default, var2 = default, ...):
```

```
        self.variable1 = var1
```

```
        self.variable2 = var2
```

```
object1 = classname(value1, value2, ...)
```

Code that Checks your ID

Let's make a "Drivers_License" object, and write a program that checks if you can buy alcohol!

Hello (name)!

You are (age) years old and you (can/cannot) buy alcohol!

(If you are <21) You must wait (number) Years...

Then we can create an array of driver's licences, and go through them 1-by-1

Functions

A function is a piece of code **that takes in variables** (inputs) and **does something with them**.

Sometimes they return variables too (outputs)

```
def addition(num1, num2):
```

```
    num3 = num1 + num2
```

```
    return num3
```

Functions inside Objects

You can have a function that comes with an object. For example, maybe the Dog can “bark”.

```
def bark(self):  
    print(self.name + “says BARK!”)
```

```
dog1.bark()
```

Upcoming Topics

Lesson 1: Coding Basics & Logic

Lesson 2: Arrays, Loops, & Logic

Lesson 3: 2D Arrays, Objects, Functions

Lesson 4: Tkinter User Interfaces

Access this lesson and extra materials online!

Visit our Wiki Page:

<https://ter.ps/STEMworkshop>



Form for Email list

Coding on our computers

Visual Studio Code

- Where we type all our code
- code.visualstudio.com



Python

- Lets our computer run our written code
- python.org/downloads



What is Visual Studio Code (VSCode)?

A nicer way to write and edit code

- Technically, all code could be written in notepad
- VSCode has tools that make coding easier
 - Autocompletes variable names
 - Colors written code
 - User extensions for additional features
 - Highlights errors
 - More!

```
# Water (includes thermal scattering)  
water = openmc.Material(name='Water')
```

Functions

Define a function for addition

```
def add(a, b):  
    return a + b
```

Define a function for subtraction

```
def subtract(a, b):  
    return a - b
```

Define a function for multiplication

```
def multiply(a, b):  
    return a * b
```

Define a function for division

```
def divide(a, b):  
    if b == 0:  
        return "Error! Cannot divide by zero."  
    return a / b
```

Using the functions

```
x = 10  
y = 5
```

Call the functions and print the results

```
sum_result = add(x, y)  
print("Addition:", sum_result)
```

```
diff_result = subtract(x, y)  
print("Subtraction:", diff_result)
```

```
product_result = multiply(x, y)  
print("Multiplication:", product_result)
```

```
quotient_result = divide(x, y)  
print("Division:", quotient_result)
```