**Virtual Pi2Go Programming:**

**Python Statements and Pi2Go Commands**



**AIM:** After completing this worksheet you should be able to control your Pi2Go simulation using simple statements issued at the Python Command Line and be able to explain what a statement in a programming language is.

**You Need:** To complete this worksheet you need to have a virtual Pi2Go simulator (see WS1) and understand how to control the virtual Pi2Go from the IDLE command line (see WS2).

**If the simulator isn’t already running:**

Start it (see WS1) and select the Pi2Go robot and default\_world.xml

**Now open a *new* IDLE window**.

**Remember:** You can scroll back through commands in IDLE by typing *alt-p*

In order to control your robot simulation you need to *initialise it properly.* You did this as part of Worksheet 2.

To initialise your simulation you type

>> import simclient.simrobot as pi2go

>> pi2go.init()

If you have not already done this from worksheet 2. Do it now.

Now you can use simple commands to control your simulated robot. Try typing the following:

>> pi2go.forward(20)

>> pi2go.stop()



**Question 1:** What happens?

Each of these commands pi2go.init(), pi2go.forward(20), pi2go.stop() is a *python statement.* Statements are the basic commands that are used to build up programs. You have several commands available to you for operating the Pi2Go robot.

These commands are described in the box.

**IMPORTANT:** Note that parts in *italics* in the box are inputs to the commands which you have to select. So, for **pi2go.spinLeft(***speed***)** you have to replace *speed* with a number between 0 and 100. For instance, **pi2go.spinLeft(30)**

**pi2go.stop()**

**pi2go.forward(***speed***)**

where *speed* is a number and 0 <= *speed* <= 100

**pi2go.reverse(***speed***)**

where *speed* is a number and 0 <= *speed* <= 100

**pi2go.spinLeft(***speed***)**

where *speed* is a number and 0 <= *speed* <= 100

**pi2go.spinRight(***speed***)**

where *speed* is a number and 0 <= *speed* <= 100

**pi2go.turnForward(***leftSpeed, rightSpeed*

where *leftSpeed* and *rightSpeed* are numbers and

0 <= *leftSpeed, rightSpeed* <= 100

**pi2go.turnReverse(***leftSpeed, rightSpeed***)**

where *leftSpeed* and *rightSpeed* are numbers and

0 <= *leftSpeed, rightSpeed* <= 100

**pi2go.go(***leftSpeed, rightSpeed***)**

where *leftSpeed* and *rightSpeed* are numbers and

-100 <= *leftSpeed, rightSpeed* <= 100

**pi2go.go(***speed***)**

where -100 <= *speed* <= 100

**pi2go.setLED(***LED, Red, Green, Blue***)**

where *LED, Red, Green* and *Blue* are numbers and

0 <= *LED* <=3 and 0 <= *Red, Green, Blue* <= 4095

**pi2go.setAllLEDs(***Red, Green, Blue***)**

where *Red, Green* and *Blue* are numbers and 0 <= *Red, Green, Blue* <= 4095

Try typing the following:

>> pi2go.setLED(3, 0, 0, 1000)



**Question 2:** What happens?

pi2go.setLED controls two *light emitting diodes (LED)*. It takes four inputs: the number for the LED (0 is front, 1 is right, 2 is back and 3 is left) , followed by numbers for the amount of Red, Green and Blue light to display. So this command gets the right LED to shine quite a lot of blue light and no red or green light.

Try typing the following:

>> pi2go.setAllLEDs(1000, 0, 0)



**Question 3:** What happens?

**Question 4:** Try six of the other commands from the table. What do they do?

**Command Result**

**Remember:** When you have finished working with your robot type:

**pi2go.cleanup()**



University of Liverpool, 2019

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).