MINECRAFT
EDUCATION EDITION

Educator Guide

Minecraft Python Course – Lesson 4
45 minutes

Animals Are Friends

LISTS AND METHODS

EDUCATION.MINECRAFT.NET

LESSON OBJECTIVES:

In this lesson the students will learn:

- the concept of lists.
- how to create lists.
- how to use methods with lists.
- how to modify and sort lists.
- zero-based numbering.
- how to use underscores when creating variable and list names.
- how to spawn mobs.

LESSON INTRODUCTION: 10 minutes

Theme:

Tell the students that they need to help CodingMine with the development of their new software that is going to be used by an animal hospital. Veterinarians need help to easily identify and categorize animals, as well as prescribing different treatments and dietary requirements. The students will help in the development of this project by using **lists.**

Coding Concepts:

List

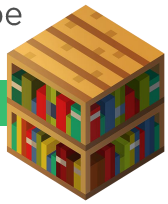
A list is an ordered collection of strings, numerical values or variables. In Minecraft Python a list can also be a collection of Minecraft items or mobs. A list's content can be edited by adding, removing or ordering it in a different way. Lists are not unlike variables/boxes; however in lists we can place more than one value. When naming lists the same rules apply as when naming variables. A list name can be named anything, however, it must have at least one letter and no spaces. Rather than using spaces it is common practice to use **underscores**. Examples:

Normal text: Animal list: elephant, lion, zebra, giraffe.

Python syntax: `animal_list = [elephant, lion, zebra, giraffe]`

Zero-based numbering

Tell the students that when they normally count, they start from 1 and continue 2, 3, 4... However, in computer science, and by extension in Python, most counting begins from 0 and continues 1, 2, 3. Tell the students that they will be



using this concept when coding with lists in this lesson, as when we want to access the second value in a list they will need to refer to it by the numerical value **1** e.g. `my_list[1]`. This is because the first value would be referred to by the numerical value **0** e.g. `my_list[0]`.

Methods

Tell the students that they will be using methods in this lesson to control the contents and the arrangement of content in lists. Tell the students that methods are special commands that affect data in many ways. Tell them that in this lesson they will be using different methods to update lists:

1. `my_list.sort()` – Sorts a list in alphabetical order.
2. `my_list.reverse()` – Reverses the order of the contents.
3. `my_list.append()` – Adds another value to the end of a list.
4. `my_list.pop()` – Removes a value from a list.

User Interface:

In this lesson the students will:

- use the button **Q** on the keyboard to drop items that they are holding in the hot bar.
- see displayed world coordinates in the top left-hand corner.

Syntax / Operators:

Tell the students in this lesson they will learn about:

Square brackets [ ]:

Square brackets are used when creating lists. When we want to define a list's contents they need to be placed within a pair of square brackets and separated by commas. Example: `name_list = [John, Mary, Chad, Isabella]`

Underscore _ :

When creating names for lists or variables there are a few guidelines on what is possible and what is not. One of the things that a list or variable name cannot contain is any spaces. So, if a space is needed the most common thing to use instead is an **underscore**. Examples:

```
my_list = [0, 1, 2, 3, 4]
important_variable = 1
```

CODING ACTIVITIES: 30 minutes

Activity 1: Animal categorizing.

Objective: Explain to the students that the developer needs their help to make a list of animals and then spawn the different animals' types in their correct pens, at predefined locations.

Tell the students to start by making a list of the animals going from left to right that matches the list in the Minecraft world. *(Hint: The students should name their list* `My_list`*).* When the students run the code correctly all the animals will spawn in their correct pens, and Activity 1 is complete.

Code snippet:

Before:

```
location1 = world(-2, 40, -11)
location2 = world(-2, 40, -5)
location3 = world(-8, 40, -0)
location4 = world(-13, 40, -5)
location5 = world(-13, 40, -11)
# Replace the lines below with your code #

# list of animals

mobs.spawn(My_list[0], location1)
#spawn the third mob from the list at location2
#spawn the fifth mob from the list at location3
#spawn the second mob from the list at location4
#spawn the fourth mob from the list at location5
```

After:

```
location1 = world(-2, 40, -11)
location2 = world(-2, 40, -5)
location3 = world(-8, 40, -0)
location4 = world(-13, 40, -5)
location5 = world(-13, 40, -11)

My_list= [COW, PIG, SHEEP, HORSE, RABBIT]

mobs.spawn(My_list[0], location1)
mobs.spawn(My_list[2], location2)
mobs.spawn(My_list[4], location3)
mobs.spawn(My_list[1], location4)
mobs.spawn(My_list[3], location5)
```

Activity 2: Dietary requirement.

Objective: Explain to the students that the veterinarian needs their help to write some code that will control a food making machine to make food for three different dogs with diverse dietary requirements The first dog will need everything that is already in the list. The second dog will need additional vitamins and the third dog will need the beef removed from the list. Explain to the students that they will need to use two methods to manipulate the list of foods, **append** and **pop**. Tell the students that to give the dogs their food they will need to drop it in their bowls.

Each time that the code is run the food will fall into the chest *(Hint: to drop the food into the bowl tell them to use the key [Q] on their keyboard). (Hint: to remove or add any of the items in the list use the given variables).* Once the students have fed all the dogs, Activity 2 is complete.

Code snippets:

Before:

```
Bone = world(-21, 45, -31)
Beef = world(-21, 45, -29)
Chicken = world(-21, 45, -27)
Biscuit = world(-21, 45, -25)
Vitamins = world(-21, 45, -23)
# Replace the lines below with your code #
Dog_Food=[Bone, Beef, Chicken, Biscuit]
# Add the variable Vitamins to the list using the append method | Step 2
# Remove the variable Beef using the pop method                | Step 3

blocks.place(REDSTONE_BLOCK, Dog_Food[0])
# Change the numerical value of the list below         | Step 1
blocks.place(REDSTONE_BLOCK, Dog_Food[0])
# Change the numerical value of the list below         | Step 1
blocks.place(REDSTONE_BLOCK, Dog_Food[0])
# Change the numerical value of the list below         | Step 1
blocks.place(REDSTONE_BLOCK, Dog_Food[0])
# Change the value of the list below                           | Step 2
blocks.place(REDSTONE_BLOCK, Dog_Food[0])
```

After:

```
Bone = world(-21, 45, -31)
Beef = world(-21, 45, -29)
Chicken = world(-21, 45, -27)
Biscuit = world(-21, 45, -25)
Vitamins = world(-21, 45, -23)
# Replace the lines below with your code #
Dog_Food=[Bone, Beef, Chicken, Biscuit]
# Add the variable Vitamins to the list using the append method | Step 2
# Remove the variable Beef using the pop method                | Step 3

blocks.place(REDSTONE_BLOCK, Dog_Food[0])
blocks.place(REDSTONE_BLOCK, Dog_Food[1])
blocks.place(REDSTONE_BLOCK, Dog_Food[2])
blocks.place(REDSTONE_BLOCK, Dog_Food[3])
# Change the value of the list below                           | Step 2
blocks.place(REDSTONE_BLOCK, Dog_Food[0])
```

```
Bone = world(-21, 45, -31)
Beef = world(-21, 45, -29)
Chicken = world(-21, 45, -27)
Biscuit = world(-21, 45, -25)
Vitamins = world(-21, 45, -23)
# Replace the lines below with your code #
Dog_Food=[Bone, Beef, Chicken, Biscuit]
Dog_Food.append(Vitamins)
#Remove the variable Beef using the pop method                | Step 3

blocks.place(REDSTONE_BLOCK, Dog_Food[0])
blocks.place(REDSTONE_BLOCK, Dog_Food[1])
blocks.place(REDSTONE_BLOCK, Dog_Food[2])
blocks.place(REDSTONE_BLOCK, Dog_Food[3])
blocks.place(REDSTONE_BLOCK, Dog_Food[4])
```

```
Bone = world(-21, 45, -31)
Beef = world(-21, 45, -29)
Chicken = world(-21, 45, -27)
Biscuit = world(-21, 45, -25)
Vitamins = world(-21, 45, -23)
# Replace the lines below with your code #
Dog_Food=[Bone, Beef, Chicken, Biscuit]
Dog_Food.append(Vitamins)
Dog_Food.pop(1)

blocks.place(REDSTONE_BLOCK, Dog_Food[0])
blocks.place(REDSTONE_BLOCK, Dog_Food[1])
blocks.place(REDSTONE_BLOCK, Dog_Food[2])
blocks.place(REDSTONE_BLOCK, Dog_Food[3])
blocks.place(REDSTONE_BLOCK, Dog_Food[4])
```

Activity 3: What is the cat's name?

Objective: Explain to the students that the data scientist needs their help to write a few pieces of code that will use the **sort** and **reverse** methods to make changes to a list of cat names that they will be given. The first code has to change the last cats name to shadow. The second code has to sort the cat names alphabetically. The third code has to reverse all of the cat names in the list. The students will then have to create an code to write out a specific cat

name in the chat each time, by changing the positional value in the given say command. Then they will have to select that cat from the line-up.

Each time that the code is run a name of a cat will appear in the chat. To select a cat tell them to press the button, **select a cat**, underneath the monitor and then to choose that cat from the line-up by pressing the button underneath the correct cat. When the students have selected all of the cats, Activity 3 and the lesson are complete.

Code snippets:

Before:

```
Cat_Names= ["Smokey", "Oreo", "Sammy", "Patch", "Princess", "Snowy"]
# Replace the lines below with your code #
#Change the last name to shadow           | Step 1
#Sort the names alphabetically in the list        | Step 2
#Reverse all names                                      | Step 3

# Change the value of the list below | Step 1,2,3
player.say(Cat_Names[])
```

After:

```
Cat_Names= ["Smokey", "Oreo", "Sammy", "Patch", "Princess", "Shadow"]
# Replace the lines below with your code #
Cat_Names[5] = "Shadow"
#Sort the names alphabetically in the list        | Step 2
#Reverse all names                                      | Step 3

# Change the value of the list below | Step 2,3
player.say(Cat_Names[5])
```

```
Cat_Names= ["Smokey", "Oreo", "Sammy", "Patch", "Princess", "Shadow"]
# Replace the lines below with your code #
Cat_Names[5] = "Shadow"
Cat_Names.sort()
#Reverse all names                                       | Step 3

# Change the value of the list below | Step 3
player.say(Cat_Names[3])
```

```
Cat_Names= ["Smokey", "Oreo", "Sammy", "Patch", "Princess", "Shadow"]

Cat_Names[5] = "Shadow"
Cat_Names.sort()
Cat_Names.reverse()

player.say(Cat_Names[3])
```

LESSON CONCLUSION: 5 minutes

Ask the students about the skills that they have learned during the lesson, to reinforce the concepts.

1. Q. What is a list?
   A. A list is a collection of strings, numerical values, variables, mobs or items.
2. Q. What do we use instead of spaces when naming a list or variable?
   A. Underscore.
3 Q. What syntax do you place around a list's content?
   A. Square brackets.
4. Q. What four methods can we use with lists?
   A **Reverse, append, sort, pop.**

EDUCATION STANDARDS:

CSTA K-12	
3A-AP-14	Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
ISTE	
7A	Provide alternative ways for students to demonstrate competency and reflect on their learning using technology.
3B	Establish a learning culture that promotes curiosity and critical examination of online resources and fosters digital literacy and media fluency.
6B	Manage the use of technology and student learning strategies in digital platforms, virtual environments, hands-on makerspaces or in the field.

COMMANDS:

Say

```
player.say(message)
```

Description: Shows something in the chat, in the game.

message parameter: The message that the player wants to display in the chat. This message can be either a piece of text (string) or a mathematical value.

World position

```
world(0, 0, 0)
```

Description: Creates a new world position.

x: The east (+x) or west (-x) coordinate, in blocks.

y: The up (+y) or down (-y) coordinate, in blocks.

z: The south (+z) or north (-z) coordinate, in blocks.

spawn mob at position

```
mobs.spawn(My_list[0], location1)
```

mob parameter: What mob should be spawned.

destination parameter: At which position should the mob be spawned.

Place block at position

```
blocks.place(block, pos(0, 0, 0))
```

Description: Places a block at a certain position.

block parameter: Defines what block type should be placed.

position parameter: Defines the position where a block is placed.