



Object-Oriented Fun

Educator's Guide

Overview 🙌

CS Hands-On is a 501(c)(3) nonprofit teaching computational thinking skills through technology-free lessons and activities. This curriculum is built to teach fundamental computer science concepts in an engaging, hands-on way. In this mission, students use object-oriented programming to play a card game.

Prerequisite Knowledge

Students should have completed the Conditional Schedule activity, which introduces the concept of if-then statements.

Lesson Details

At Decomposphere, students will learn to break problems down into smaller parts with Dot. Students will learn the fundamentals behind object-oriented programming using variables, functions, and classes. Then, students will play a Mystical Elements card game, where each card represents a card object with its own traits and attributes.

This lesson was developed for students ages 8 to 13, and can be modified for students of all skills and ages. This lesson takes around 30 minutes.

Learning Objectives

Key Question

How can we use object-oriented programming to create different objects?

Key Terms

Object-oriented programming: A programming model used to break down objects into their own unique variables and functions

Curriculum Standards

Students should be able to...

- Explain how object-oriented programming is used (Decomposition)
- Read, write, and interpret objects (Literacy)
- Use object-oriented programming to play a card game (Creative Arts)

[View standards addressed here](#)

Lesson Plan

Materials

- Object-Oriented Fun worksheet (per student)

Setup

- Hand out an Object-Oriented Fun worksheet to each student
- Set up your classroom to form students in groups of 2

ANSWER KEY & LESSON ANNOTATIONS

Name: _____ Date: _____

Object-Oriented Fun

Drivin' Around Decomposphere

Welcome back to the scenic landscape of Decomposphere! Get ready to follow Dot around in his Dot-mobile to learn about object-oriented programming.

Vroom Vroom...

We can use **object-oriented programming** to model objects using variables and functions. We can represent just about anything using variables and functions, from cars to dogs to houses!

In Decomposphere, Dot and his friends love driving around town and catching the summer breeze. Let's take a deeper look at object-oriented programming through Dot's car: the Dot-mobile!

Dot-mobile is our **object**



COLOR	SEATS	OWNER	} These are our object's (Dot-mobile's) variables
Green	4	Dot	
START	STOP		} These are our object's (Dot-mobile's) functions
Starts the engine when the gas pedal is pressed	Stops when the brakes are pressed		

Reflect

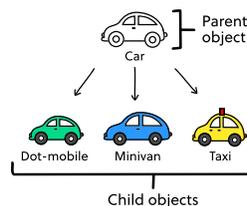
Using our Dot-mobile object, how can we change the properties to create our own Dot-mobile?

We can change the object's variables (color, seats, owner) and functions (start, stop) to create our own personalized Dot-mobile.

More cars!

On the right, we can see that the Dot-mobile, minivan, and taxi are all different types of cars.

Looking at the complete picture, Dot-mobile belongs to a large **group** of cars. Since all of these cars share **similar features**, we can create specific types of vehicles like the

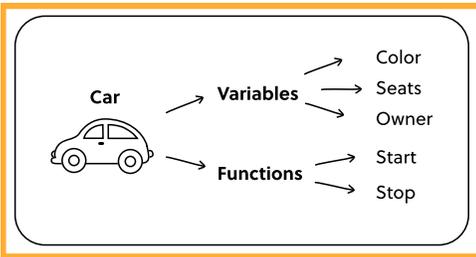




Dot-mobile, Minivan, and Taxi (these are called child objects) from the car (this is called the parent object).

We call the car our parent object because it **passes down shared variables and functions** to its child objects.

Let's take a closer look at the parent object:



Since all cars have a specific color, number of seats, owner, and can start and stop, **every child car** (the Dot-mobile, Minivan, and Taxi) **shares the same structure as the parent car**.

Reflect

Can you think of other variables and functions that a car would pass down to its child objects?
Ex. Variables: Number of windows, price, brand
Functions: Honk, wiping windows, turning on headlights

Why is Object-Oriented Programming useful?

Programs often have lots of code, which can become **messy and complicated**. When we have similar objects that share similar qualities but are not exactly the same, we can use inheritance (the passing down of features) in object-oriented programming to take the features of a parent object and apply them to its child objects. This template saves us time when creating child objects.

Referencing back to our car example, we can use our parent car object to create other cars like limousines or electric cars!

Educator's Note

Object-oriented programming can be used to describe many different scenarios and objects. Encourage your students to brainstorm different objects that this thinking can be applied to.



Let's Review Functions!

A **function** includes **actions** used to complete a task. Here's a quick refresher on how we can create our own functions:

Write the **steps** necessary to accomplish our function



Honk a Car Horn

Include a descriptive **title**

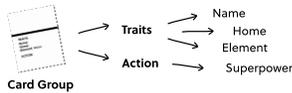
Press the button at the middle of the steering wheel

Mystical Elements: The Game

Use your knowledge of object-oriented programming to play a fun game of Mystical Elements with a friend! The first six cards each represent an object (a character from our six planets) with its unique variables and functions on your Card sheet. Each function is a superpower that relates to water, snow, or fire.

Setup

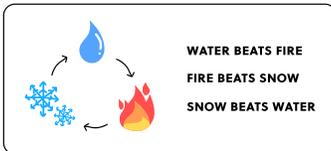
- Customize three different child objects on your Card sheet from the parent Card object. Assign your child objects variables (name and home) and a superpower function!



- Once you finish customizing your child object cards, carefully cut the cards out along dashed lines.

How to Play

- From your nine cards, pick one card to play and place it face down.
- Once both of you have placed your cards down, flip them face up and compare your object's **element** variable to decide the winner! Water puts out fire, fire melts snow, and snow freezes water.
- The player who wins the round takes both cards.
- The player with the most cards wins!



Ex. Water beats fire



MYSTICAL ELEMENTS CARDS

<p>VARIABLES Name = Dot Home = Decomposphere Element = Water</p> <p>FUNCTION Water Spiral Destroys fire with a tornado of water</p>	<p>VARIABLES Name = Ellis Home = Evaluatus Element = Water</p> <p>FUNCTION Whopping Waves Destroys fire with a tide of waves</p>	<p>VARIABLES Name = Lex Home = Logicland Element = Snow</p> <p>FUNCTION Icy Attack Freezes water into large ice crystals</p>
<p>VARIABLES Name = Alon Home = Abstractopia Element = Snow</p> <p>FUNCTION Snowstorm Freezes water with a gust of snow</p>	<p>VARIABLES Name = Ansel Home = Algorithmopoly Element = Fire</p> <p>FUNCTION Fireball Melts snow with a mighty ball of fire</p>	<p>VARIABLES Name = Pancho Home = Patteron Element = Fire</p> <p>FUNCTION Fire Frenzy Melts snow with a blazing fire</p>
<p>VARIABLES Name = _____ Home = _____ Element = _____</p> <p>FUNCTION _____</p>	<p>VARIABLES Name = _____ Home = _____ Element = _____</p> <p>FUNCTION _____</p>	<p>VARIABLES Name = _____ Home = _____ Element = _____</p> <p>FUNCTION _____</p>