

### Mission 3

What do you expect the following code to do?

+5 XP

```
from codex import *  
display.show(pics.HEART)  
display.show(pics.HAPPY)
```

- Show the images for about 1 second each
- Display only the first image
- Display each image quickly and end showing the last one

What does `from codex import *` do?

+5 XP

- Turns on the codex LEDs
- Provides access to built-in codex code
- Imports asterisks from the land of codex

What does `delay = 1` do?

+5 XP

- Puts the CPU into sleep mode for 1 second
- Assigns the value 1 to a variable named 'delay'
- Delays execution for 1 second

### Mission 4

Which of the following is **NOT** a standard Python type?

+5 XP

- `'text'`
- `'int'`
- `'str'`

What will happen if you run this code?

+5 XP

```
from codex import *  
display.show(8)
```

- The program will error
- An 8 will be shown below any text already on the screen
- The display will show an 8

What code will tell me if the **UP** button is currently pressed?

+5 XP

- `buttons.is_pressed(BTN_D)`
- `buttons.was_pressed(BTN_U)`
- `buttons.is_pressed(BTN_U)`

What will happen if you run this code?

+5 XP

```
x = False  
if x:  
    display.show('if')  
else:  
    display.show('else')
```

- Your program will error
- `'if'` will print on the display
- `'else'` will print on the display

## Mission 5

Choose the two places that the CodeX can output sound.

+5 XP

- The Display
- The USB Port
- The Headphone Port
- The Speaker

Answered 2 of 2

Which of these is **NOT** a tool to make your code more readable? +5 XP

- Blank lines in your code
- Variable names without meaning (like `x`)
- Comments that explain your code

## Mission 6

What happens if you press button 'A' when stepping? +5 XP

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

- Next `buttons.was_pressed(BTN_A)` will be False
- Buttons are ignored when stepping and paused
- Next `buttons.was_pressed(BTN_A)` will be True

What does the `break` statement do? +5 XP

- Causes the code to stop.
- Jumps over the next line of code.
- Breaks out of a loop.
- Crashes the program.

```
if buttons.was_pressed(BTN_A):
    delay = delay + 0.2
```

## Mission 7

```
choice = 0

if buttons.was_pressed(BTN_L):
    choice = choice - 1
if buttons.was_pressed(BTN_R):
    choice = choice + 1
```

What does your "scroll both directions" program do when you keep pressing button 'R' after `pics.ASLEEP` is shown? **+5 XP**

- The `choice` variable starts over at zero.
- The `choice` variable stops at 3 which is the number of the last image.
- The `choice` variable goes to 4 and keeps counting up.

What does the double-equals sign mean in `if choice == 0`? **+5 XP**

- Selects a choice of either the symbol `==` or `0`.
- Compares `choice` to zero, branching when `choice` is zero.
- Assigns the variable `choice` a value of zero.

<https://sim.firialabs.com/>

Why do you have to **subtract 1** from `len(my_list)` to get `LAST_INDEX`? **+5 XP**

- List indexes start at 0 so the index of the last item is `len(my_list) - 1`.
- List indexes are negative numbers so `len(my_list) - 1` is required.
- Because the program needs to count up as well as down in the list.

## CONCEPT: type checking

The built-in function `type()` is used to read the *type* of object a variable refers to.

- Each `data type` has a name such as:
  - `str` for `strings`
  - `int` for `integers`
  - `tuple` for `tuples`
- The `type(object)` function returns the the object's type.

## Mission 8

Why does `range(10)` only go up to 9??

+5 XP

- The `range(10)` also includes -1 so the max is 9.
- The variable itself consumes one integer so there are only 9 values in `range(10)`.
- Ranges start at 0 (zero), and there are 10 values from 0-9.

What is the `count` variable doing for you in this program? +5 XP

```
answers = ["0", "1", "2"]

while True:
    if buttons.was_pressed(BTN_A):
        count = len(answers)
        index = random.randrange(count)
```

- The `count` is a beloved character in educational television.
- The `count` variable automatically scans the list and counts the number of items.
- The `count` variable stores `len(answers)` to give to the 'randrange' function.

The `random.choice()` function simplifies what your code is already doing

- Behind the scenes it does *exactly* what your code was doing to pick an item from a list.

```
# Choose a random color from the list
color = random.choice(COLOR_LIST)
```

- Use `random.choice()` to set each pixel to its own `random` item from `COLOR_LIST`

## Mission 9

What are the possible values of `num`?

+5 XP

```
import random
num = random.randrange(8)
```

- 0,1,2,3,4,5,6,7    1,2,3,4,5,6,7,8    -3,-2,-1,0,1,2,3,4

Which image is displayed by `display.show()` below?

+5 XP

```
pics.ALL_ARROWS = [
    pics.ARROW_N,
    pics.ARROW_NE,
    pics.ARROW_E,
    pics.ARROW_SE,
    pics.ARROW_S,
    pics.ARROW_SW,
    pics.ARROW_W,
    pics.ARROW_NW
]

display.show(pics.ALL_ARROWS[3])
```

- `pics.ARROW_E`    `pics.ARROW_SW`

Why is the `if` statement below **indented** beneath the `while`?

+5 XP

```
while True:
    if buttons.is_pressed(BTN_A):
        display.show(pics.ALL_ARROWS[0])
```

- Because `if` statements have to be indented.
- So that the arrow is only displayed when a button is pressed.
- So that it runs completely inside the loop.

<https://sim.firialabs.com/>

Which [condition](#) stops the loop in this code?

+5 XP

```
index = 0
while index < 8:
    index = index + 1
```

- The loop stops when `index` reaches 0.
- The loop stops when `index` reaches 8.
- An infinite loop never stops.
- The statement `index = index + 1` ends the loop.

What is `show_random_arrow` in the code below?

+5 XP

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

- A Function
- A Party
- A String
- A Loop

## CONCEPT: *Parameters and Arguments*

[Functions](#) in Python can be defined with a list of [parameters](#).

- When you call a function, you can supply values for those parameters.
- For example when you call `display.show("hello")` you are providing the value `"hello"` to the function.
  - Values you pass when calling a function are called [arguments](#).
- Functions are *always* defined and called with **parentheses**, even if there are no parameters.

Change your `spin_animation()` function to define a `count` parameter like this:

```
def spin_animation(count):
    index = 0
    while index < count:
        # ...show img and delay
```

- Variables defined *inside* your function (and parameters like `count`) are **local variables**.

- You'll find them separately listed in the debug **console**, as shown here.



- You will need to hold the button down on the CodeX when you **STEP** on the `is_pressed()` call!

## Locals and Globals

### Variable scope and lifetime

Variables that you define outside of a function are called **global** variables.



- Their "lifetime" (how long they retain value) is the whole time the program is running, and their "scope" (where they can be seen/used by code) is the whole file. That's *global*, dude!

The other kind of variable is **local**. Variables created inside **functions** are *local*.

- They only exist while the function is running, then they go away.
- They only exist while the function is running, then they go away. That's *local* scope.

### What if I want to change a global variable from inside a function?

- You have to declare it in the function with the `global` keyword, like:

```
count = 0

def check_buttons():
    global count      # <----- Tells Python to use the
                    #               variable, not to make a new one
    if button_b.was_pressed():
        count = count + 1
    return str(count)
```

Now when you assign to `count`, you're assigning to the **global** one



Now when you assign to `count`, you're assigning to the **global** one rather than creating a new **local** one.

#### Note:

Variables that are *referenced* inside a `function` but **not** assigned a value within it are assumed to be global. So your code can *access* any global variable it can see! Only when you need to **change** the variable is a `global` declaration required.

### Have you found the error?

The `list` `pics.ALL_ARROWS` has just 8 elements, indexed 0 through 7.

When your `index` variable reaches 8, it is *past the end* of the list!

**How can you keep `index` in the range 0 - 7?**

Here's an idea:

- Use *another variable* called `loops` to count the *total* number of repeats.
- Keep using `index` too, but *reset* it back to 0 when it reaches 8.

**NOTE:** Beware the difference between `=` `assignment` and `==` `comparison operations`!

### Mission 10

Select the three correct statements about functions. **+5 XP**

They ensure values are always increasing monotonically.

You can reuse them multiple times.

They help keep your code organized.

It is easier to make a change in one place than in repeated code.

*Answered 3 of 3*

What does the `time.ticks_diff(end, start)` function do? **+5 XP**

- It predicts the end of time given a start time.
- It returns the time difference between start and end.
- It changes the clock on your computer by the diff.

```
reaction_time = time.ticks_diff(end_time, start_time)
```

### Mission 11

If the accelerometer returns an (x, y, z) tuple then what direction force is the `d` variable below? **+5 XP**

```
val = accel.read()
d = val[1]
```

- z
- x
- y

### Mission 12

What are the colors of the 4 CodeX pixels after running this **+5 XP** code?

```
from codex import *
pixels.set([BLUE, BLUE, BLUE, BLUE])
pixels.set(2, RED)
```

- BLUE, RED, BLUE, BLUE
- BLUE, BLUE, RED, BLUE
- BLUE, BLUE, BLUE, BLUE
- OFF, OFF, RED, OFF

### Mission 13

Which of the following is true?

+5 XP

- A variable which is *assigned to* inside of a *function* is considered **local**, unless it is explicitly named in a `global` statement..
- Any variable used inside of a function is considered **local** to that function, unless it is explicitly named in a `global` statement.
- A *global* variable is not visible inside a function, unless it is explicitly named in a `global` statement.

The following function reports an error on Line 2.

+5 XP

```
1 def add_counter(amount, limit):  
2     if counter < limit:  
3         counter = counter + amount
```

What is the expected error message?

- Undefined variable: counter
- Global variable cannot be used in comparison.
- Local variable referenced before assignment

What is displayed by the following code?

+5 XP

```
for i in range(4):  
    display.print(i, end=',')
```

- 0,1,2,3    1,2,3,4    0,1,2,3,4

What is displayed by the following code?

+5 XP

```
for i in range(1, 5):  
    display.print(i, end=',')
```

- 0,1,2,3,4,5    1,2,3,4    1,2,3,4,5

What is a **blocking** function?

+5 XP

- A function that creates square 2D shapes or 3D cubes.
- A function composed of blocks.
- A function that returns immediately, even if that would block it from completing its goal.
- A function that blocks program execution and does not return until it has completed its goal.

`sleep(5)` will delay the program for 5 seconds.

+5 XP

Is it a "blocking function"?

- Yes    No

What is the final value of `toggle` after this code runs?

+5 XP

```
toggle = False

toggle = not toggle
toggle = not toggle
toggle = not toggle
```

False  True  3

## Mission 14

Answer the questions below about the following code:

```
display.set_pixel(200, 200, CYAN)
c = display.get_pixel(200, 200)
print("Color = ", c)
```

Where on the CodeX `display` does the CYAN pixel appear?

+5 XP

Upper Right  Upper Left  Lower Right  Lower Left

What is printed on the `console`?

+5 XP

- You either need to consult the [RGB Colors](#) documentation, or just try it for yourself!

Color = (0, 255, 255)  Color = (0, 252, 248)  Color = (144, 210, 48)

Color = (0, 0, 0)

The following code draws a dashed line across the screen.

```
for d in range(0, 240, 20):  
    for x in range(d, d + 10):  
        display.set_pixel(x, 120, WHITE)
```

What is the orientation of the *dashed line*?

+5 XP

Diagonal

Vertical

Horizontal

How many pixels *long* is each dash?

+5 XP

10

1

40

20

Mission 15

How many *milliseconds* are in 1 second?

+5 XP

- 0.001  1000  1 million  100

Say the ball moves at a velocity of  $\frac{1}{2}$  pixel per millisecond.

+5 XP

- How far would it move in 10 milliseconds?

- 20 pixels  5 pixels  50 pixels  10 pixels

Your *game loop* uses a global variable `ball_v` for the ball's velocity. Where is this variable initialized? (*first assigned to*)

+5 XP

- Inside the `draw_ball()` function.  At the beginning of the game loop.

- Inside the `serve_ball()` function.

What do the letters D T stand for in the variable `dt` ?

+5 XP

- "delta tricep"  "delta time"  "dog tired"  "data test"

- "difference time"

What would you expect the value of `dt` to be after the following code runs?

+5 XP

```
elapsed_ms()
time.sleep_ms(42)
dt = elapsed_ms()
```

- Error: No target for assignment.  40  42  10

How does the `function elapsed_ms()` "remember" the millisecond value the last time it was called? **+5 XP**

- All `variables` inside a function are preserved across calls.
- Computers don't remember. Eschew anthropomorphism.
- The `global ms` saves milliseconds between function calls.

Which *three* of the following `comparisons` are `True`? **+5 XP**

- `"one" == 1`
- `(1, 2, 3) == (3, 2, 1)`
- `10 > 9`
- `1 < 0`
- `(1, 2, 3) == (1, 2, 3)`
- `("Right", "On") == ("Right", "On")`

Answered 1 of 3

What's the purpose of the `sound_cut` variable in your Handball program? **+5 XP**

- To count down the milliseconds till you turn off the sound.
- It is the cut-off frequency of the sound.
- To count up seconds until sound stops.

What is `max(min(3, 2), 1)` **+5 XP**

- 4
- 2
- 3
- 1



What is the value of `count` after the following code runs?

+5 XP

```
count = 0
x = 0
while x < 5:
    x = x + 1
    if x == 2:
        continue
    count = count + 1
```

5

4

3

### Mission 16

For a card game, I've created the following matrix. The cards will be laid out in rows and columns.

- Face up → `True`
- Face down → `False`

```
cards = [
    [True, False, True],
    [True, True, True],
    [False, False, True],
    [True, False, False],
]
```

How many *rows* are in the `cards` matrix?

+5 XP

4

2

3

How many *columns* are in the `cards` matrix?

+5 XP

4

2

3

Is the card at `cards[2][1]` *Face up* or *Face down*?

+5 XP

Face down  Face up

What is the value of `my_list` after the following code runs?

+5 XP

```
my_list = [5, 4, 9]
my_list.append(2)
```

`[5, 4, 9, [2]]`  `[5, 4, 9], [5, 4, 9]`  `[2, 5, 4, 9]`

`[5, 4, 9, 2]`

What is the value of `result` after the following code runs?

+5 XP

```
result = 4 + 2 * 3 - 1
```

23  12  17  9

What is the result of the following code?

+5 XP

```
my_list = [14, 16, 18]
result = my_list[1.5]
```

17  TypeError  IndexError  18  16