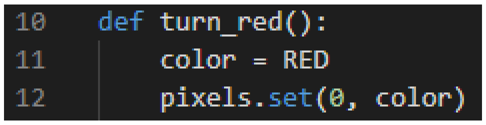
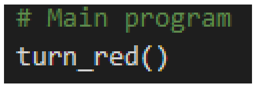
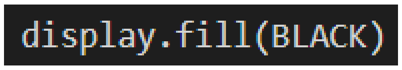
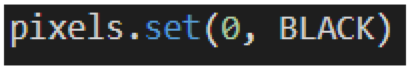
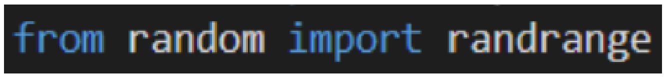
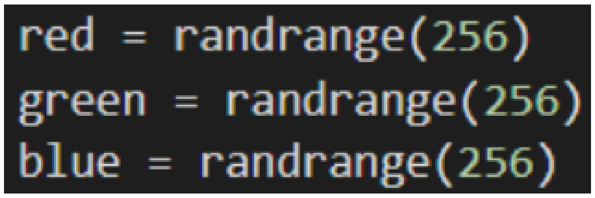
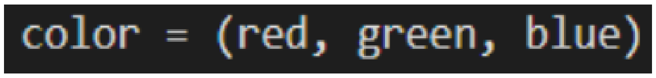
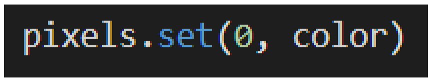
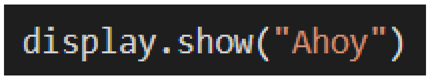
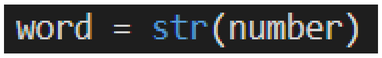
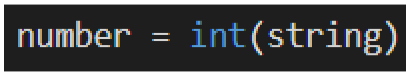
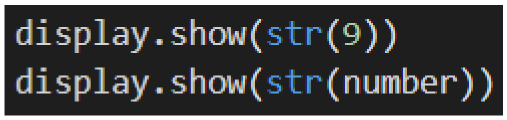
AP CSP CodeX Python Code By Mission

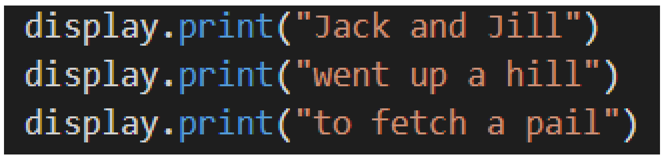
Mission 2 – Introducing CodeX	
Import codex	`from codex import *`
Display a built-in image	`display.show(pics.HEART)`
All built-in images:	• `pics.HEART` • `pics.HEART_SMALL` • `pics.MUSIC` • `pics.HAPPY` • `pics.SAD` • `pics.SURPRISED` • `pics.ASLEEP` • `pics.TARGET` • `pics.TSHIRT` • `pics.PLANE` • `pics.HOUSE` • `pics.TIARA` • `pics.ARROW_N` • `pics.ARROW_NE` • `pics.ARROW_E` • `pics.ARROW_SE` • `pics.ARROW_S` • `pics.ARROW_SW` • `pics.ARROW_W` • `pics.ARROW_NW`
Mission 3 – Light Show	
Turn on ONE pixel (pixels are numbered 0, 1, 2, 3)	`pixels.set(0, GREEN)`
All built-in colors	BLACK, BROWN, RED, ORANGE, YELLOW, GREEN, BLUE, PURPLE, GRAY, WHITE, CYAN, MAGENTA, PINK, LIGHT_GRAY, DARK_GREEN, DARK_BLUE
Import time to use sleep()	`from time import sleep` or `from time import *` (either will work)
Cause a pause or delay in the code	`sleep(1)` (this will pause for 1 second)
Define a variable (assign a value)	`delay = 1` or `color = RED`
Use a variable with sleep()	`sleep(delay)`
Instructions for using the debugger are included in this mission (Objectives 5 & 6)	
Defining Functions	

Define a function	 `def turn_red():`
	`    color = RED`
	`    pixels.set(0, color)`

Call a function	 `# Main program` `turn_red()`
Mission 3 RGB Colors	
Clear the display	 `display.fill(BLACK)`
Clear a pixel (turn black)	 `pixels.set(0, BLACK)`
Import random module	 `from random import randrange`

Assign a random color (RGB)	 `red = randrange(256)`
	`green = randrange(256)`
	`blue = randrange(256)`

Assign color from RGB	 `color = (red, green, blue)`
Use color variable	 `pixels.set(0, color)`
Mission 4 – Display Games	
Display a word	 `display.show("Ahoy")`
Convert number to string	 `word = str(number)`
Convert string to number	 `number = int(string)`
Display a number	 `display.show(str(9))` `display.show(str(number))` Can be a literal value (9) Or a variable (number)

Display more than one line	 `display.print("Jack and Jill")`
	`display.print("went up a hill")`
	`display.print("to fetch a pail")`
	use display.**print()** instead of display.**show()**

If / else statement (branching)	`pressed = True`
	`if pressed:`
	`    pixels.set(0, GREEN)`
	`else:`
	`    pixels.set(0, RED)`
		Look for **:** and the indenting -- very important!

Assign a value to a button press (True or False)	`pressed = buttons.is_pressed(BTN_A)`
	`pressed = buttons.was_pressed(BTN_B)`
		Checks if currently pressed
		
		Checks if was pressed since last time

Mission 5 – Micro Musician

Play a built-in audio clip	`audio.mp3("sounds/welcome")`
All built-in audio clips	

a.mp3	eight.mp3	off.mp3	six.mp3
africa.mp3	five.mp3	okay.mp3	techstyle.mp3
b.mp3	four.mp3	on.mp3	ten.mp3
bohemia.mp3	funk.mp3	one.mp3	three.mp3
button.mp3	led.mp3	power.mp3	two.mp3
codetrek.mp3	left.mp3	right.mp3	up.mp3
codex.mp3	mic.mp3	roll.mp3	welcome.mp3
display.mp3	nine.mp3	seven.mp3	yes.mp3
down.mp3	no.mp3	shire.mp3	zero.mp3

Mission 6 - Heartbeat

Infinite while loop	`while True:`
	`    # Indent code to loop`
	`    display.show(pics.HEART)`
	`    sleep(delay)`

Break out of a loop
Can be any button
	`if buttons.was_pressed(BTN_A):`
	`        break`

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

Decrement
With if statement
	`if buttons.was_pressed(BTN_B):`
	`    delay = delay - 0.2`

Mission 7 - Personal Billboard	

Compare a variable to a specific value	`if choice == 0:`
	`    # do something`

Last index of a list	`LAST_INDEX = len(my_list) - 1`

List index wrap around (end back to beginning)	`if buttons.was_pressed(BTN_L):`
	`    choice = choice - 1`
	`    if choice < 0:`
	`        choice = LAST_INDEX`

List index wrap around (beginning back to end)	`if buttons.was_pressed(BTN_R):`
	`    choice = choice + 1`
	`    if choice > LAST_INDEX:`
	`        choice = 0`

Define (create) a list	`my_list = [pics.HAPPY,`
	`            pics.SAD,`
	`            pics.SURPRISED,`
	`            pics.ASLEEP]`
	
	`my_list = [pics.HAPPY, pics.SAD, pics.SURPRISED, pics.ASLEEP]`

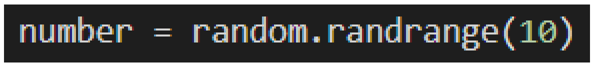
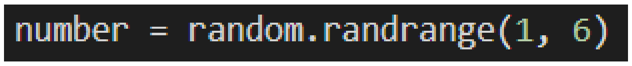
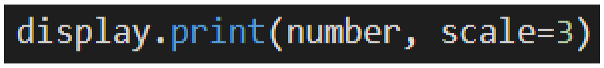
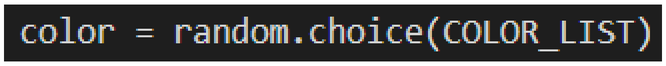
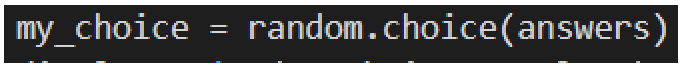
Access an item from the list	`index = 3`
	`my_item = my_list[index]`
	
	`my_item = my_list[2]`

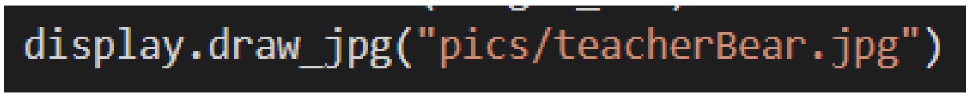
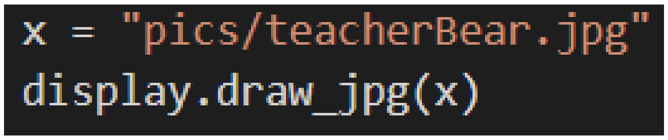
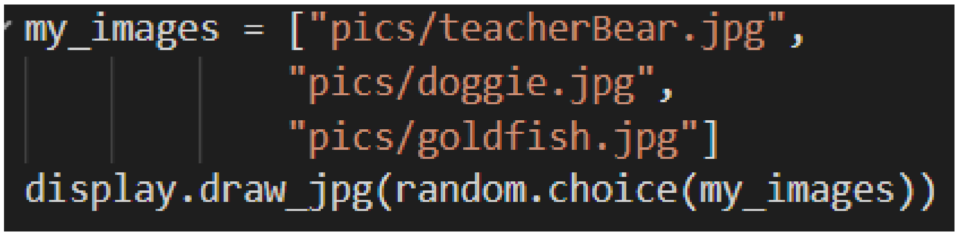
Last index	`LAST_INDEX = len(my_list) - 1`

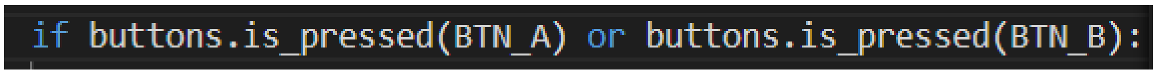
Get the data type of a variable (can also use console panel)	`>>> type(7)`
	`<class 'int'>`
	`>>> type(1.15)`
	
	`my_type = type(7)`
	`if type(my_item) == tuple`

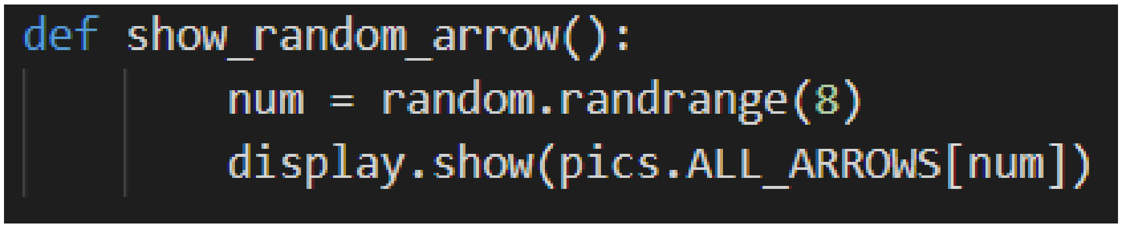
Fill screen with a color	`display.fill(RED)`
	`display.fill(my_color)`

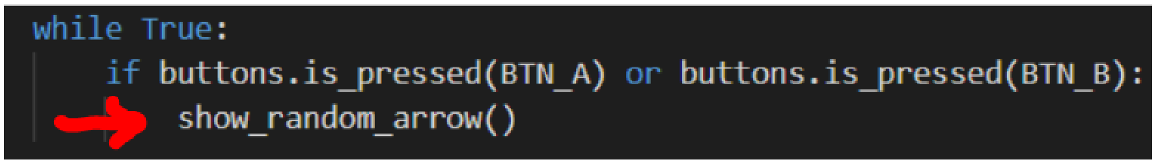
Mission 8 - Answer Bot	
Import random module	`import random`

Generate a random integer	 `number = random.randrange(10)` gives a number between 0 and 9  `number = random.randrange(1, 6)` gives a number between 1 and 5 ** default starting value is 0 unless specifically stated. Integers will go from the starting value to one less than the ending value.
Change the size of text	 `display.print(number, scale=3)` scale adjusts the size of the text. If the scale is too big, the text will appear as gibberish or shapes on the display screen. scale=1 is the default size.
Select a random number from a list	 `color = random.choice(COLOR_LIST)`  `my_choice = random.choice(answers)`
Lists with JPG images - Optional Lesson - Adding JPG images	

Displaying a JPG image	 `display.draw_jpg("pics/teacherBear.jpg")`
	 `x = "pics/teacherBear.jpg"`
	`display.draw_jpg(x)`
	 `my_images = ["pics/teacherBear.jpg",`
	`             "pics/doggie.jpg",`
	`             "pics/goldfish.jpg"]`
	`display.draw_jpg(random.choice(my_images))`

Mission 9 - Game Spinner	
Using a logical operator:	 `if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):`

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

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

Finite loop with condition
 	
(increment the control variable)
	`while index < 8:`
	`    my_arrow = pics.ALL_ARROWS[index]`
	`    display.show(my_arrow)`
	`    sleep(0.1)`
	`    index = index + 1`

Finite loop with condition and list wrapping	`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`

Mission 10 - Reaction Tester	
Turn off all pixels using a list	`pixels.set([BLACK, BLACK, BLACK, BLACK])`
Turn all pixels a color using a list	`pixels.set([GREEN, GREEN, GREEN, GREEN])`
Clear the display	`display.clear()`
Get current clock time	`start_time = time.ticks_ms()`
Find the difference between two clock times	`reaction_time = time.ticks_diff(end_time, start_time)`
Reset the button state	`buttons.was_pressed(BTN_A)`
Mission 11 - Spirit Level	
Math module	`import math` used for math operations, like math.pi, math.asin, etc.
Get values from the accelerometer	`val = accel.read()`

Get a single value from the accelerometer	`val = accel.read()`
	`tilt_x = val[0]`

Change display color	`display.fill(WHITE)`

Draw a line	`display.draw_line(x1, y1, x2, y2, color)`
	`display.draw_line(CENTER, 0, CENTER, 105, BLACK)`

Draw a circle	`display.draw_circle(x, y, radius, color)`
	`display.draw_circle(x, CENTER, 15, ORANGE)`

Mission 11 Remix -- these commands are optional but can be used in the remix projects	
Filled in circle	`display.fill_circle(CENTER, CENTER, 15, RED)`
Display text with a specific location	`display.draw_text(str(score), x=20, y=20, scale=3, color=BLACK)`
Mission 12 - Night Light	
Read from the light sensor	`value = light.read()`
Set all pixels the same color	`pixels.fill(WHITE)` -- on `pixels.fill(BLACK)` -- off

Adjust brightness of pixels	`pixels.fill(WHITE, brightness=20)`
	`pixels.fill(WHITE, brightness = level)`