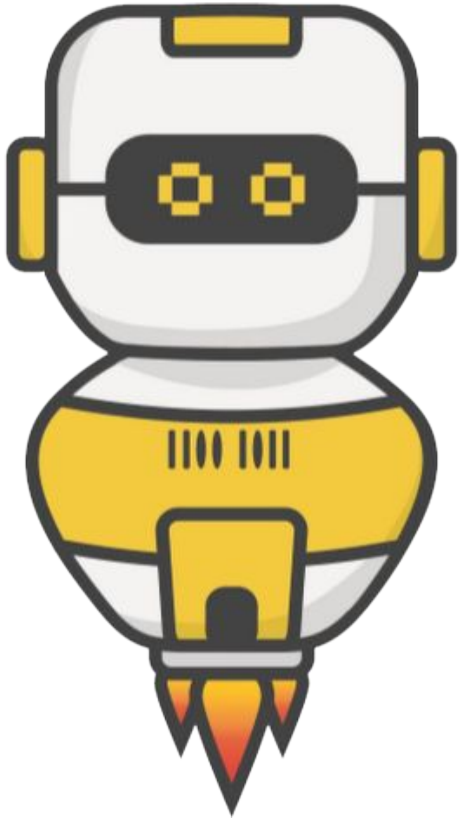
# Game Spinner

Mission 9

## Pre-Mission Preparation

In the last mission, you learned about random numbers.

- Other than a game, give an example of when you might want to select a random person.



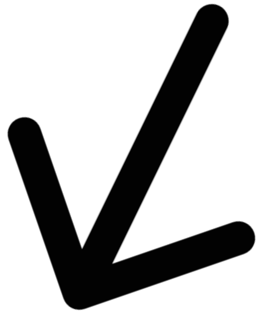
## Mission 9: Game Spinner

**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!

## 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



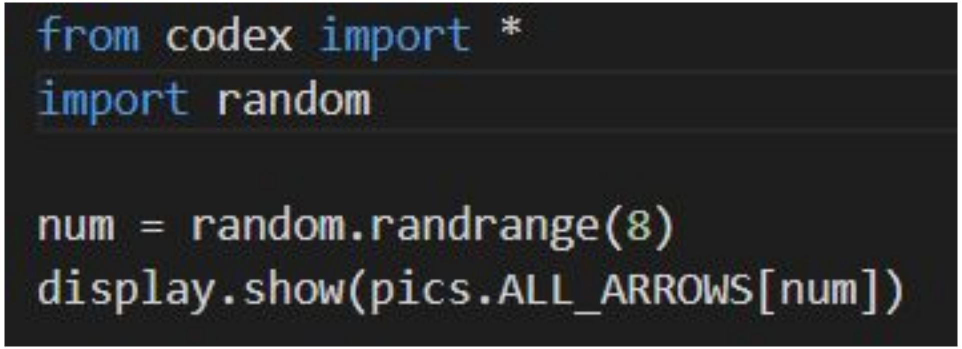
You can either:

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

## Mission Activity #1

**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



```
from codex import *
import random

num = random.randrange(8)
display.show(pics.ALL_ARROWS[num])
```

## Which arrows

In previous missions, you learned about random number and lists.

- Answer 2 quiz questions about random numbers and the list index

## 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.

## Objective #2: Click to flick

Two things to learn:

- Current button press
    - Mission 4
- Logical operators

## Objective #2: Click to flick

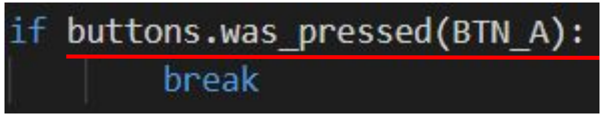
Current button press

- 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

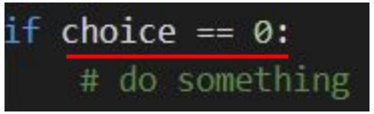
## Objective #2: Click to flick

Logical operators

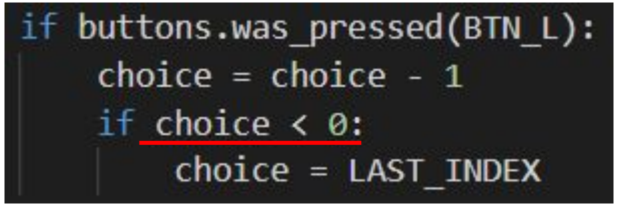
- In earlier missions, you used a condition in an if statement



```
if buttons.was_pressed(BTN_A):
        break
```



```
if choice == 0:
    # do something
```



```
if buttons.was_pressed(BTN_L):
    choice = choice - 1
    if choice < 0:
        choice = LAST_INDEX
```

# Objective #2: Click to flick

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

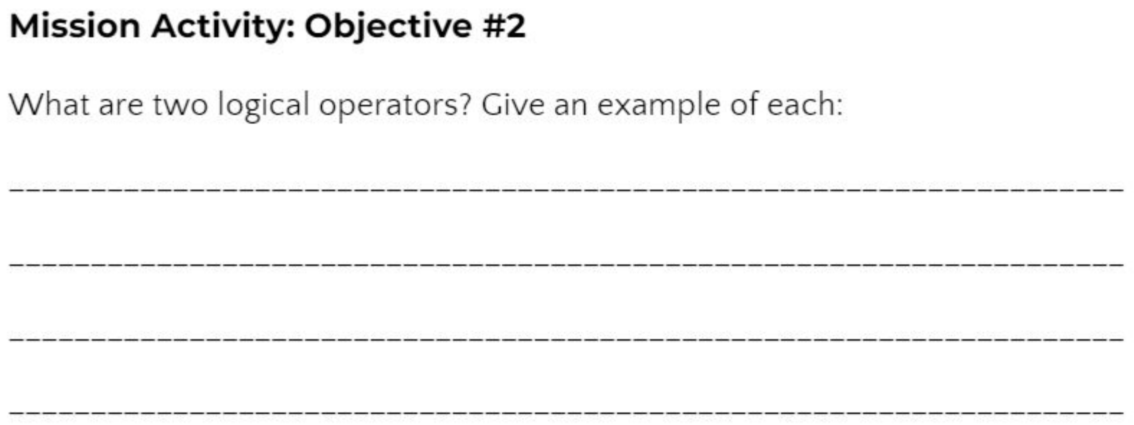
**LOGICAL OPERATORS:**

- **AND**
  - two conditions must be true
- **OR**
  - one of two conditions (or both) must be true

## Mission Activity #2

**DO THIS:**

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



**Mission Activity: Objective #2**

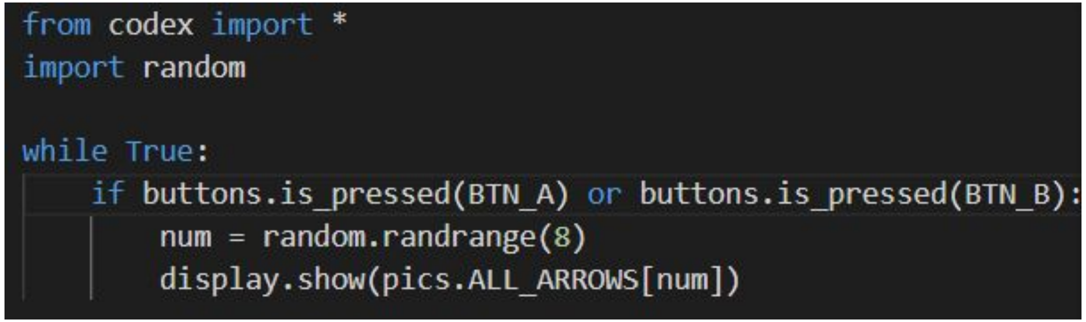
What are two logical operators? Give an example of each:

## Mission Activity #2

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.

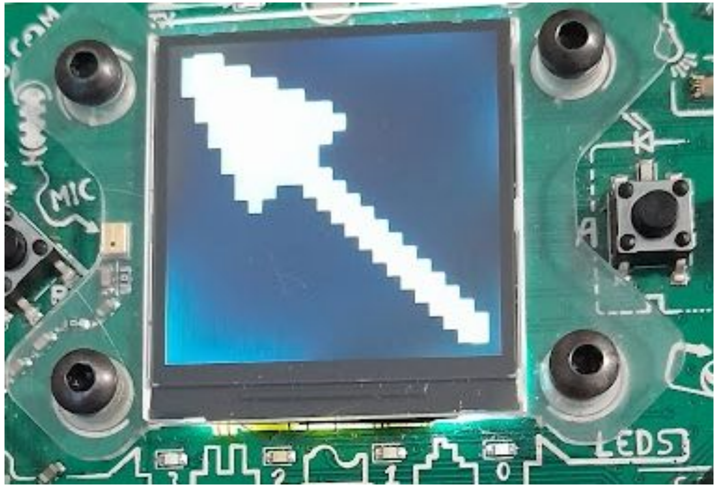


```
from codex import *
import random

while True:
    if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):
        num = random.randrange(8)
        display.show(pics.ALL_ARROWS[num])
```

## Objective #3: Fun Functions

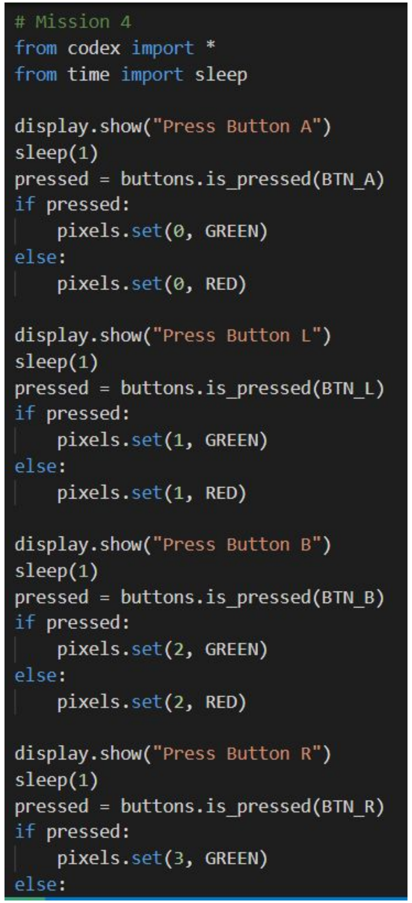
The arrow appears, but it isn't very realistic.

- 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

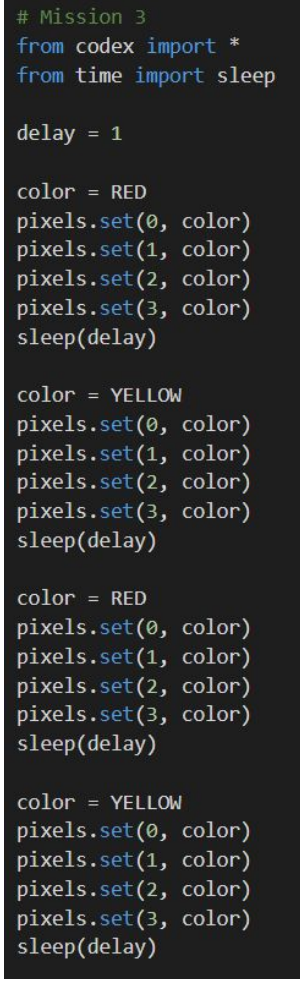


# Objective #3: Fun Functions

- 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



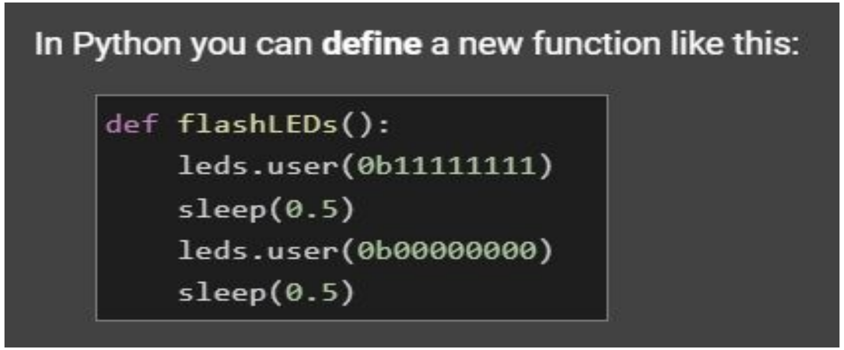
```
# Mission 4
from codex import *
from time import sleep

display.show("Press Button A")
sleep(1)
pressed = buttons.is_pressed(BTN_A)
if pressed:
    pixels.set(0, GREEN)
else:
    pixels.set(0, RED)

display.show("Press Button L")
sleep(1)
pressed = buttons.is_pressed(BTN_L)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)

display.show("Press Button B")
sleep(1)
pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(2, GREEN)
else:
    pixels.set(2, RED)

display.show("Press Button R")
sleep(1)
pressed = buttons.is_pressed(BTN_R)
if pressed:
    pixels.set(3, GREEN)
else:
```



```
# Mission 3
from codex import *
from time import sleep

delay = 1

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)
```

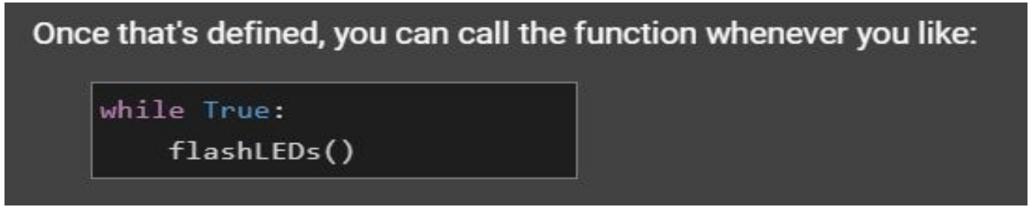
## Objective #3: Fun Functions

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



In Python you can **define** a new function like this:

```
def flashLEDs():
    leds.user(0b11111111)
    sleep(0.5)
    leds.user(0b00000000)
    sleep(0.5)
```



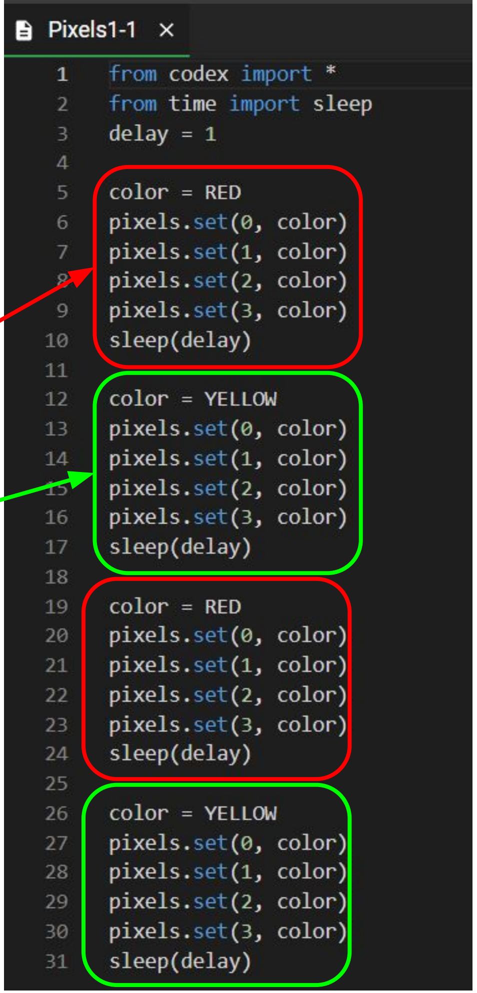
Once that's defined, you can call the function whenever you like:

```
while True:
    flashLEDs()
```

## Functions in Mission 3

Take a look at code from Mission 3

- In this example, some code is repeated
- These are places to make functions

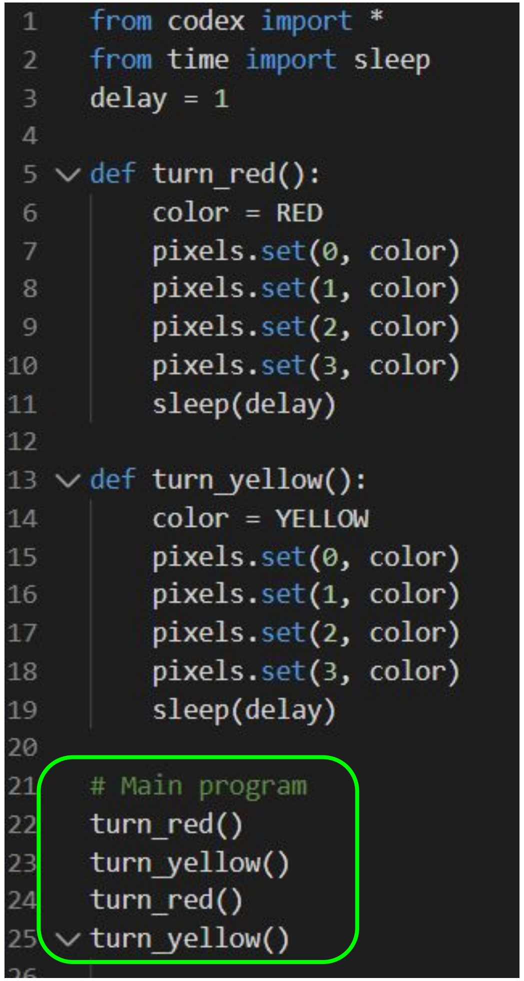


```
from codex import *
from time import sleep
delay = 1

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)
```

## Call a function

- 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!

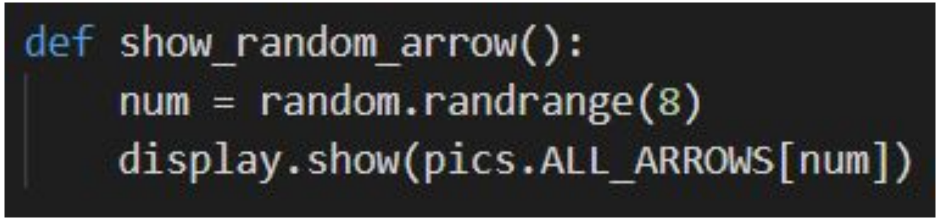


```python
from codex import *
from time import sleep
delay = 1

def turn_red():
    color = RED
    pixels.set(0, color)
    pixels.set(1, color)
    pixels.set(2, color)
    pixels.set(3, color)
    sleep(delay)

def turn_yellow():
    color = YELLOW
    pixels.set(0, color)
    pixels.set(1, color)
    pixels.set(2, color)
    pixels.set(3, color)
    sleep(delay)

# Main program
turn_red()
turn_yellow()
turn_red()
turn_yellow()
```

## Objective #3: Fun Functions

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

- The keyword `def` 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

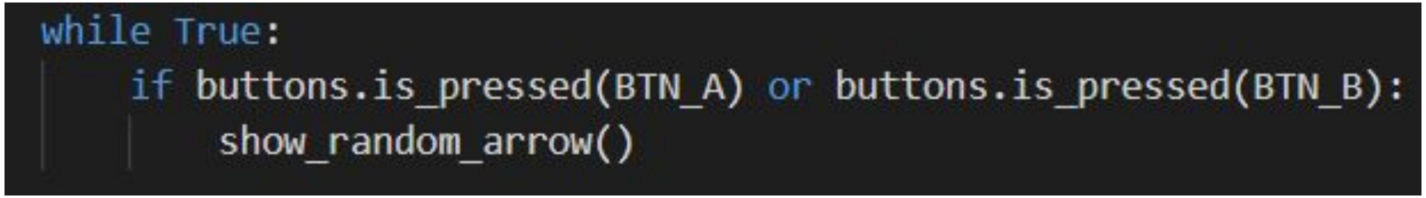


```
def show_random_arrow():
    num = random.randrange(8)
    display.show(pics.ALL_ARROWS[num])
```

## Objective #3: Fun Functions

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 (:)



```
while True:
    if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):
        show_random_arrow()
```

## Mission Activity #3

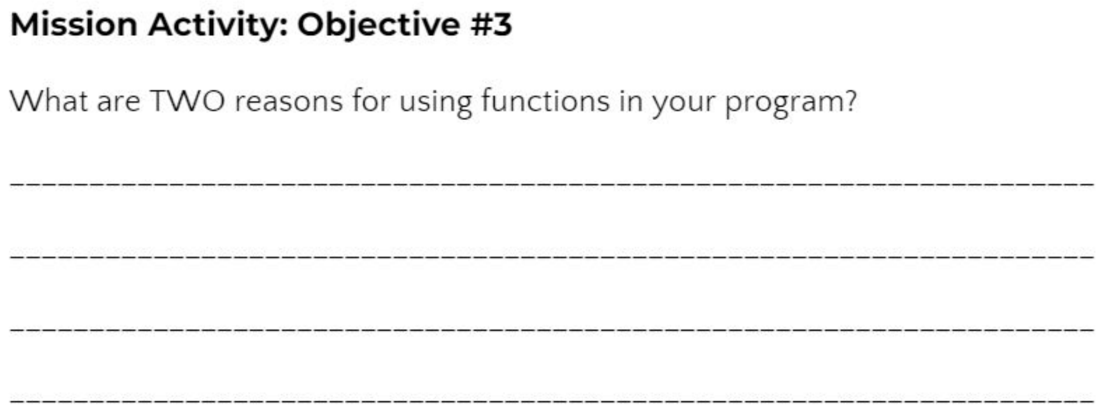
### DO THIS:

- 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.

## Mission Activity #3

**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



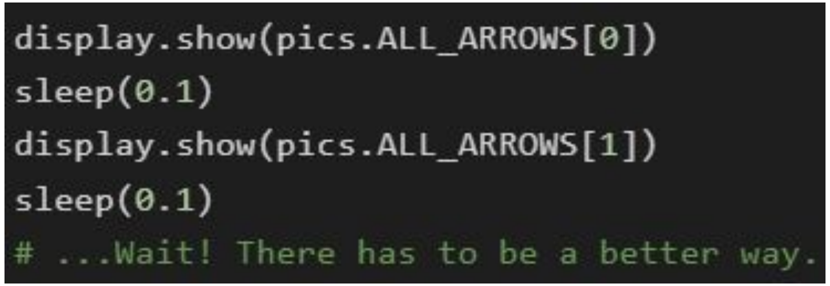
**Mission Activity: Objective #3**

What are TWO reasons for using functions in your program?

## Objective #4: Animation

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

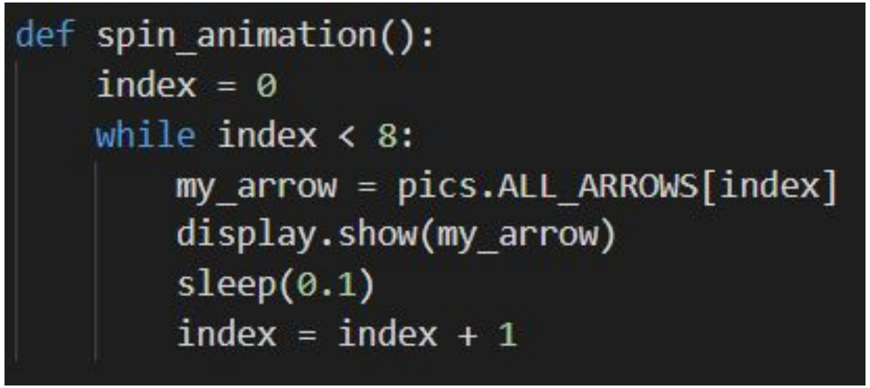
- You will create an animation by going through the arrows in the list in order – quickly!
- There are 8 arrows in the list
- You could call each arrow with a short delay:



```
display.show(pics.ALL_ARROWS[0])
sleep(0.1)
display.show(pics.ALL_ARROWS[1])
sleep(0.1)
# ...Wait! There has to be a better way.
```

## Objective #4: Animation

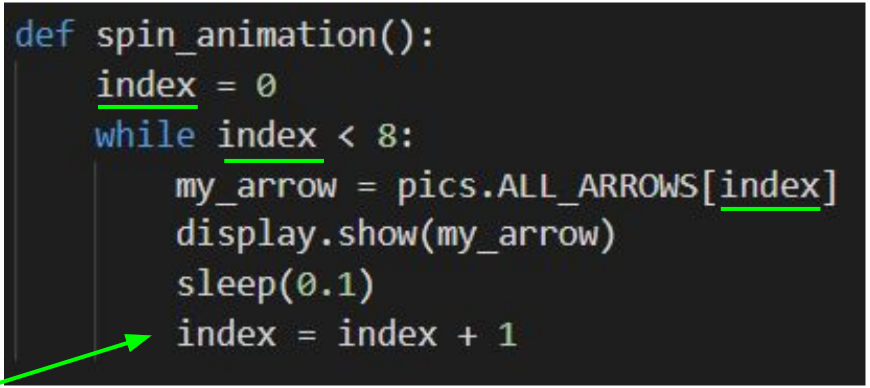
- Or, you can use a loop!
- Not an infinite loop (like while True) but a loop that goes 8 times.



```
def spin_animation():
    index = 0
    while index < 8:
        my_arrow = pics.ALL_ARROWS[index]
        display.show(my_arrow)
        sleep(0.1)
        index = index + 1
```

## 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

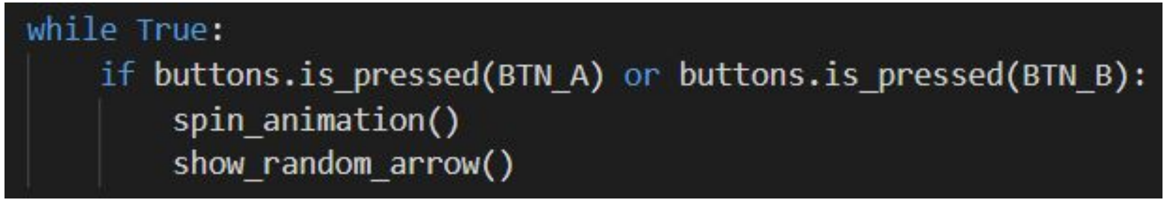


```
def spin_animation():
    index = 0
    while index < 8:
        my_arrow = pics.ALL_ARROWS[index]
        display.show(my_arrow)
        sleep(0.1)
        index = index + 1
```

## Mission Activity #4

### 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

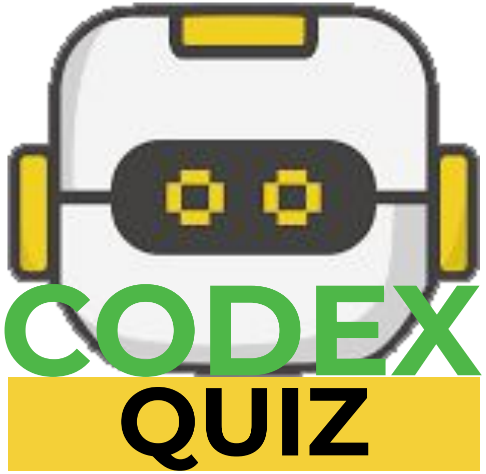


```
while True:
    if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):
        spin_animation()
        show_random_arrow()
```

## Indented?

During this mission you have learned about indenting, loops and functions.

- Answer 3 quiz questions about the Objectives 1-4



## 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

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

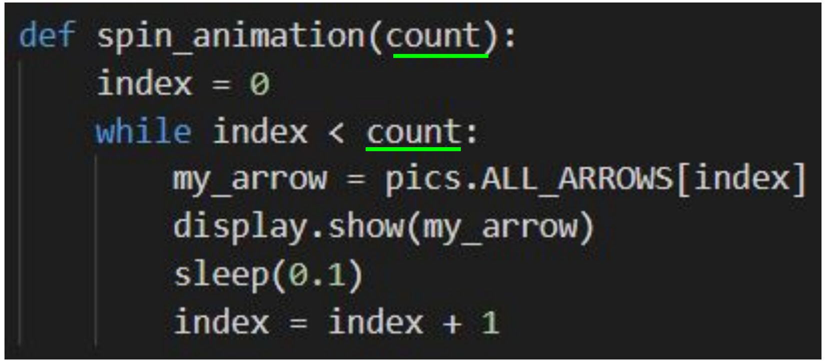
Right now the spinner goes one time because the loop checks for index < 8.

- 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.

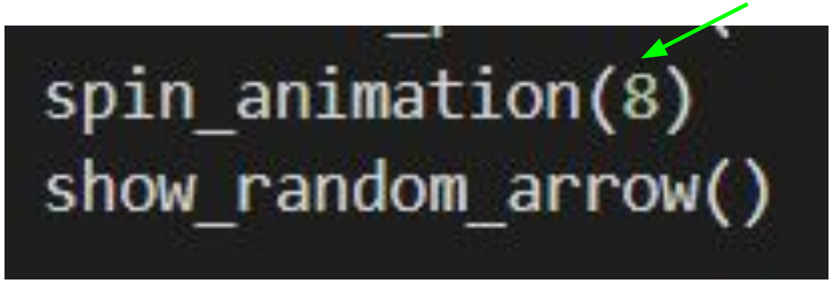


```
def spin_animation(count):
    index = 0
    while index < count:
        my_arrow = pics.ALL_ARROWS[index]
        display.show(my_arrow)
        sleep(0.1)
        index = index + 1
```

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

An argument

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



```
spin_animation(8)
show_random_arrow()
```

## Mission Activity #5

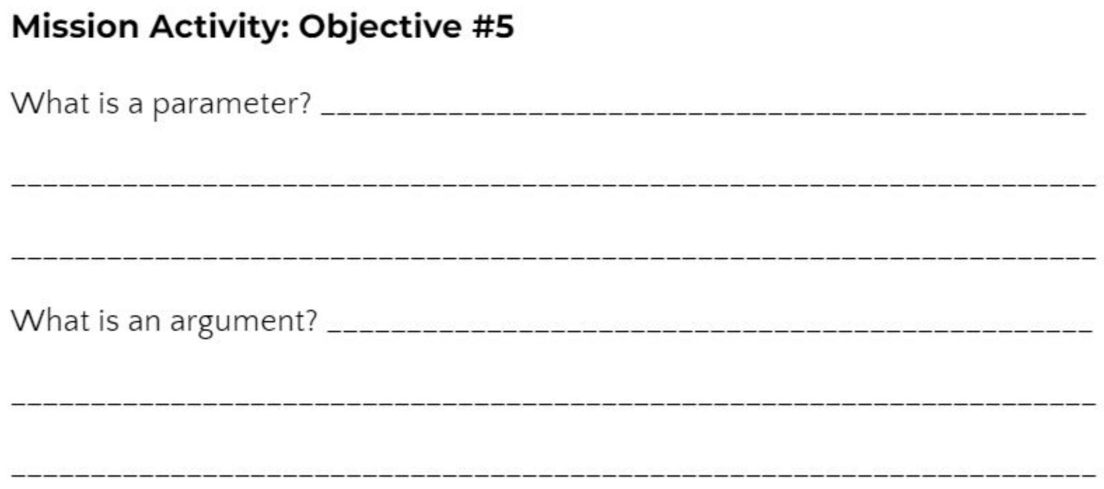
### DO THIS:

- 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

# Mission Activity #5

## DO THIS:

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



**Mission Activity: Objective #5**

What is a parameter? ____________________________________________

____________________________________________________________

____________________________________________________________

What is an argument? ____________________________________________

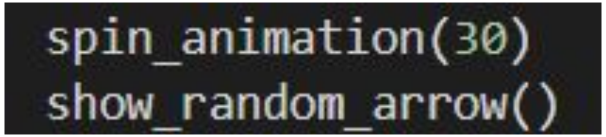
____________________________________________________________

____________________________________________________________

## Objective #6: Unruly index

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

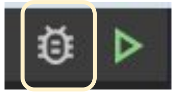
- Change the argument in the function call to 30
- Run the code and press BTN_A
- Do you get an error?
- Let’s find out why, and how to fix it

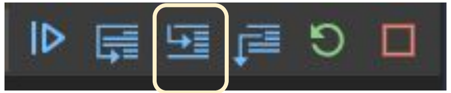


```
spin_animation(30)
show_random_arrow()
```

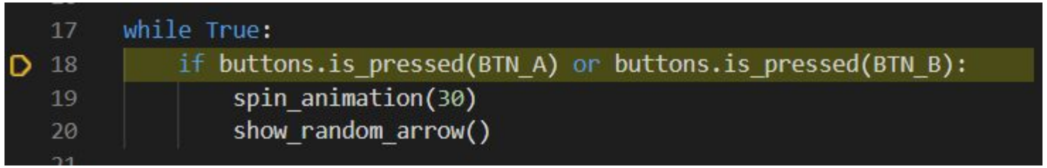
# Mission Activity #6

## DO THIS:

- Use the debugger 
- Step in the program



- When you see this line, press BTN_A while pressing



```
while True:
    if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):
        spin_animation(30)
        show_random_arrow()
```

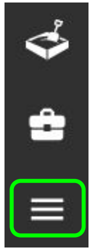
< continued on next slide>

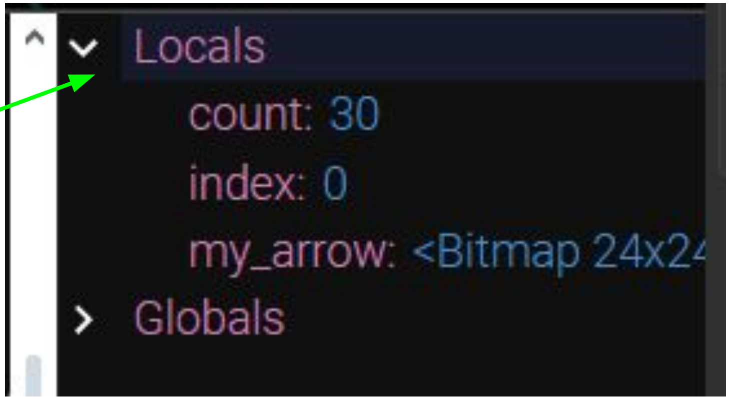
## Mission Activity #6

**DO THIS:**

- 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.




Locals
count: 30
index: 0
my_arrow: <Bitmap 24x24
Globals

## Objective #7: Tame the unruly index

Did you find the error?

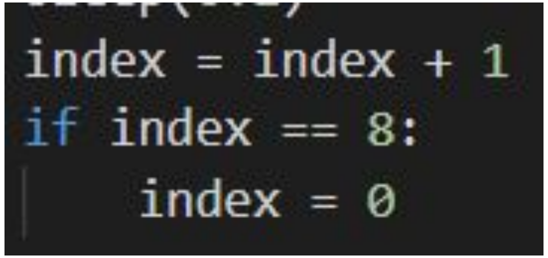
- The list has eight arrows
- The index values are 0 through 7
- When the index value is 8, it is past the end of the list

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

## Objective #7: Tame the unruly 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:

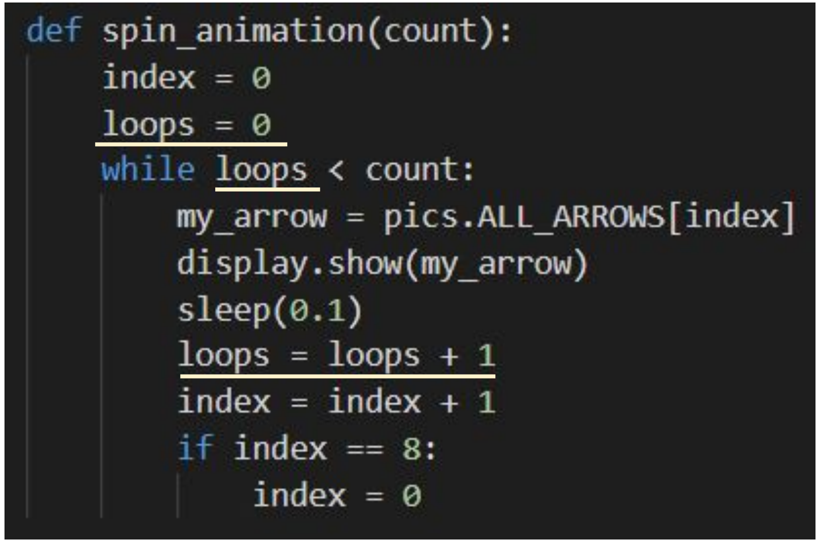


```
index = index + 1
if index == 8:
    index = 0
```

## Objective #7: Tame the unruly index

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

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



```
def spin_animation(count):
    index = 0
    loops = 0
    while loops < count:
        my_arrow = pics.ALL_ARROWS[index]
        display.show(my_arrow)
        sleep(0.1)
        loops = loops + 1
        index = index + 1
        if index == 8:
            index = 0
```

## Mission Activity #7

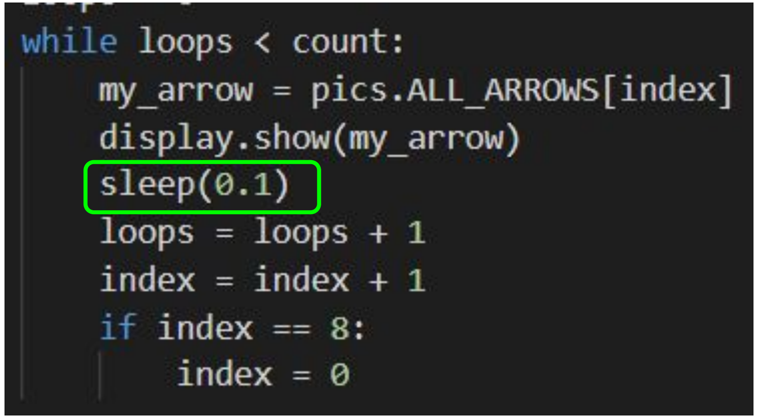
**DO THIS:**

- 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

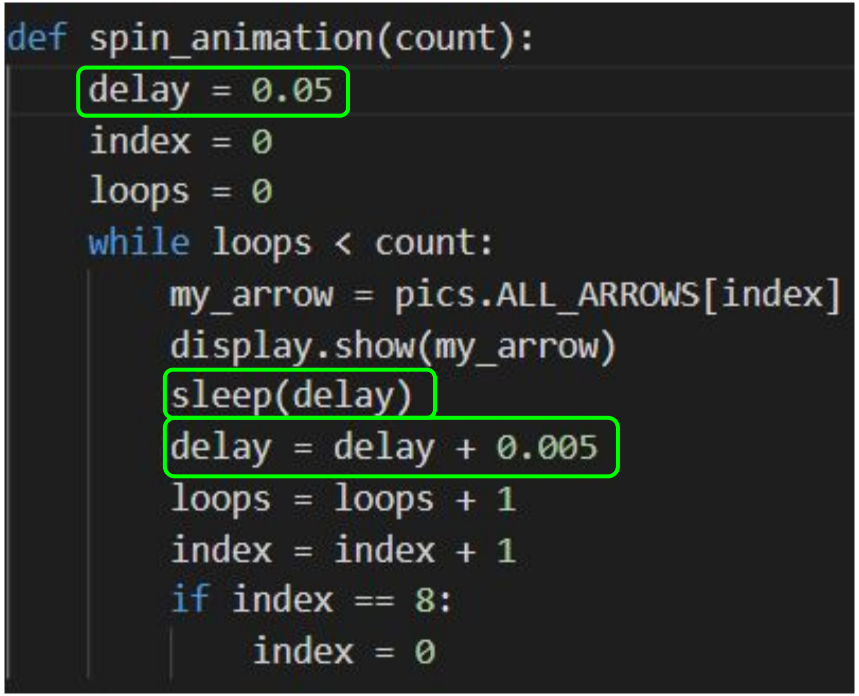


```
while loops < count:
    my_arrow = pics.ALL_ARROWS[index]
    display.show(my_arrow)
    sleep(0.1)
    loops = loops + 1
    index = index + 1
    if index == 8:
        index = 0
```

## Mission Activity #8

**DO THIS:**

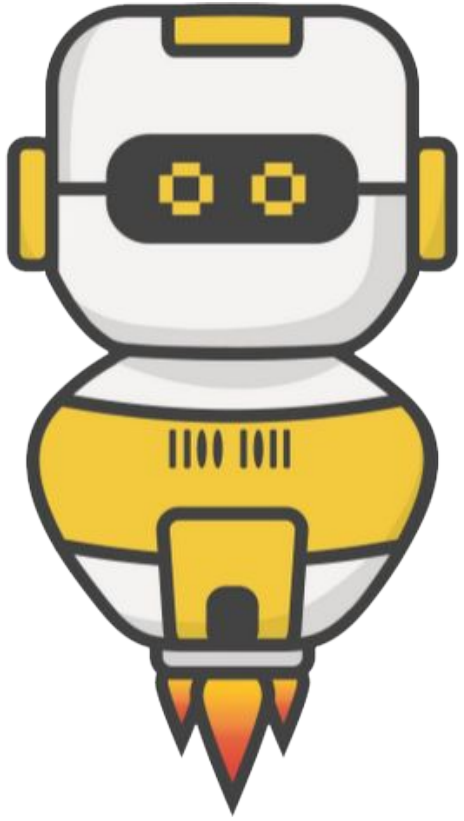
- Define the **delay** variable
- Use **delay** in sleep()
- Increment **delay**



```
def spin_animation(count):
    delay = 0.05
    index = 0
    loops = 0
    while loops < count:
        my_arrow = pics.ALL_ARROWS[index]
        display.show(my_arrow)
        sleep(delay)
        delay = delay + 0.005
        loops = loops + 1
        index = index + 1
        if index == 8:
            index = 0
```

## Post-Mission Reflection

- Read the “completed mission” message and click to complete the mission
- Complete the Mission 9 Log



# Clearing your CodeX

Go to FILE -- BROWSE FILES
Select the “**Clear**” file and open it
Run the program to clear the CodeX