

# GoPiGo Tutorials and Resources

## Contents

GoPiGo Tutorials and Resources .....	1
Run Python Code from the Terminal .....	2
GoPiGo Library Documentation.....	2
GoPiGo Tutorials .....	2
Modular Robotics GoPiGo Documentation .....	2
Code Examples and Projects on the GoPiGo .....	2
Sample GoPiGo Python3 Programs in WNCCNASA GitHub .....	3
Learning Python .....	3
General Raspberry Pi Resources .....	3
Easy Movement Tutorial .....	3
Driving School Remote Control Console.....	5
Requirements .....	6
Stage 1 .....	6
Stage 2 .....	6
Stage 3 .....	7
Driving School Remote Control Tkinter.....	7
Draw It .....	7
Pseudocode .....	8
Tkinter Driving School Sample Code .....	8
Random Numbers.....	11
Obstacle Avoidance .....	11
Minimum Requirements.....	11

**NOTE:** All Python code is compatible with Python 3.5. This is the current version of Python on GoPiGo Raspbian for Robots. If you build a Raspberry Pi OS (Currently Buster) and install the GoPiGo3 software, you will be using Python 3.7.

## Run Python Code from the Terminal

Run all Python scripts from the terminal.

```
python3 <scriptname>
```

## GoPiGo Library Documentation

This is the location for the documentation on the GoPiGo Python library.

<https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#easygopigo3>

## GoPiGo Tutorials

- [Video Streaming Robot](#)
- [Basic Robot Control](#)
- [GoPiGo Scratch Control Panel](#)
- [Build Your Own GoPiGo License Plate](#) (3D Printed)

## Modular Robotics GoPiGo Documentation

This has the latest documentation for the GoPiGo3. It has basic tutorials and a reference to get you started programming the robot.

<https://readthedocs.org/projects/gopigo3/downloads/pdf/latest/> (pdf version)

<https://gopigo3.readthedocs.io/en/latest/> (Web version)

## Code Examples and Projects on the GoPiGo

On your GoPiGo3, there are some code examples.

- 1) In File Manager → go to **\Dexter\GoPiGo3** or **\Dexter\GoPiGo**
- 2) **\Dexter\GoPiGo3\Software\Python** is a good place to start.

GoPiGo projects explained in more detail.

<https://www.dexterindustries.com/GoPiGo/projects/python-examples-for-the-raspberry-pi/>

GoPiGo3 Python Tutorials

<https://edu.modrobotics.com/course/view.php?id=16>

## Sample GoPiGo Python3 Programs in WNCN NASA GitHub

There are sample Python3 programs in the GitHub Code folder. They use the Easy GoPiGo library. Movements and the led's on the GiPiGo are a good place to start programming.

<https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#easygopigo3> (Easy GoPiGo Library)

## Learning Python

There are hundreds of tutorials and videos on learning Python. Here is a good resource for learning Python.

<https://www.w3schools.com/python/default.asp>

## General Raspberry Pi Resources

Run this command at the terminal to determine which Raspberry Pi model you have.

```
cat /sys/firmware/devicetree/base/model
```

MagPi magazine is free on pdf. They have several good general books on the Raspberry Pi.

1. <https://magpi.raspberrypi.org/books> (Raspberry Pi Books)
2. <https://magpi.raspberrypi.org/issues> (MagPi Magazine)

## Easy Movement Tutorial

It is time to do some tutorials to learn the GoPiGo library better.

Learning points

- Functions, Loops, Movement, GoPiGo Blinkers

```

1  #!/usr/bin/env python3
2  """
3      Name: easy_movement.py
4      Author: William A Loring
5      Created: 09-18-21 Revised:
6      Purpose: Demonstrate a sampling of GoPiGo dead reckoning movements
7  """
8  # This uses the EasyGoPiGo3 library. You can find more information on the li
9  # here: https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#
10
11 # Import the time library for the sleep function
12 import time
13 # Import GoPiGo3 library
14 from easygopigo3 import EasyGoPiGo3
15
16 # Create an instance of the GoPiGo3 class
17 # GPG is the GoPiGo3 object used to access methods and properties
18 gpg = EasyGoPiGo3()
19
20
21 #----- SQUARE RIGHT -----#
22 def square_right(distance):
23     """
24         Drive a right square based on the distance argument
25     """
26     # Loop four times, Loop starts at 0,
27     # Ends at 1 less than the last number
28     # The loop increments 0, 1, 2, 3
29     print("Square Right")
30     for x in range(0, 4):
31         # Print the loop counter
32         print(x)
33         gpg.led_off("right")
34         gpg.drive_inches(
35             distance, # How far to drive in inches
36             True      # Blocking, nothing else can happen while moving
37         )
38         gpg.led_on("right")
39         # Turn right 90 degrees, positive number is right
40         gpg.turn_degrees(90)
41     # Turn both blinkers off
42     gpg.led_off("right")
43     gpg.led_off("left")
44
45 #----- SQUARE LEFT -----#
46 def square_left(distance):
47     """
48         Drive a left square based on the distance argument
49     """
50     print("Square Left")
51     for x in range(0, 4):
52         print(x)
53         gpg.led_off("left")
54         gpg.drive_inches(distance, True)
55         gpg.led_on("left")
56         # Turn left 90 degrees, - is left
57         gpg.turn_degrees(-90)
58     gpg.led_off("left")
59
60
61

```

```

62 #----- GOPIGO WAGGLE -----#
63 def waggle():
64     """ Waggle back and forth """
65     print("Waggle")
66     for x in range(0, 4):
67         print(x)
68         gpg.led_on("left")
69         gpg.turn_degrees(-10)
70         gpg.led_off("left")
71         gpg.led_on("right")
72         gpg.turn_degrees(10)
73         gpg.led_off("right")
74     # Turn off both blinkers
75     gpg.led_off("right")
76     gpg.led_off("right")
77
78
79 def main():
80     """ Main Program Entry Point """
81     # Drive a 5" square turning left
82     square_left(5)
83
84     # Turn left to reverse the square
85     print("Turn Left 90")
86     gpg.turn_degrees(-90)
87
88     # Drive a 5" square turning right
89     square_right(5)
90
91     print("Spin left.")
92     gpg.spin_left()
93     time.sleep(1)
94
95     # Waggle back and forth
96     waggle()
97
98     print("Spin right.")
99     gpg.spin_right()
100    time.sleep(3)
101
102    print("Stop!")
103    gpg.stop()
104    print("Done!")
105
106
107 # If a standalone program, call the main function
108 # Else, use as a module
109 if __name__ == '__main__':
110     main()

```

## Driving School Remote Control Console

You have seen this before in Intro to Robotics. Here it is again!

You will find all of the assignments from Intro to Robotics in the mBot folder in GitHub.

These can be used to see the shape that you want your code to be.

---

## Requirements

1. Each movement will have its own function. This is demonstrated in the example program: square\_left, square\_right, orbit\_right, and waggle.
2. **DRY:** Don't Repeat Yourself (Reuse functions, build bigger functions from smaller functions).
3. Create a console based menu to choose which function you wish to perform.

---

## Stage 1

1. **Square** - Trace the path of a square that is 1-foot square. It will start and end in the same place and the same orientation.
2. **Rectangle** - Trace the path of a rectangle that is 1-foot x 2-foot. It will start and end in the same place and the same orientation.
3. **Sentry** - Trace a 1-foot square around an object. Start the square one way, then turn around and go back the other way. Return to the beginning point and orientation.
4. **Retrace** - Move in a 1-foot square forward, then move in reverse to retrace that same square backwards to the beginning point and orientation. One solution would be to build a Reverse block that uses negative numbers for motor movement.
5. **ForwardReverse** - Move forward 12", turn 180°, move backwards 12" (which will be the same direction), turn 180° again, and then continue to move forward 12". The robot should move in one direction, but do part of the trip moving backwards.
6. **OrbitRight and OrbitLeft** - Orbit your GoPiGo around a central point.

---

## Stage 2

1. **Pentagon** - Move your robot in a 12" pentagon. Start and end in the same place and the same orientation.
2. **Octagon** - Move your robot in a 12" octagon. Start and end in the same place and the same orientation.
3. **Equilateral Triangle** - Move your robot in a 12" equilateral triangle. Start and end in the same place and the same orientation.

4. **5-Point Star** – Teach your robot to trace a 5-point 12” star. Start and end at the same location and orientation. Look up the inside angle and subtract from 180 degrees.
5. **3-PointTurn** - Using 3 or more turns, teach your robot how to make a 3-point turn, like a regular car. You don’t have to do curves, you can use straight angles if you wish.

---

### Stage 3

1. **Circle** - your robot will trace the path of a circle that is 1 foot in diameter. It will start and end in the same location, and in the same orientation.  
**HINT:** Adjust the power of your left and right motors to create a left half circle block and right half circle block. Put those together to make your curved shapes.
2. **S-Shape** – your robot will trace two half-circles to create an S-shaped curve. Your robot will start and end in the same orientation, and the two half-circles will be the same size.
3. **Figure-8** - Move in a figure-8 shape.



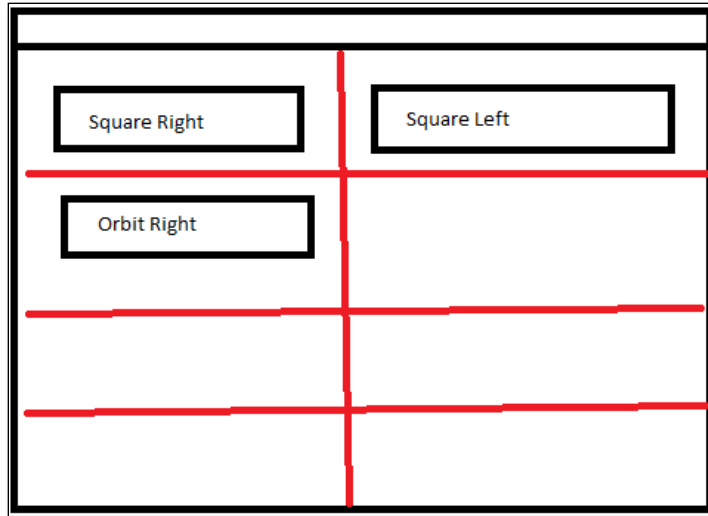
### Driving School Remote Control Tkinter

Copy your console menu based program and convert it to Tkinter OOP. You can use all of the functions you created with the console program as methods in your Tkinter Driving School.

---

### Draw It

When designing a GUI, start with a sketch. The example sketch shows a grid of 2 columns and multiple rows. You can have as many columns and rows as you wish.



---

## Pseudocode

```
Create interface  
Add functions from console program  
Attach methods to buttons
```

---

## Tkinter Driving School Sample Code

Some sample code to get you started.



```

1  #!/usr/bin/env python3
2  """
3      Name:      rc_tkinter_driving_school.py
4      Author:    William A Loring
5      Created:   12/04/2021
6      Purpose:   GoPiGo Tkinter based driving school program
7  """
8
9  # -----
10 # History
11 # -----
12 # Author      Date          Comments
13
14
15 from tkinter import *      # Import tkinter for GUI
16 from tkinter.ttk import *  # Add ttk themed widgets
17 import sys                # Used to as sys.exit() to exit the program
18 import easygopigo3 as easy # Import EasyGoPiGo3 library
19
20
21 class GoPiGoGUI:
22     def __init__(self):
23         """ Initialize the program """
24         self.gpg = easy.EasyGoPiGo3() # Create EasyGoPiGo3 object
25         self.gpg.set_speed(200)       # Set initial speed
26         self.window = Tk()            # Create Tkinter window
27         self.window.title("GoPiGo Driving School")
28         # Set the window size and location
29         # 375x320 pixels in size, location at 50x50
30         self.window.geometry("375x320+50+50")
31         self.create_widgets()         # Create and layout widgets
32         mainloop()                   # Start Tkinter program main loop
33
34 #----- CREATE WIDGETS -----#
35 def create_widgets(self):
36     """ Create and layout widgets """
37     # Create main label frame to hold remote control widgets
38     self.main_frame = LabelFrame(
39         self.window,
40         text="Driving School",
41         relief=GROOVE)
42
43     # Pack the frame to the width (X) of the window
44     self.main_frame.pack(fill=X, padx=10, pady=(10))
45     # Keep the frame size regardless of the widget sizes
46     self.main_frame.pack_propagate(False)
47
48     # Create widgets and attach them to the correct frame
49     btn_square = Button(
50         self.main_frame,
51         text="Square Right",
52         command=self.square_right)
53
54     btn_exit = Button(
55         self.main_frame,
56         text="Exit",
57         command=self.exit_program)
58

```

```

59         # Grid the widgets
60         btn_square.grid(row=0, column=0,)
61         btn_exit.grid(row=0, column=1)
62
63         # Set padding between frame and window
64         self.main_frame.pack_configure(padx=10, pady=(10))
65         # Set padding for all widgets in frames
66         pad = 10
67         for child in self.main_frame.winfo_children():
68             child.grid_configure(padx=pad, pady=pad)
69
70         #----- DRIVE RIGHT SQUARE -----#
71         def square_right(self):
72             for i in range(4):
73                 self.gpg.drive_inches(12)
74                 self.gpg.turn_degrees(90)
75
76         #----- EXIT PROGRAM -----#
77         def exit_program(self):
78             print("\nExiting")
79             sys.exit()
80
81
82         # Create program object
83         gopigo_gui = GoPiGoGUI()

```

Example run.



## Random Numbers

Create and display random numbers on your GoPiGo terminal.

## Obstacle Avoidance

Another blast from the past! Base this on the mBlock program samples in the repository.

---

### Minimum Requirements

Start with a simple Obstacle Avoidance program. Add in other requirements as you work on this project.

- Use the shape of your mBot mBlock or Arduino code to help your GoPiGo learn autonomous obstacle avoidance.
- Use your servo to look right and left to gauge the distance.
- Use random numbers to change the obstacle avoidance.