

Activity 1.1.7

Traversing Turtles

Distance Learning Support

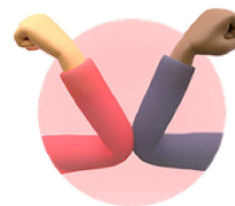
Check with your teacher about:

- ☐ What work you need to turn in and how to submit it
- ☐ Collaboration strategies
- ☐ If you are using a Chromebook, open a [blank code editor](#).



GOALS

- Use a list to represent a collection of data.
- Add and remove elements to and from a list.
- Use a `for` loop to iterate through all of the elements in a list.





Lists

The first five activities asked you to work with individual turtles and variables, but it's also useful to know how to group similar types of items together.

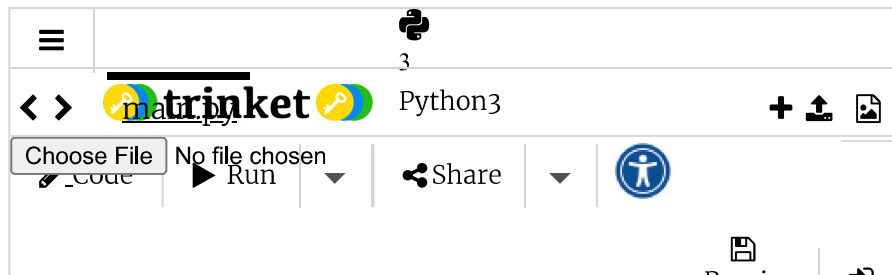
For example, if you were representing animals in code, you might want to group animals according to similar properties. Even different animals like dogs and cats could be grouped to capture their movement, temperament, eating, sleeping habits, or describe physical properties, such as fur color.

In *Python*®, we group data types with a construct known as a **list**. Some other programming languages refer to a similar data type as an **array**. Anything can be grouped together into a list—numbers, words, fruits, animals, colors, and even turtles can be made into a list.

In this activity, you will create and manage multiple turtles at the same time using a list.

List Method

- 1 Review the code in the code editor below. Can you guess what it does?



Note: When you run this program, `new_list` shows the individual list items in single quotes. In *Python*, you can use either single or double quotes for string **literals**. The code samples throughout the course will use double quotes.

2

Run the program. Did it do what you expected?

If you guessed that it was going to display a list in the *Python* terminal, you were correct. The first line of code created the list, and because you wanted the program to store the list, you need to create a variable to do it. In this case, the variable name for the list is named `new_list`. The square brackets contain all of the items in the list; each item is separated by a comma and is called an **element**. The second line of code prints out all of the **list element**s defined in the list.

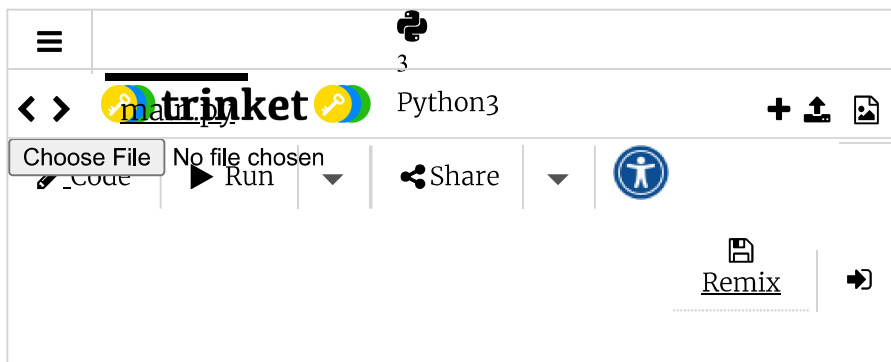
College Board Connection: Creating a List

What if you wanted your program to later add an element to the end of the list? You can do this by using the `append` method.



Append to the End of a List video (The video has no sound.)

- 3 Run the program to see how it adds `lion` to the end of the list.



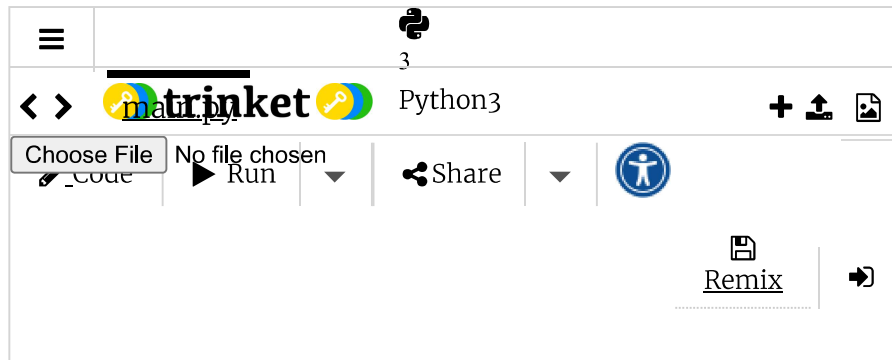
Note: The length of `new_list` is increased by 1, and the value is placed at the end of the list.

College Board Connection: APPEND to a List

What if you wanted to get the last item that was placed into the list and remove it, much like you would grab the top item out of a basket or bag? The `pop` method in *Python* removes the last element from a list and assigns it to a variable. Think of the `pop` method as “popping an item off” the list.

4


Run this program.



Note: Although this example assigns a variable, you do not have to do so when you `pop`. In other words, you can just use `new_list.pop()` by itself. This will remove the last item, but it won't be assigned anywhere.

The `for` Loop Revisited

Once you have a list, it's common that a program needs to repeat a task for each item in the list. To do this, you can use a `for` loop. Recall that the `for` loop is similar to the `while` loop as it will iterate through a sequence of code. However, the `for` loop is different in that it tells the program to run through a range as you have seen, or a list, and for each item in the list, complete a given task.

When used on lists, the `for` loop retrieves each element from the list and assigns it to the **loop variable** . Using the loop variable, you can have the program execute the same set of statements for each

element. In effect, the loop variable can be assigned to each element of a list on each interaction of the list. One example might be the sequence to print each element.

Figure 1. `for` Loop Syntax

5

Examine the code in the code editor. Any code that is indented directly beneath the `for` statement is called the body of the loop. It will be executed in each iteration of the loop. Notice how the `for` loop is written—with the loop variable assigned as `animal` and the code to be run during the loop indented after the `for` statement. What do you think this loop will do?




6

Run the program.



- a. What is different about this output versus `print(new_list)`?

[Check your response](#)

- b. The `for` loop continues to execute as long as it has elements in the list. This repetition is called **iteration** . (The `while` loop also iterates.) How many times does this `for` loop iterate?

[Check your response](#)

- c. The `for` loop assigns each element in the list to the loop variable. What is the loop variable in this code?

[Check your response](#)

College Board Connection: The FOR EACH loop



- 7 Add the following *indented* line of code below the `for` statement and above the `print (animal)` statement.

```
print("next animal:")
```
- 8 Run your code. You should see ten lines of output, two for each iteration of the loop.

If you were to add two more `print` statements indented underneath the `for` statement, how many lines of output would you see?

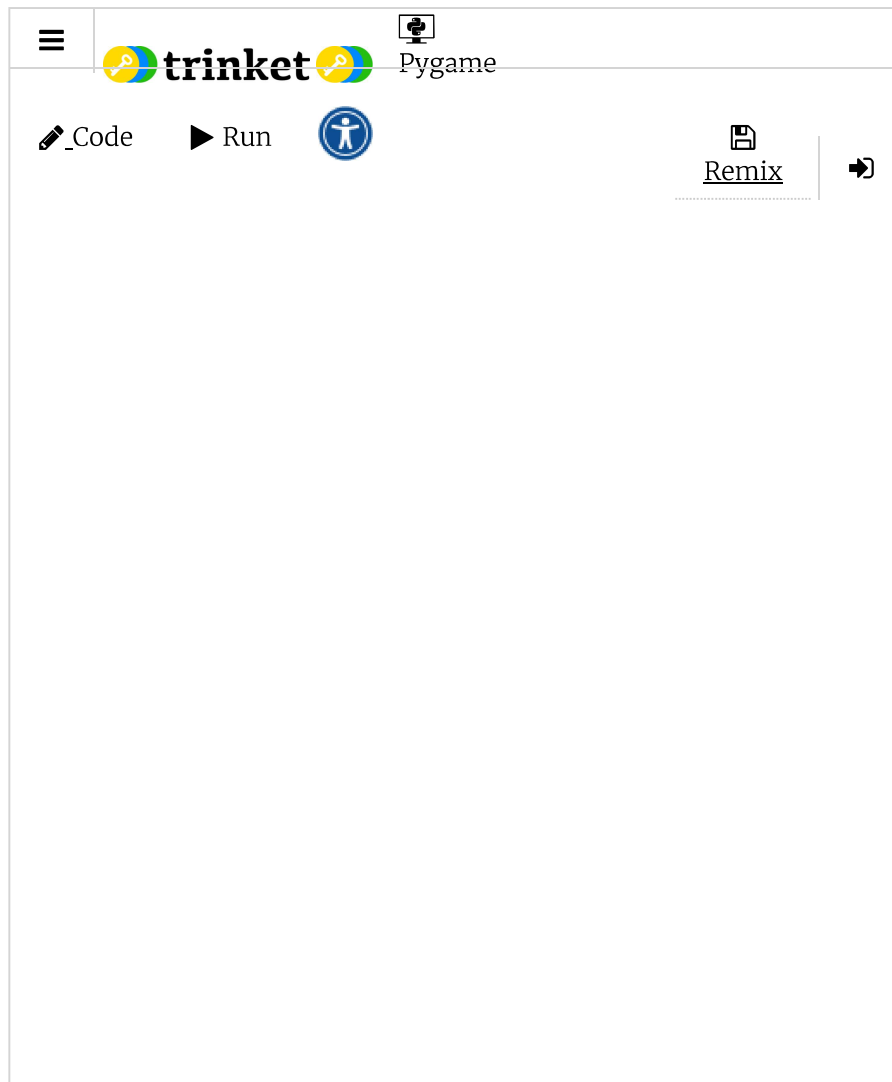
[Check your response](#)

It is time to use your knowledge of lists and the `for` loop in a larger program.

Looping with Lists of Turtles

Lists and `for` loops go naturally together. You can create many elements in a list and process all of them in a small amount of code using `for` loops. In this part of the activity, you will create and process lists of turtles. Up until now, you've been using the default shape of the turtles' "arrow". Here you will loop through a list of other shapes and change your turtles' shape.

- 9 Create a new file in VS Code called `a117_traversing_turtles_[initials].py` and copy this code into it.



10

Explore some of the new and/or interesting parts of the code:

- a. At line 6, there is nothing between the brackets. This means the `my_turtles` list has nothing in it. It is an empty list. Later in the code, you `append`, or add, turtles to this list.
- b. Lines 9 and 10 create two lists with references to shapes and colors. This makes the turtles more interesting.
- c. Lines 12 through 14 populates or adds items to the `my_turtles` list. The `for` loop assigns the loop variable as `s` and then iterates through all of the shapes in the `turtle_shapes` list. On each iteration, the loop

variable `s` gets a shape from the list. Then, a turtle object is created with that shape, using the parameter `shape=s`. Finally, the new turtle (with the new shape) is appended to the `my_turtles` list.

d. The remaining lines of code should be familiar to you.

- 11 Run this program to see what it does. Add comments, in your own words, at lines 16, 20, and 26 describing what each section of code does.

It's Your Turn

Now that you've examined the code and understand how a `for` loop is used to create and draw different shapes using turtles of different shapes, you will modify the code to change the color of turtles, delete colors available to the turtles, add more turtles, and change program parameters to see the results.

- 12 Make your first modification. Set the pen up when you create the turtle object, and set the pen down after you move it to its starting position.

Reminder: Build your code incrementally. First, add the code to set the pen up, and then run and test it. Then add the code to set the pen down, and then run it and test it.

- 13 Make your second modification. Change the body of the `my_turtles` loop so that the next turtle object begins drawing where the last turtle left off. You will find the following turtle methods useful.

Method Name	Example Call	Return Value
<code>xcor()</code>	<code>my_turtle.xcor()</code>	The current x coordinate of the turtle
<code>ycor()</code>	<code>my_turtle.ycor()</code>	The current y coordinate of the turtle

[Need Help?](#)

14

Next, change the color of your turtles:

- Your color-changing code should occur in the loop that creates the turtles.
- The color names you will use are in your `turtle_colors` list.
- On each iteration of the `turtle_shapes` list, pop the `turtle_colors` list.
- Assign the new color with the appropriate turtle methods.

[Need Help?](#)

15

Finally, change the body of the `my_turtles` loop so that the next turtle begins facing the same direction as the previous turtle. This will require initializing a new variable outside the loop to set and save the direction of the previous turtle. Your turtles should end up drawing something like this.

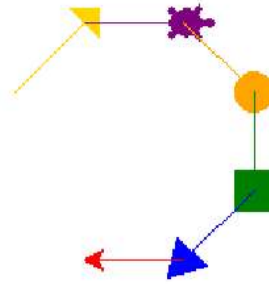


Figure 2. Sample Drawing

Need Help?

16

Experiment with your lists:

- Delete one of the colors from your `turtle_colors` list.
What happens?



Expected Error: This will cause an error.



PLTW COMPUTER SCIENCE NOTEBOOK

With an elbow partner, discuss why the code behaved this way. Document the behavior this resulted in and why you think this behavior occurred.

- Restore the missing color. (If you haven't done so already, you can use `Ctrl+Z` to undo your edits in VS Code.) Check that your program works again.
- Remove a shape from the `turtle_shapes` list. What happens?

**PLTW COMPUTER SCIENCE NOTEBOOK**

With an elbow partner, discuss why the code behaved this way. Document the behavior this resulted in and why you think this behavior occurred.

d. Restore the missing shape.

17

Extend your program:

- a. Add more turtles to your list by copying and pasting the contents of the shapes and colors list.
- b. Expand the drawing so that it spirals outward, increasing the forward path of the drawing turtle on each loop iteration.

18

Experiment by drawing different lengths, different headings, and different pen sizes.

CONCLUSION

- 1 In what situations would lists be used instead of single variables?
- 2 What are some other types of elements that you didn't use in the activity that you think would be appropriate to represent in a list?

Proceed to next activity