Ozaria

Chapter 4

Module 3: Writing Functions

Before you teach. . .

You may want to prepare the following before you implement this module in your classroom:

- ❑ **Make a copy** of the slides if you'd like to revise or adapt them for your classroom
- ❑ Use the teacher dashboard to check student progress.
- ❑ Review **Concept Check** activities
- ❑ Navigate to **Independent Practice** slides to:
  - ❑ **Unlock specific modules** and levels as necessary
  - ❑ **Review extension activities** and have activity links ready to share
- ❑ Try out some **practice levels** on your own
- ❑ Review the **solution guide**

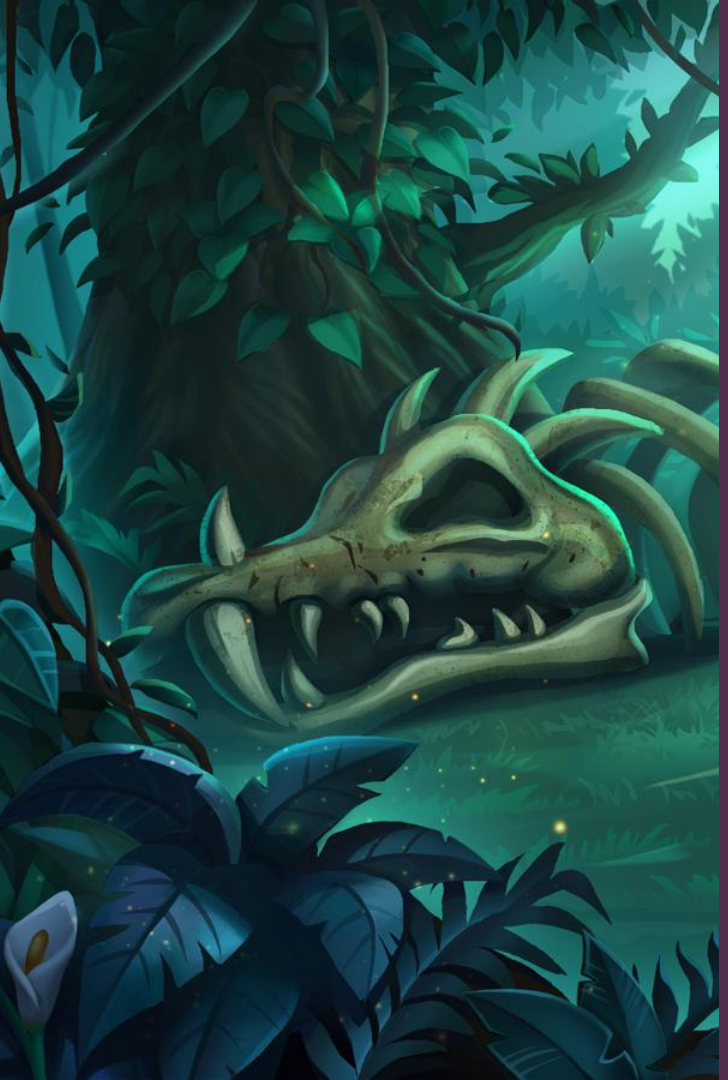
How to teach a lesson. . .

- Use the **first section** of the lesson slides (up to Independent Practice slide) to introduce the new concept for that lesson
- If applicable, watch any **Cutscene** videos as a whole class to introduce the story line
- **Extension Activities** are found in the **speaker notes** of the I**ndependent Practice slide**. You can use them to differentiate pacing for fast finishers or for students who need reinforcement.
- Have students independently play through the suggested levels on the **Independent Practice slide**
  - This section which will also include Cinematics and Concept Checks
- Use the **Check-In** section as an opportunity for students to reflect on what they have learned in the lesson

Table of Contents

Lessons

Additional Resources



Lesson 1

Writing Functions

What did we learn last time?

-How did you use **functions** in Ozaria?

-What is the difference between a **function definition** and a **function call?**

-How do you use **parameters** to customize your function?

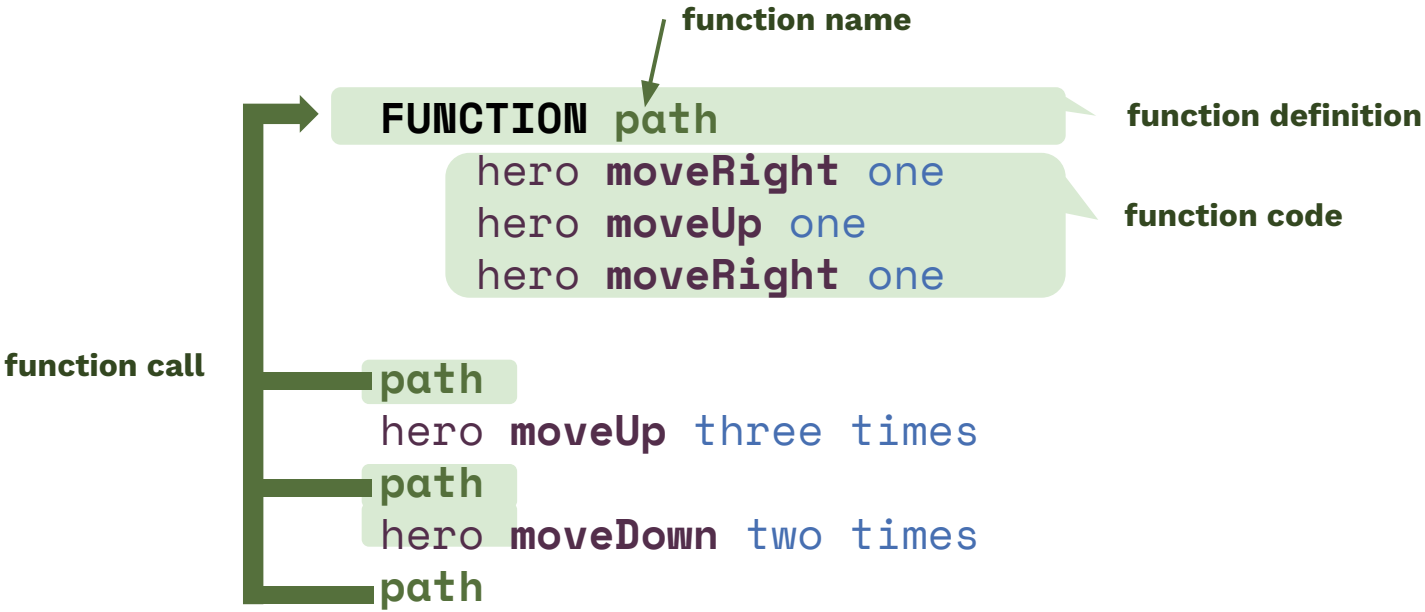
-How did you create an **acodus in the past**?

Today's Journey

- Review: **Functions Review & Syntax**
- Warm Up: **Functions with Mouse**
- Introduction: **Functions with Parameters**
- **Play Ozaria**: Use the Acodus to fight the Darkness
- **Optional Extension**: Design a Noodles Map

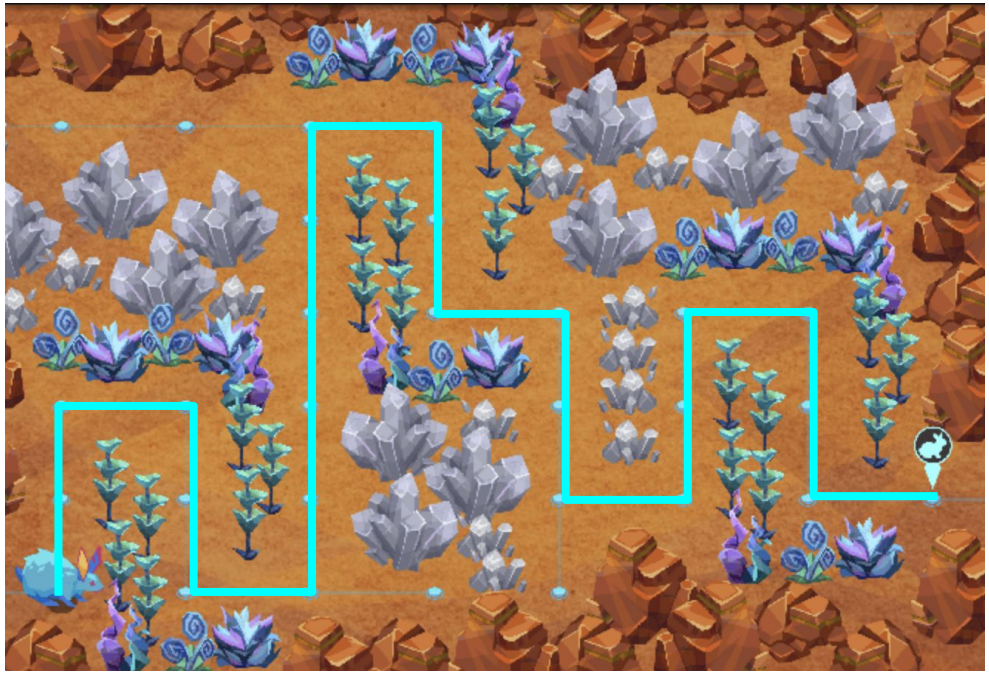
Function Review

A **function** contains a piece of code that you can use over and over again in your program. It provides a shortcut that helps your code stay organized and easier to read.



Writing Functions

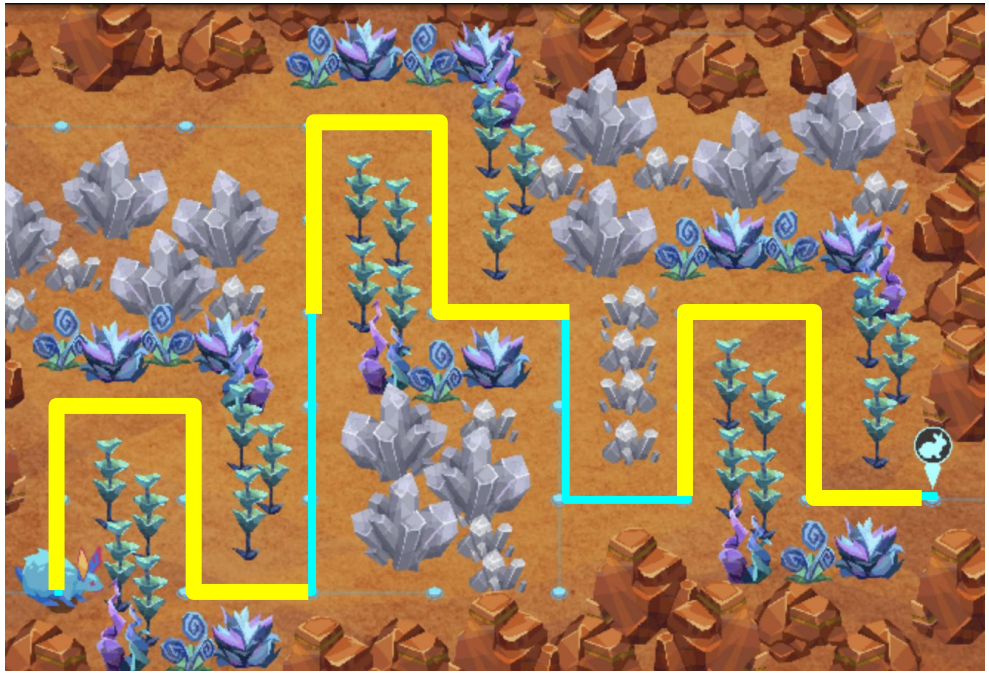
How do you decide what code should go inside your function?



- Trace the path that you need to take
- Identify any repeated movement patterns in the path

Writing Functions

The **repeated movement pattern** is the code that should go inside the function.



```
FUNCTION path
    noodles moveUp two
    noodles moveRight one
    noodles moveDown two
    noodles moveRight one
```

```
path
noodles moveUp three
path
noodles moveDown two
noodles moveRight one
path
```

Function Syntax: Python

Things to remember when defining and writing functions.

```
def crossRiver():
    hero.jumpRight()
    mouse.moveRight(2)

crossRiver()
hero.moveUp(2)
crossRiver()
```

- `def` - used to start the definition of the function
- `crossRiver` - choose a name for your function that will help you remember what it can do
- `()` - make sure you put () after the name of your function when you define it
- `:` - the definition line needs to end with a colon
- make sure your function code is indented

Function Syntax: JavaScript

Things to remember when defining and writing functions.

```
function crossRiver(){
    hero.jumpRight();
    mouse.moveRight(2);
}
crossRiver();
hero.moveUp(2);
crossRiver();
```

- `function` - used to start the definition of the function
- `crossRiver` - choose a name for your function that will help you remember what it can do
- `()` - make sure you put () after the name of your function when you define it
- `{}` - the function code needs to start and end with brackets (just like loops & conditionals)
- make sure your function code is indented

Concept Check: Functions with Mouse

Define and write the ladder() function to help Mouse get to the exit.

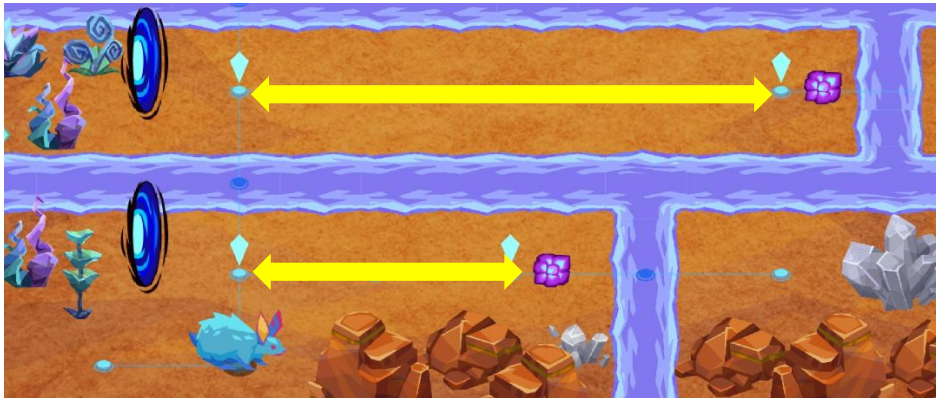
```
# define and write the ladder function here

ladder()
mouse.moveRight()
ladder()
mouse.moveUp()
```

Independent Practice

Functions with Parameters

We have used **parameters** to customize functions so that they can be used in different ways.

define a parameter when defining a function and you can use it **like a variable** in the function code



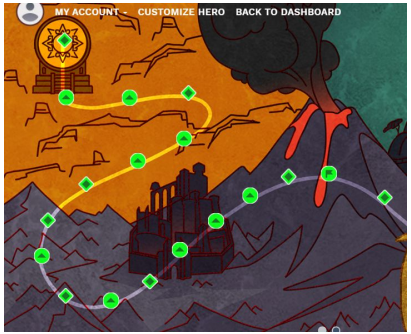
```
FUNCTION deliver(steps)
    noodles moveRight steps
    noodles take
    noodles moveLeft steps
    noodles drop
```

```
noodles moveUp one
deliver(2)
noodles jumpUp one
deliver(4)
```

Use an **argument** to set the **parameter value** when you **call the function**

Independent Practice

Play Ozaria
Start with **Cinematic: Leading the Charge** and stop after you finish **Cinematic: Restored**



Extension Activity: Design a Noodles Map
Use the link provided by your teacher to complete this activity.

Need Help?
Always try it once & try using the troubleshooting guide first. If you still need help after that, then reach out to a classmate or the teacher.

Check-In

What did you discover?

Chapter 4, Module 3, Lesson 1

What I did:

-How far did you get in Ozaria? What was your favorite part?

What I noticed:

-What did you notice as you played through the game?
-Did you discover or learn something new?

What was challenging?

-Was something confusing about the story or the levels?
-Do you need more practice on a specific part of the code or the game?

Resources

Standards Alignment &
Additional Resources

CSTA Standards Alignment

Algorithms & Programming

-1A-AP-10: Develop programs with sequences and simple loops, to express ideas or address a problem.
-1A-AP-11: Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
-1A-AP-14: Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
-1A-AP-15: Using correct terminology, describe the steps taken and choices made during the iterative process of development.
-1B-AP-10: Create programs that include sequences, events, loops, and conditionals.
-1B-AP-11: Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
-1B-AP-12: Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
-1B-AP-15: Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
-2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
-2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
-2-AP-17: Systematically test and refine programs using a range of test cases.

Common Core State Standards

Writing

-CCSS.ELA-LITERACY.W.6.6: Use technology, including the Internet, to produce and publish writing as well as to interact and collaborate with others; demonstrate sufficient command of keyboarding skills to type a minimum of three pages in a single sitting.

-CCSS.ELA-LITERACY.W.6.10/CCSS.ELA-LITERACY.W.7.10/CCSS.ELA-LITERACY.W.8.10: Write routinely over extended time frames (time for research, reflection, and revision) and shorter time frames (a single sitting or a day or two) for a range of discipline-specific tasks, purposes, and audiences.

Common Core State Standards

Math: Expressions & Equations

-CCSS.MATH.CONTENT.6.EE.B.6: Use variables to represent numbers and write expressions when solving a real-world or mathematical problem; understand that a variable can represent an unknown number, or, depending on the purpose at hand, any number in a specified set.

-CCSS.MATH.CONTENT.7.EE.B.4: Use variables to represent quantities in a real-world or mathematical problem, and construct simple equations and inequalities to solve problems by reasoning about the quantities.

Resource Links

Teacher Materials
-Chapter 4 Solution Guide - Python
-Chapter 4 Solution Guide - JavaScript

Student Materials
-Check-in Journal Template
-Lesson 1 Extension Activity: Design a Noodles Map
-Troubleshooting Guide