

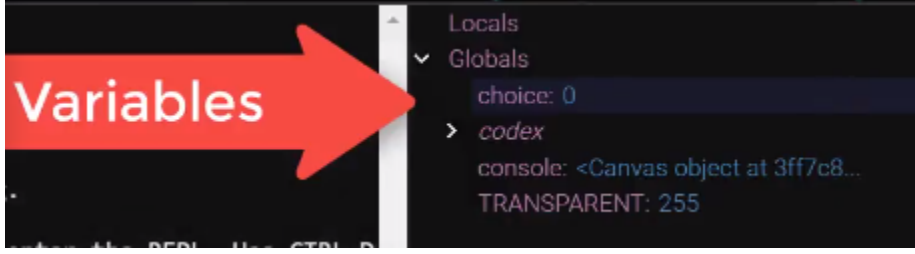


CodeBot Python Code By Mission

Mission 3 – Light the Way	
Import botcore library (all pre-built code)	<pre>from botcore import *</pre>
Turn on user led	<pre>leds.user_num(0, True)</pre> (turns on LED 0) <pre>user_num(num, is_on)</pre> (in general)
Comment (starts with #)	<pre># Turn on LEDs 0, 3, 4, 7</pre>
Set LEDs with a binary pattern (no _num, use 0b prefix)	<pre>leds.user(0b11111111) leds.ls(0b11111)</pre>
Import from time module	<pre>from time import sleep</pre>
Delay program execution (slow down the computer)	<pre>sleep(0.2)</pre>
Set LEDs with a decimal	<pre>leds.user(0)</pre>
Mission 4 – Get Moving!	
Enable CodeBot's motors	<pre>motors.enable(True)</pre>
Apply power on the wheels to spin	<pre>motors.run(LEFT, 30) motors.run(RIGHT, -30)</pre> <p>(one positive and one negative speed will result in a spin)</p>
Apply power on the wheels to move in a curve	<pre>motors.run(LEFT, 50) motors.run(RIGHT, 40)</pre> (both positive but different speeds)
Play a tone using the speaker for a specific amount of time	<pre>spkr.pitch(440) sleep(0.5)</pre> pitch is in Hz
Turn off the tone	<pre>spkr.off()</pre>
Mission 5 – Dance Bot	

<p>While loop for repeating code while a condition is true</p>	<pre>count = 0 while count < 8: leds.user_num(0, True) sleep(0.1) leds.user_num(0, False) sleep(0.1) count = count + 1</pre>
<p>Assign a value to a variable (use the assignment operator =)</p>	<pre>count = 0 count = count + 1</pre>
<p>Initialize a variable</p>	<pre>count = 0</pre>
<p>increment (update) a variable</p>	<pre>count = count + 1</pre>
<p>Use the debugger</p>	<p> DEBUG then use the  STEP IN button to <i>step</i> through your code.</p>
<p>Use debugger to view variables (open the console panel while debugging)</p>	
<p>Print a message on the Console</p>	<pre>print(count)</pre> <p>the message displays in the console, so the panel must be open to see the message.</p>
<p>For loop with range() function</p>	<pre>for count in range(8): print(count) leds.user_num(0, True) sleep(0.1) leds.user_num(0, False) sleep(0.1)</pre>
<p>Use a loop variable as part of an instruction</p>	<pre>leds.user_num(count, True)</pre>

For loop with a full range() function to count backwards	<pre>for count in range(6, -1, -1): print(count) leds.user_num(count, True) sleep(0.1) leds.user_num(count, False)</pre>
Define a function	<pre>def sweep_left(): for count in range(8): print(count) leds.user_num(count, True) sleep(0.1) leds.user_num(count, False)</pre>
Call a function	<pre>sweep_left() sweep_right()</pre>
Check the current state of a button	<pre>buttons.is_pressed(0)</pre>
Check the last state of a button	<pre>buttons.was_pressed(0)</pre>
not operator (result is the opposite)	<pre>not buttons.was_pressed(0)</pre> <p>If buttons.was_pressed(0) is True, the result will be False. If buttons.was_pressed(0) is False, the result will be True.</p>
Mission 6 – Robot Metronome	
Infinite loop (Does not stop because the loop condition is always True. The code must be manually stopped by clicking the STOP button.)	<pre>while True: leds.user(0b11111111) spkr.pitch(784) sleep(0.1) spkr.off() leds.user(0) sleep(0.9)</pre>
Assign a value to a variable	<pre>pause = (60 / tempo) - beat_duration</pre>
Boolean variable	<pre>sound_on = True</pre>
If statement (branching)	<pre>if sound_on: spkr.pitch(784)</pre>

	<pre>if buttons.was_pressed(0): sound_on = False</pre>
Toggle a Boolean variable (use the 'not' operator)	<pre>if buttons.was_pressed(0): sound_on = not sound_on</pre>
Use the opposite of a Boolean to turn on/off an LED	<pre>leds.pwr(not sound_on)</pre>
Bit shift operator (shift the bit to the left by the index)	<pre>leds.ls(1 << tempo_select)</pre>
Define a List	<pre>tempo_list = [50, 70, 100, 140, 180]</pre>
Access an item from a list using a variable for the index	<pre>tempo = tempo_list[tempo_select]</pre>
len() function (number of items in a list)	<pre>len(tempo_list)</pre>
Wrap-around an index variable when the length of the list is reached	<pre>if tempo_select == len(tempo_list): tempo_select = 0</pre>
Mission 7 – Line Sensors	
Read the brightness level detected by a line sensor	<pre>left = ls.read(0) right = ls.read(4) reading = ls.read(num)</pre>
Print numbers with a , in between	<pre>print(left, right, sep=',')</pre>
Compare a variable to see if it is within two values	<pre>if 3547 < left < 3567:</pre>
Break out of a loop	<pre>break</pre> (can only be used inside a loop)
Stop the motors	<pre>motors.enable(False)</pre>
Define a constant (all CAPS)	<pre>MIN_DIFF = 100</pre>
Find the absolute value	<pre>abs(left - right)</pre>

Define a matrix	<pre>sensor_data = [['N', 3557], ['NE', 4080], ['E', 286], ['SE', 868], ['S', 1391], ['SW', 1944], ['W', 2481], ['NW', 3019]]</pre>
Prompt for input from the console	<pre>target_direction = input("Enter target direction: ")</pre>
Function return	<pre>return d[0]</pre>
Call a function with a return	<pre>found = find_name(left)</pre>
Mission 8 – Boundary Patrol	
Define a threshold	<pre>threshold = 2000</pre>
Use a threshold in an if statement	<pre>if val > threshold: break</pre>
Default parameter	<pre>def go(left, right, delay=0):</pre> default parameter is delay <pre>def backturn(turn_power=50):</pre> default parameter is turn_power
Use a default parameter if a values is assigned	<pre>if delay: sleep(delay)</pre>
Docstring (used for commenting functions)	<pre>'''A function to simplify motor commands'''</pre>
Define an empty list	<pre>sensors = []</pre>
Add an item to a list	<pre>sensors.append(is_line)</pre> add item to the end of the list
Use a Boolean list to control LEDs	<pre>leds.ls(vals)</pre> Where vals is a list of Booleans. Example: vals = [True, True, False, False, False]
any() function – returns True if any item in the list is True	<pre>if any(vals):</pre>

If statement with multiple branches	<pre> if vals[0] and not vals[4]: backturn(30) elif vals[4] and not vals[0]: back_turn(-30) else: back_turn() </pre>
Mission 9 – Line Following	
List comprehension (shortcut for populating a list)	<pre> even_numbers = [x * 2 for x in range(5)] [ls.read(i) > 2000 for i in range(5)] </pre>
Function that reads all line sensors, compares the reading to a threshold and returns a list of Booleans	<pre> vals = ls.check(2000) </pre>
Not equals operator	<pre> if vals != prev_vals: </pre>
Using a logical operator in a compound condition	<pre> elif vals[1] or vals[2] or vals[3]: </pre>
Defining a dictionary This example uses a tuple as the key and an integer as the value.	<pre> ls_err = { (0,0,1,0,0) : 0, (0,1,1,1,0) : 0, } </pre>
Look up a value from a dictionary using a key	<pre> err = ls_err[vals] </pre>
get() method	<pre> err = ls_err.get(vals, err) </pre>
PID controller	<pre> def apply_control(err): Kp = 0.1 steering = err * Kp drive(SPEED, steering) </pre>
Mission 10 – Fido Fetch	
Look up a value of a key:value pair	<pre> value = dictionary[key] response = commands[command] </pre>
Add a key:value pair after the dictionary is defined	<pre> dictionary['key'] = value commands['come'] = [30, 30] </pre>

Iterate over a dictionary Print all keys to the console:	<pre>for k in commands: print(k)</pre> k stands for 'key'
Remove an item from the dictionary	<pre>del_key = input("Command to Forget: ") del commands[del_key]</pre>
Mission 11 – Airfield Ops	
Increment a counter	<pre>count = count + 1</pre>
Augmented assignment (shorter way to write common expressions)	<pre>count += 1</pre>
Integer division	<pre>num_leds_on = (count * NUM_USER_LEDS) // TOTAL_LINES</pre>
Use multiplication on a list with a Boolean	<pre>progress = [True] * num_leds_on</pre>
Modulo division	<pre>remainder = count % 8</pre>
Power (exponent) operator	<pre>marker_dash = 2**next_marker</pre>
Mission 12 – King of the Hill	
Read the current axis values of the accelerometer	<pre>x, y, z = accel.read()</pre>
Print the 3-axis values to the debug console	<pre>accel.dump_axes()</pre> The display: <pre>X=-1044, Y=4261, Z=-15786</pre>
Calculate radians	<pre>pitch = math.asin(y/16384)</pre>
Convert radians to degrees	<pre>pitch = pitch * 180 / math.pi</pre>
Round a value to the nearest integer	<pre>pitch = round(pitch)</pre>
Cascaded assignment (Sets two variables to the same value)	<pre>bars_left = bars_right = ''</pre>
Formatted string with replacement fields and format specifiers	<pre>dash = "[-90 {:>30} }:+3} {:<30} +90]".format(bars_left, pitch, bars_right)</pre>
Add degree character	<pre>+90\xB0</pre>

Mission 13 – Going the Distance	
Read the wheel encoder sensor	<pre>val = enc.read(LEFT)</pre>
Integer division	<pre>n = val//100</pre>
Create a repeated character string	<pre>print(val, n * '*')</pre> creates “n” number of “*”
Use a Boolean to indicate a slot	<pre>val = enc.read(LEFT) is_slot = val > SLOT_THRESHOLD</pre>
Calculate wheel circumference	<pre>WHEEL_CIRC_CM = math.pi * WHEEL_DIAM_CM</pre>
Calculate counts for given distance	<pre>cm * (COUNTS_PER_REV / WHEEL_CIRC_CM)</pre>
Get + or - direction	<pre>direction = deg / abs(deg)</pre>
Current time-tick count in milliseconds	<pre>t_poll = ticks_ms() + POLL_MS</pre> current time is ticks_ms()
f-string formatting	<pre>print(f'speed = {speed} cm/s')</pre>
Calculate distance for given counts	<pre>counts * WHEEL_CIRC_CM / COUNTS_PER_REV</pre>
Feedback loop with code	<pre># Feedback Loop! Adjust power in proportion to err err = target_speed - cur_speed power += err * Kp motors.run(LEFT, power) motors.run(RIGHT, power)</pre>
Mission 14 – Music Box	
Iterate over a list of strings	<pre>notes = ['C', 'C', 'G', 'G', 'A', 'A', 'G', 'F', 'F', 'E', 'E', 'D', 'D', 'C'] for note in notes: f = freqs[note] spkr.pitch(f) sleep(beat_duration)</pre>
Split a string into a list	<pre>text = 'EEEEEEEGCDEFFFFFEEEEEDEDDG' notes = text.split()</pre>
Open a file	<pre>f = open('old_man_song.txt', 'r')</pre>

Read a file	<pre>text = f.read()</pre>
Close a file	<pre>f.close()</pre>
Open a new file	<pre>f = open('my_song.txt', 'w')</pre>
Write to a file	<pre>f.write('C D E C C D E C E F G E F G')</pre>
Flush a file (all data is saved)	<pre>f.flush()</pre>
Matrix of notes and beats	<pre>song = [['E', 1], ['D', 1], ['C', 1]]</pre>
Unpack a matrix using a loop	<pre>for note, beats in song: f = freqs[note] spkr.pitch(f) sleep(beats * beat_duration)</pre>
Convert a string to an integer	<pre>beats = int(str_beats)</pre>
Get a list of strings, one from each line in a file	<pre>f = open("rain_rain_song.csv", "r") file_lines = f.readlines()</pre>
Split a list of strings into a list of lists	<pre>song = [] for line in file_lines: note_beat = line.split(",") song.append(note_beat)</pre>
Mission 15 – Cyber Storm	
Open a file using a 'with' block	<pre>with open(email_file, 'r') as f: file_contents = f.readlines() print(file_contents)</pre>
Strip whitespace from a line	<pre>line.strip()</pre>
Concatenate (join) a string	<pre>email = email + line.strip()</pre>

<p>Create a dictionary with the heading</p>	<pre>with open(email_file, 'r') as f: email = {} for line in f: clean_line = line.strip() if line.startswith('Date: '): email['date'] = clean_line[6:] elif line.startswith('From: '): email['from'] = clean_line[6:] elif line.startswith('To: '): email['to'] = clean_line[4:] elif line.startswith('Subject: '): email['subject'] = clean_line[9:] print(email)</pre>
<p>Find the body of the email (Use after the heading if statements)</p>	<pre>elif line.strip() == '': break email['body'] = f.read()</pre>
<p>Look for a word in a string</p>	<pre>if 'virus' in email['body']:</pre>
<p>Replace a word in a string</p>	<pre>email['body'] = email['body'].replace('virus', 'REMOVED')</pre>
<p>Import a library to use os functions and methods</p>	<pre>import os</pre>
<p>Check to see if a file exists</p>	<pre>if not os.path.exists('blocklist.csv'): print("Creating blocklist!")</pre>
<p>Write a new item to a file</p>	<pre>f.write(bl_entry)</pre>
<p>Remove a file</p>	<pre>if sender in blacklist: os.remove(filename)</pre>