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 roughly 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

Decomposphere
Mission 2

Name: _____ Date: _____

Object Oriented Fun

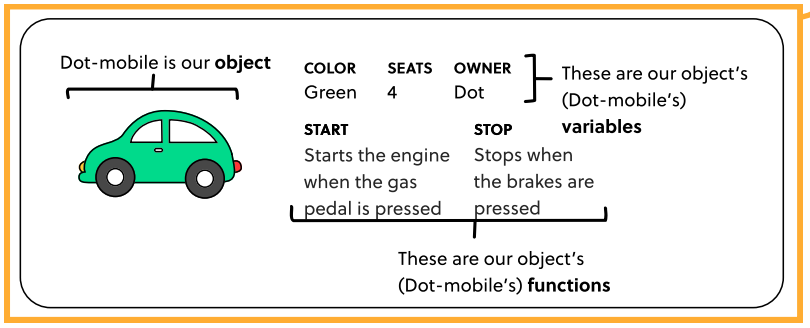
Welcome to Decomposphere!

Ready, set, learn! Dive into the grassy lands of Decomposphere with Dot and learn all about breaking down problems into smaller parts.

Vroom Vroom...

We can use **object-oriented programming** to break down objects into their unique variables and functions. We can make these objects can be just about anything, from cars to dogs to houses!

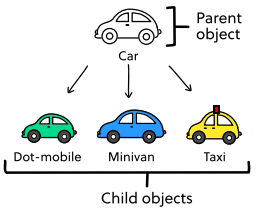
In Decomposphere, Dot and his friends love driving around town and catching the summer breeze. Let's take a deeper look at object oriented programming through Dot's car: the Dot-mobile!



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 **class** of cars. Since all of these cars share **similar properties**, we can create specific types of vehicles like the



CS Hands-On Object Oriented Fun 1

Reflect

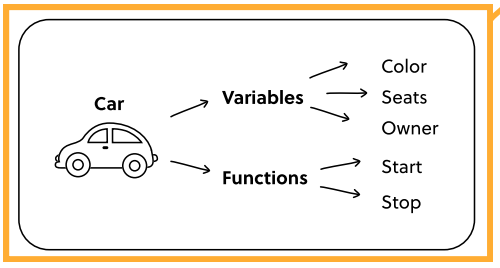
Using our Dot-mobile object, how can we change the properties to create our own Dot-mobile?

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

Dot-mobile, Minivan, and Taxi (called child objects) from our general car (called the parent object).

We call the general car our parent object, as it **passes down a set of commonly shared variables and functions** to its child object.

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) **can be modeled after the parent car**.

Why is Object-Oriented Programming useful?

Programs often contain lots of code, which can be **messy and complicated**. When we have similar objects that share similar qualities but are not precisely the same, we can use inheritance in object-oriented programming to take the properties of a parent object and apply them to its child objects.

Following our car example, we can use our parent car object to create other cars like limousines or electric vehicles.

Reflect

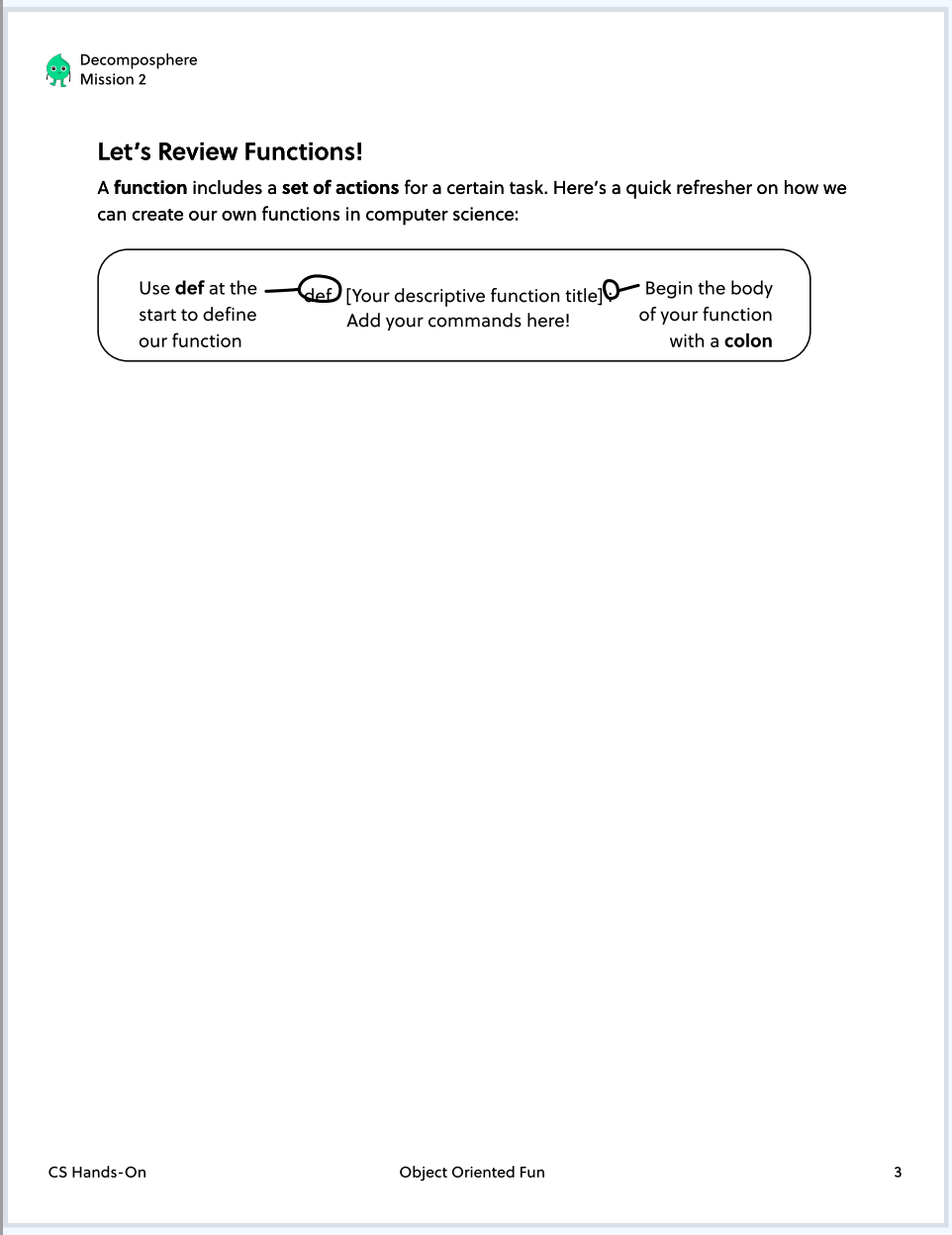
Can you think of other variables and functions that a car would pass down to its children?

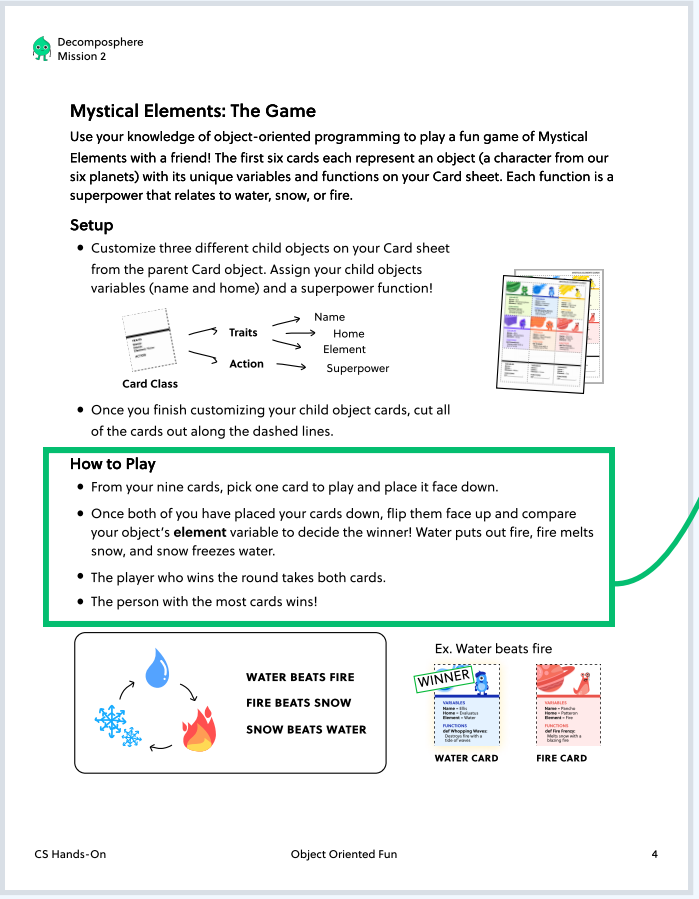
Ex. Variables: Number of windows, price, brand

Functions: Honk, wiping windows, turning on headlights

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 modeled after.


Decomposphere
Mission 2
Let's Review Functions!
A function includes a set of actions for a certain task. Here's a quick refresher on how we can create our own functions in computer science:
Use def at the start to define our function
def [Your descriptive function title]():
Add your commands here!
Begin the body of your function with a colon
CS Hands-On
Object Oriented Fun
3



Decomposphere
Mission 2

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!

Card Class → Traits → Name, Home, Element
Card Class → Action → Superpower

- Once you finish customizing your child object cards, cut all of the cards out along the 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 person with the most cards wins!

WATER BEATS FIRE
FIRE BEATS SNOW
SNOW BEATS WATER

Ex. Water beats fire

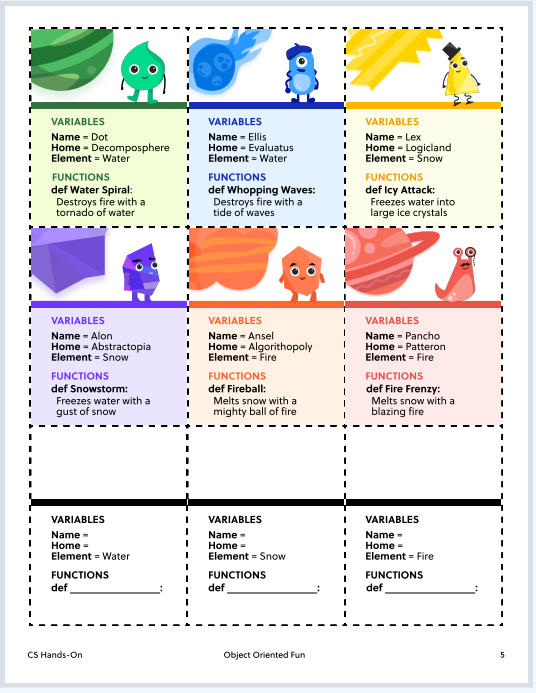
WINNER

WATER CARD FIRE CARD

CS Hands-On Object Oriented Fun 4

Extension

If students finish early, have them play more rounds with different variations of the game. For instance, students could shuffle the deck and continue to draw random cards to face off (similar to a game of War) or swap decks and play with their partner's customized cards.



VARIABLES
Name = Dot
Home = Decomposphere
Element = Water
FUNCTIONS
def Water Spiral: Destroys fire with a tornado of water

VARIABLES
Name = Ellis
Home = Evaluatus
Element = Water
FUNCTIONS
def Whopping Waves: Destroys fire with a tide of waves

VARIABLES
Name = Lex
Home = Logicland
Element = Snow
FUNCTIONS
def Icy Attack: Freezes water into large ice crystals

VARIABLES
Name = Alon
Home = Abstractopia
Element = Snow
FUNCTIONS
def Snowstorm: Freezes water with a gust of snow

VARIABLES
Name = Ansel
Home = Algorithopoly
Element = Fire
FUNCTIONS
def Fireball: Melts snow with a mighty ball of fire

VARIABLES
Name = Pancho
Home = Patteron
Element = Fire
FUNCTIONS
def Fire Frenzy: Melts snow with a blazing fire

VARIABLES
Name =
Home =
Element = Water
FUNCTIONS
def _____:

VARIABLES
Name =
Home =
Element = Snow
FUNCTIONS
def _____:

VARIABLES
Name =
Home =
Element = Fire
FUNCTIONS
def _____:

CS Hands-On Object Oriented Fun 5

Wrap up & reflect

Group students into pairs and have them discuss the following reflection questions. Afterwards, have students share their ideas as a class.

- Can you think of another parent object that we could model using object-oriented programming? What properties would it pass down to its child objects?

  Ex. Parent object: House
  Child objects: Specific houses
  Properties to pass down: Number of bedrooms, Number of bathrooms, Location, Price, Color

- Imagine you were tasked to create objects for every cat on the planet. How would using parent and child objects help in this scenario?

  Ex. Using a parent object as a template for the child objects would speed up the process of creating these objects. As we can pass down properties to child objects (Name, Breed, Number of paws, Meow, Play), we don't have to write out each variable or function for every single cat!