Educator Guide

Python Island 5

45 minutes

Single Student

Using Functions in Python.

Causeway Digital

THEME OVERVIEW

Welcome to the Kingdom of the Floating Islands. This mystical kingdom floats high above the world below. Throughout this collection of lessons, students will learn the basics of Python, through a series of themed tasks in each world. Completing these tasks will allow the kingdom to develop further technologically.

LESSON OBJECTIVES

- Understand the purpose of functions within Python.
- Become familiar with using parameters within functions to pass in information.
- Become familiar with using return statements to return back information to the parent code.
- Understand the basics of events and their use in Minecraft.
- Continue to develop an understanding of the concept of **Decomposition,** as students break problems down into codable solutions for the Agent.
- Continue to see the importance of **Sequencing**, putting things in order, so that the Agent can complete its tasks.

THINGS TO KEEP IN MIND

- Students are given a whistle in the first slot of their hotbar. This allows students to teleport their Agent to them at any point.
- Remind students there may be more than one solution for each of the activities.

MINECRAFT MECHANICS

C	**C** Summons the Agent and opens the Notebooks interface.
T	**T** Opens chat panel in Minecraft for commands to be typed
ESC	**ESC** When a student wants to leave the game, leave chat, or pause the game.

PYTHON COMMANDS

`say("Message")`	**Say command** Output a message in chat
`agent.move("Direction")`	**Agent move** Tells the Agent to move in a certain direction 1 block.
`agent.place(1, "Direction")`	**Agent place** Tells the Agent place a block from inventory slot 1, in a certain direction.
`agent.destroy("Direction")`	**Agent destroy** Tells the agent to destroy/break a block in a certain direction.
`agent.till("Direction")`	**Agent till** Tells the agent to till a block of dirt ready for planting seeds, in a certain direction.
`agent.inspect("Direction")`	**Agent inspect** Get the name of the block in the direction the agent is checking.
`player.add_effect("levitation", 1, 10)`	**Player add effect** Adds a "potion" effect to a player (by default, yourself). In this case, the levitation effect for 1 seconds, with an amplifier of 10.

`if (logical expression):`
`  statement()`
`else:`
`  statement()`
	If / else statement
	Compare 2 values. If they are equal, run one piece of code. If they are not equal, run a different piece of code (else).

`for num in range(x, y):` `  statement()`	**For loop** Repeat the statement a fixed (known) number of times, between *x* and *y* variables.
`while (logical expression):` `  statement()`	**While loop** Repeat the statement a variable (unknown) number of times, while the logical expression is true.
`def function_name(parameter):` `  statement()`	**Function** A small program contained within the main program that can be called as many times as needed. Supports multiple parameters which allow data to be passed into the function dynamically.

KEYWORDS

Function – A small "mini-program" that can be contained within the main program and called as many times as needed.

Parameter – A variable that is used to pass information from the main program into a function at runtime.

Return statement – A mechanism to hand the result of the function, back out to the main program.

Event – Something that happens within the game, that can be used to trigger student code, for example *on_player_bounced* or *on_player_travelled*.

START OF LESSON PROCEDURE

Number of Activities: 4
Optional Activity: 0

INTRODUCTION AND LEAD-IN: 5 minutes

Introduction:

Welcome to Python Islands, a place to learn all about the basics of the Python programming language.
In Island 5, you will be focusing on using Functions. Functions are mini programs, that you can then run as many times as you want, all from within your program.

Lead-in:

Explain to students, that this island builds upon the previous work they have done in Island 4. Once again, their agent will be returning, although in this lesson they will be focusing on building out their own functions. Their agent will help them complete a series of tasks that they wouldn't be able to otherwise complete.

CODING ACTIVITIES: 30-45 minutes

Activity 1: (Supporting towers)

On entering the world, players are met by the Prime Minister, who asks if they can help the town out with its most recent endeavour, building a radio telescope!
On arrival at the telescope island, they are met by the Telescope Engineer, who explains the task at hand involves creating a Function to build the supporting towers for the telescope. On successful completion of the code in the final code box, the three towers will take a bit of time to be built by the Agent.

If the student needs to reset the tower, they can speak to the telescope engineer again and click “Reset Test Tower” to try again.

Activity 1 Final Solution

```
# Final code
# Must be run once
# Note may take a few mins

def build_tower():
    for height in range(0, 15):
        agent.move("up")
        draw_square(3)
```

Activity 2: (Light translator)

Next, players are asked to visit the Centre for Computer Research. On arrival, the Computer Technician explains that their new computer is up and running, but the decoder system isn’t ready yet. The computer is needed, to be able to calculate the ideal angle for the new telescope to be pointing.
In the meantime, the technician asks if the students can help create their own decoder function, using their agent, to decode the output from the computer.

Activity 2 Final Solution

```
# Final code
# Must be run once

def decode(block):
    if block == "a":
        return 1
    elif block == "b":
        return 2
    elif block == "c":
        return 3
    elif block == "d":
        return 4
```

Activity 3: (Bouncing)

With the required angle now calculated, the telescope technician can start dialling in the settings. In the meantime though, she still requires the final receiver module to be placed on the telescope. To bring the beacon item up to the required location, students must create a program allowing them to jump higher than usual.

This involves hooking into Minecraft events, specifically the *on_player_bounced* event. Once the code is completed, the students can then bounce their way up to the top of the telescope, following the pathway marked out in the Notebook.
Do note the diagram at the end of the Notebook. It shows the recommended pathway of jumps to get up to the top of the telescope.

Activity 3 Final Solution

```
# Final code
# Must be run once

def on_player_bounced(height, block):
    if block == "slime":
        say("I'm jumping on slime!")
        player.add_effect("levitation", 1, 10)
```

To finish the task, add the beacon from the inventory to the black block on the end of the telescope's antennae.

Activity 4: (Power linkup)

With the telescope fully constructed and ready to go, all that is left is connecting to the power, from the power station on the main island. After speaking with the engineer in the power station, students must run a (redstone) cable from the power station to the Telescope Electrical Input building.
This is achieved by using the agent to lay the cable, while using the *on_player_travelled* event to have the agent place the cable behind the player as they walk.

Activity 4 Final Solution

```
# Final code
# Must be run once

def on_player_travelled(location, mode, distance):
    loc = correct_location(location)
    agent.teleport(loc)
    agent.place(1, "down")
    # Add your code to the line above
```

Completion:

Once the telescope has been hooked up, the students are encouraged to speak with the Prime Minister in the parliament building. This completes the world.

LESSON CONCLUSION: 5 minutes

Upon completion of this lesson students should be able to answer the following questions:

1. What is a function in Python?

   Answer: A function is a mini Python program, contained within the main program. It allows a collection of commands to be run at any time from the main program.
2. How can you pass in data/variables to these functions?

   Answer: This is achieved by using parameters.
3. What are common use cases for functions?

   Answer: To reduce code reptition (copy and pasting), reduced code complexity by splitting it up into more manageable and defined sections.

CSTA STANDARDS

The following Computer Science Teachers Association K-12 Computer Science Standards (2017) are covered by this lesson.

Identifier	Standard
1A-AP-10	Develop programs with sequences and simple loops, to express ideas or address a problem.
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.
2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.