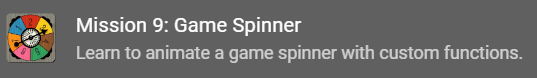


**Mission 9:**

**Game Spinner**

**Student Workbook**



****

**Let’s get animated**

In the last mission, the program used random numbers. For this mission, you will ramp up the random and use it for a game spinner animation.

Go to the Mission 9 Log and fill out the   
Pre-Mission preparation.

* In the last mission, you learned about random numbers. Other than a game, give an example of when you want to select a random person?

**Mission 9: Game Spinner**



**In this project you will make a game spinner that can:**

* Choose the next person to answer a question
* Select a path to take in a maze
* Decide which pizza slice to eat
* Be a spinner for a game
* And anything else you can think of!

**Mission 9: Get started**

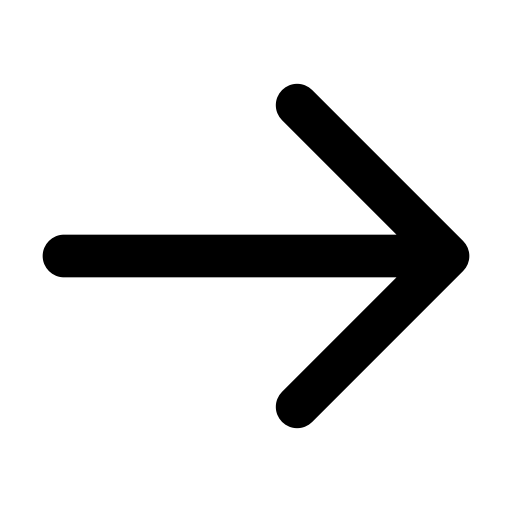
* Go to <https://make.firialabs.com/> and log in.



* Go to Mission 9



* Click and start Mission 9.

**Objective #1: Random arrow**

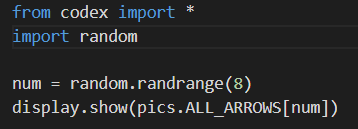
Review getting a random number from Mission 8. You will use the random number to display an arrow from the predefined list: pics.ALL\_ARROWS

You can either:

* Get a random number and use it to display an arrow
* Use random.choice to display a random arrow



**DO THIS:**

* Start a new file named **Game\_Spinner**
* Import the codex module
* Import the random module
* Assign a random **number** to a variable
* Use **display.show()** to display the list item at the number
  + Use CodeTrek if you need help

**Mission Quiz: Which arrows**

Test your skills by **taking the quiz**.

**Objective #2: Click to flick**

Some board games have a game spinner with an arrow that you flick.

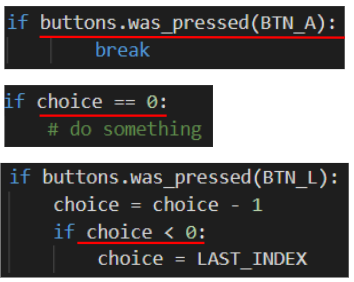
* You will use a button press instead of a flick to “spin” your arrow.
* Make your game spinner arrow change whenever you press either BTN\_A or BTN\_B.

Two things to learn:

* Current button press (Mission 4)
* Logical operators

Current button press

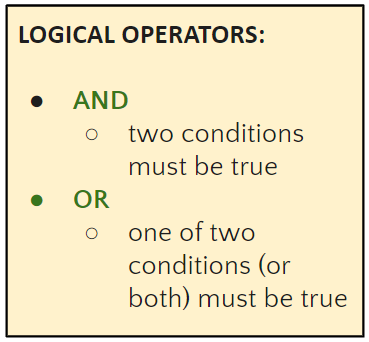
* In Mission 4, you learned about buttons.is\_pressed()
* It checks ***right then*** to see if a button is pressed and will do something if it is.
* For this program, you will use **buttons.is\_pressed()** for both A and B buttons

**Objective #2: Click to flick**

**Comparisons**

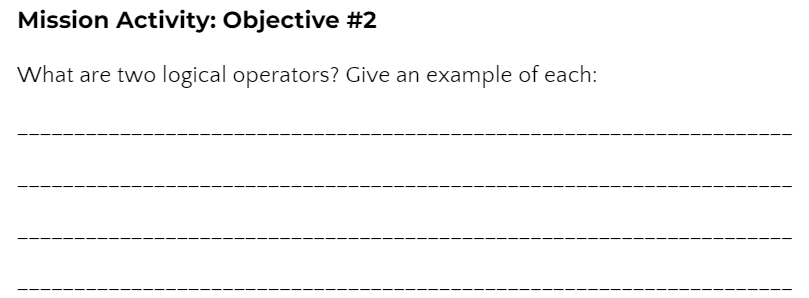
* In earlier missions, you used a condition in an if statement

**Logical operators**

* What if you have two conditions you want to check at the same time?
* For example:
  + Button A pressed OR button B pressed
* Use a logical operator!
* They combine two conditions together

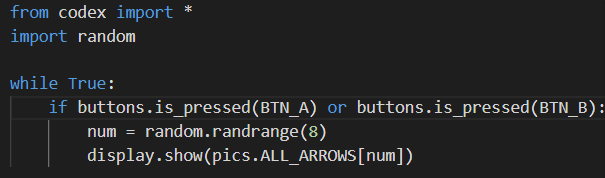
**Objective #2: Click to flick**

**DO THIS:**

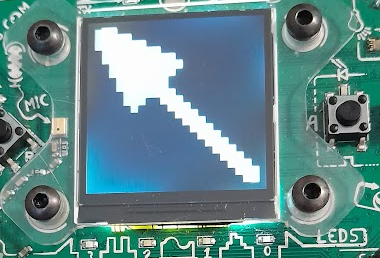
* Go to your Mission Log and write down examples of logical operators.

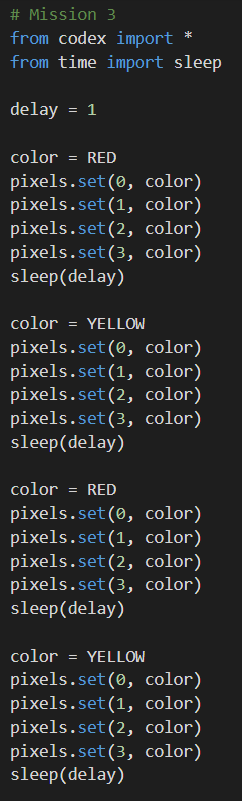
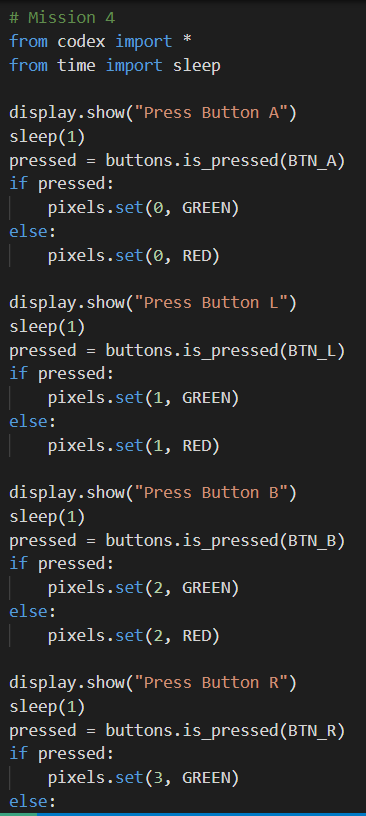
Apply these concepts to make the spinner go

**DO THIS:**

* Add a while True loop
* Add an if statement using buttons.is\_pressed and check for BTN\_A or BTN\_B.
* Be careful with the indenting!
* Test the code by pressing Button A and Button B at different times.

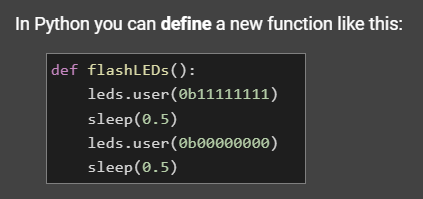
**Objective #3: Fun functions**

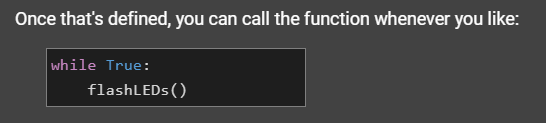
The arrow appears, but it isn’t very realistic. 

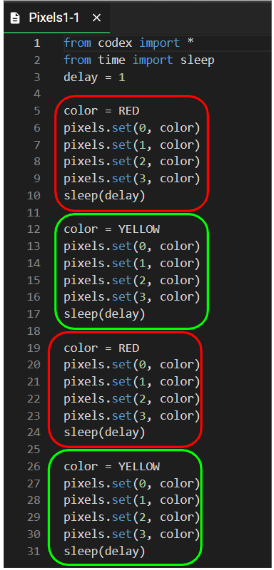
* You want to see some spinning action before the arrow stops.
* You will add some animation
* This could add several lines of code to your program
* Adding several lines of code to your program makes it harder to read
* You have already written programs with several lines of code
  + - Mission 3 and 4 are fairly long
    - Your remixes may have a lot of code as well

**Objective #3: Fun functions**

You can take chunks of code from the main program and make them into functions!

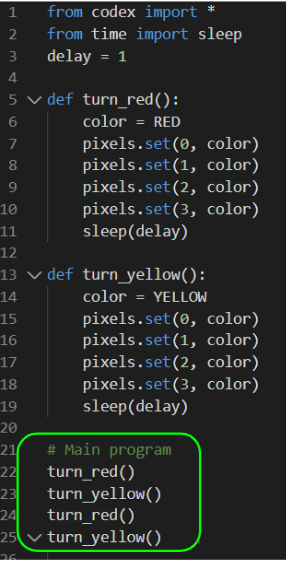




Take a look at code from Mission 3

* In this example, some code is repeated
* The boxes of repeated code are places to make functions

**Objective #3: Fun functions**

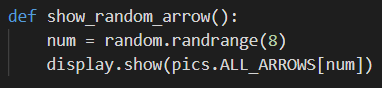
* Create a function for each color
* Delete any extra code
* Call the functions in the order you want to run them

Do you see how much easier the code is to read? And it is shorter!

**Objective #3: Fun functions**

You can create a function any time you want to keep the code easy to read.

* The keyword means “define function”
* Use a colon (:) at the end of the line, just like a **while loop** and an **if statement** – you are making a block of code
* Indent the lines of code inside the function



A function must be defined before it can be called.

* Define your function above the **while True** loop
* Move the code from the while loop to the function
* Call the function in the while loop
  + NOTE: a function call DOES NOT have the word “def” or a colon (:)



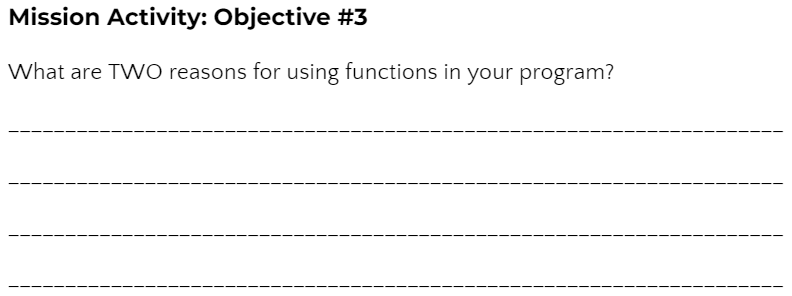
**Objective #3: Fun functions**

**DO THIS:**

* Define the function **show\_random\_arrow()**
* Call the function **show\_random\_arrow()**
* Follow the steps from the two previous slides, or use CodeTrek for help
* Test your program by pressing both Button A and Button B at different times. It should work just the same as before, but with a function.

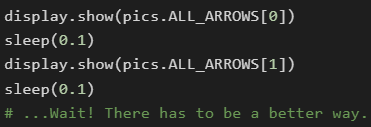
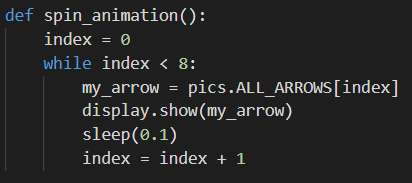
**DO THIS:**

* After objective 3 is completed, the message displayed gives another reason for using functions.
* Go to your Mission Log and answer the question about functions

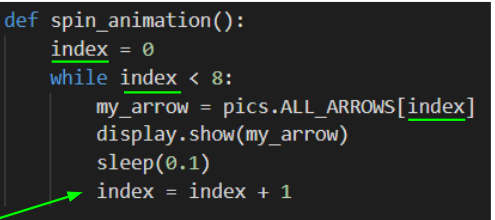


**Objective #4: Animation**

You added a function, but your code still does the same thing – displays one arrow.

* You will create an animation by going through the arrows in the list in order – quickly!
* There are 8 arrows in the list
* You could call each arrow with a short delay:
* Or, you can use a loop!
* Not an infinite loop (like while True) but a loop that goes 8 times.

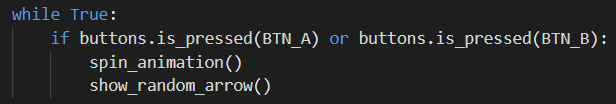
**Objective #4: Animation**

* This loop counts how many times it is executed
* The variable **index** is used both to count the loops and to display an arrow in the list
* The variable **index** must be incremented inside the loop
  + You learned about this in Mission 7

**Objective #4: Animation**

**DO THIS:**

* Import **sleep** from the time module
* Define the function **spin\_animation**
  + Use the previous slide for help
* Call the function spin\_animation just before you call **show\_random\_arrow()**
* Test your code by pressing both Button A and Button B



**Mission Quiz: Indented?**

Test your skills by **taking the quiz**.

**Objective #5: Style points - physics part 1**

The game spinner is nice – but still not very realistic.

* It needs to spin more than one time around
* It needs to gradually slow down before   
  stopping
* Let’s start with the first problem –   
  making the spinner go more than one time around

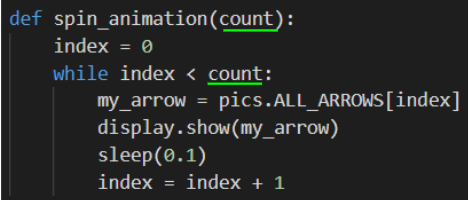
Right now the spinner goes one time because the loop checks for index < 8.

* What if it checked for a number larger than 8?
* You can tell the function how many times to loop
* When you give the function information, it is called a **parameter**

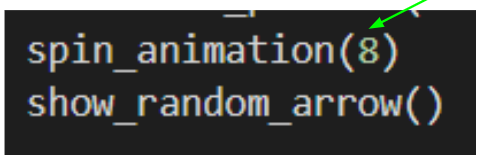
**Objective #5: Style points - physics part 1**

**A Parameter**

* The parameter is shown in the parenthesis

The function uses the parameter, or data, to complete its task. 

**An argument**

* When you call a function, you can supply values for those parameters.
* Values you pass when calling a function are called arguments.
* An argument can be a variable or a literal value.

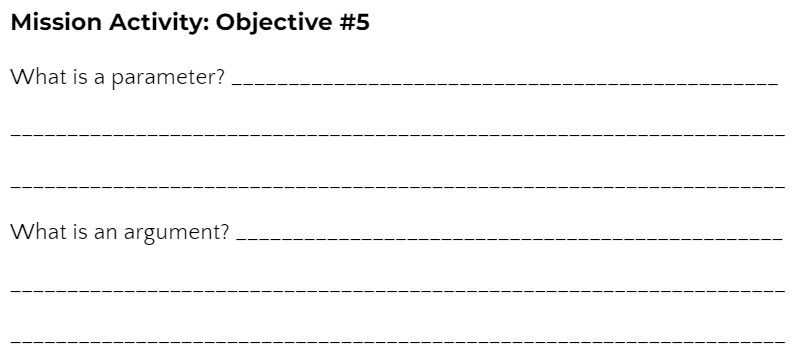
**Objective #5: Style points - physics part 1**

**DO THIS:**

* Add the parameter **count** to the **spin\_animation()** function
* Use **count** in the while loop
* Call **spin\_animation()** with the argument **8**
* Follow the steps from the two previous slides, or use CodeTrek for help

**DO THIS:**

* Go to your Mission Log and answer the questions about parameters and arguments



**Objective #6: Unruly index**

Time to increase the number of spins. Can you make the arrow spin longer than 8 times?

* Change the argument in the function call to 30

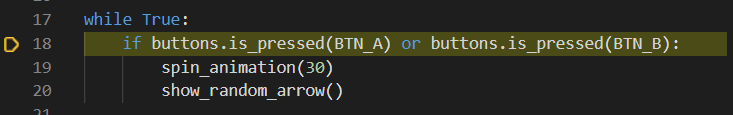
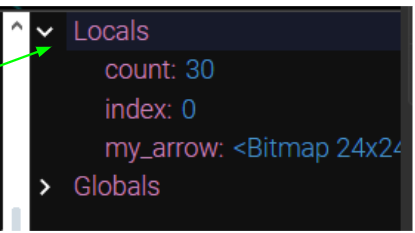


* Run the code and press BTN\_A
* Do you get an error?
* Let’s find out why, and how to fix it

**Objective #6: Unruly index**

**DO THIS:**

* Use the debugger
* Step in the program

* When you see this line, press BTN\_A while pressing “Step In”
* Open the console
* Watch the local variables as you continue to step in the code
* What is the value of **index** when the error occurs?
* Go to your Mission Log and answer the question.

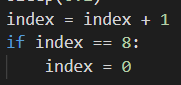
**Objective #7: Tame the unruly index**

Did you find the error?

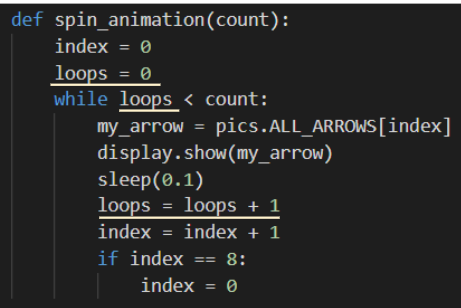
* The list has eight arrows
* The index values are 0 through 7
* When the index value is 8, it is past the end of the list

Solve this problem by using another variable for the loop, instead of index.

You will still increment **index** in the loop.

* Do you remember in Mission 7 you scrolled through a list?
* You will use the same wrapping code

Now you can use a different variable to count the loops.

Remember to increment the **loops** variable inside the loop

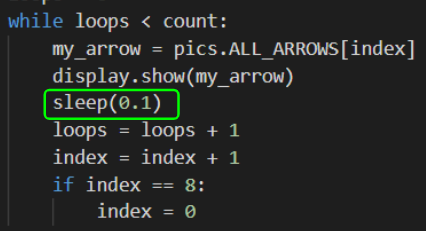
**Objective #7: Taming the unruly index**

**DO THIS:**

* Define the **loops** variable
* Compare **loops** to **count** in the while loop
* Increment **loops**
* Write code to wrap **index**
* Follow the steps from the two previous slides, or use CodeTrek for help

**Objective #8: Style points - physics part 2**

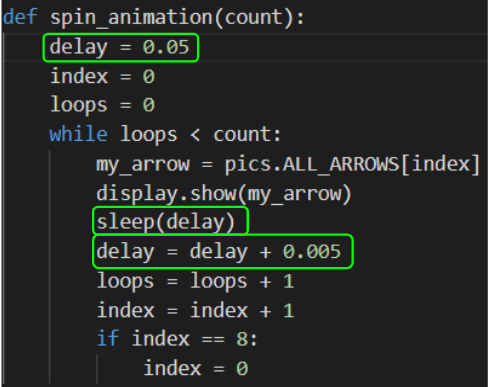
Spin down. For a more realistic spin, you can make the arrows gradually slow down before stopping.

* Right now the same amount of time is used: **sleep(0.1)**
* Use the variable **delay** instead of a literal value!

Increment **delay** by a little bit



**DO THIS:**

* Define the **delay** variable
* Use **delay** in sleep()
* Increment **delay**

**Mission Complete**

You have completed the ninth mission. 

**Do this:**

* Read your “Completed Mission” message
* Complete your Mission 9 Log
  + Post-Mission Reflection
* Get ready for your next mission!

**Wait! Before you go … Clear the CodeX**

Go to FILE -- BROWSE FILES

Select the “**Clear**” file and open it

Run the program to clear the CodeX

**Okay. Now you can go.**