**Programming in Python with the Pi2Go Simulator: Part 1 Answer Booklet**

**Chapter 1**

**Answers:** The answers to the question in chapter 1 are mostly descriptive.

* World items can be moved around by clicking and dragging.
* Lines have to be moved/deleted by clicking in the centre of the whole shape.
* Blocks can not be moved over the robot.
* Black squares and lines can move under the robot.
* The Light source can be placed and then its ray moved to point towards the robot.

**Potential Problems:** Moving between the Object Window and the Simulator Window is not entirely intuitive and may involve more clicks than users anticipate.

**Chapter 2**

**Sample Answers:** The questions here are all descriptive and should be easy to answer provided the student is watching the robot and doesn’t enter the commands too rapidly one after the other.

**Potential Problems:**

* Confusion may be caused by students failing to press Return after entering a command.

**Chapter 3**

**Sample Answers:** The questions here are all descriptive and should be easy to answer once the student has correctly understood what to type.

**Potential Problems:**

* Confusion may be caused by students failing to press Return after entering a command.
* Confusion may be caused by the use of argument for *speed* etc. in the commands. This may need to be explained to students.
* Watch out for the difference between lower case and upper case letter Ls in setAllLEDs
* Weaker students may not like the open nature of the final question and perhaps will need direction to try out a specific set of commands and see what they do.

**Chapter 4**

**Sample Answers:** The questions here are all descriptive again and the difficulties may lie in understanding how to change the values returned by the various sensors. Particularly with manipulating the objects from the Object Window.

**Chapter 5**

**Potential Problems:**

* Confusion about starting and stopping the simulator and where to type pi2go.cleanup() and pi2go.init() particularly if they have been using IDLE to start the simulator as well as to control the robot.

cleanup() and init() exist to cleaning stop and start the connection to the simulator from the exercise window in IDLE (the equivalent functions on the real robot establish connections to the sensors and motors). If cleanup() isn’t called then errors about sockets already being in used (Address already in use) will be raised. If init isn’t called then the robot will not respond to commands.

**Sample Answer Exercise 1:** For the first question the answers should be 90.0 (for the distance sensor) and 0 for both the IR sensors.

For the second question: When the robot hits a block it is stopped and judders slightly as it continues to hit the block. This often makes it move sideways and sometimes means it eventually gets around the block.

**Sample Answer Exercise 1:** The two commands both give the same answer. The exact value depends upon the position of the light source. They give the same answer because they both check the same sensor FL means (front left) and the 0 supplied as an input to getLight(0)tells the system the number of the sensor to use and the front left sensor is number 0.

There are three other paired light sensor commands: getLight(1) and getLightFR(), getLight(2) and getLightBR(), and getLight(3) and getLightBL().

**Sample Answer Exercise 3:** Both sensors should return 1.

**Potential Problems:**  There is a bug which means the simulator things the two line detection sensors for the Pi2Go are a short distance in front of the robot. This should be fixed in the next version of the software.

**Chapter 6**

**Troubleshooting:** The biggest cause of bugs in the exercises in WS5 are likely to be spelling errors (and possibly spacing errors if the students indent anything for some reason). If they do get errors, they should be encouraged to look at the line number indicated in the error message and check for spelling.

The IDLE IDE is good at highlighting syntax errors and the like. If the program won’t even run then students should be encouraged to look at the parts of the code highlighted in the IDE.

**Sample Answer 1:** The robot lights up its LEDs. Bright students may realise that although the program contains pi2go.forward(10) they don’t see this execute because it is interrupted by pi2go.stop() soon after.

**Sample Answer 2:**

import simclient.simrobot as pi2go

pi2go.init()

pi2go.forward(10)

pi2go.setAllLEDs(2000, 2000, 2000)

pi2go.stop();

pi2go.setAllLEDs(0, 0, 0)

**Potential Problems:**  Until WS6, there are no commands available to delay the execution of the next command in a Pi2Go program. There is obviously quite a lot of scope for confusion here that may need explaining.

**Chapter 7**

**Troubleshooting:** Students can still type commands into the IDLE window (e.g. pi2go.stop()) even after running a program from a file.

**Sample Answer 1:** The robot moves forward for 10 seconds and then stops.

**Sample Answer 2:** Changed the input for time.sleep(10) to time.sleep(20).

**Sample Answer 3:**

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.sleep(30)

pi2go.forward(10)

time.sleep(10)

pi2go.spinRight(10)

time.sleep(10)

pi2go.stop()

**Chapter 8**

**Sample Answer 1:**

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.spinRight(10)

time.sleep(2)

print(pi2go.getDistance())

pi2go.stop()

**Sample Answer 2:**

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.setAllLEDs(0, 0, 2000)

time.sleep(5)

pi2go.setAllLEDs(0, 2000, 0)

time.sleep(5)

pi2go.setAllLEDs(2000, 0, 0)

time.sleep(5)

**Sample Answer 3:** Note that the answers may vary depending upon the speed the robot turns

import simclient.simrobot as pi2go, time

pi2go.init()

print(pi2go.getLight(0))

print(pi2go.getLight(1))

print(pi2go.getLight(2))

print(pi2go.getLight(3))

pi2go.spinLeft(10)

time.sleep(9)

print(pi2go.getLight(0))

print(pi2go.getLight(1))

print(pi2go.getLight(2))

print(pi2go.getLight(3))

pi2go.spinLeft(10)

time.sleep(9)

print(pi2go.getLight(0))

print(pi2go.getLight(1))

print(pi2go.getLight(2))

print(pi2go.getLight(3))

pi2go.spinLeft(10)

time.sleep(9)

print(pi2go.getLight(0))

print(pi2go.getLight(1))

print(pi2go.getLight(2))

print(pi2go.getLight(3))

pi2go.spinLeft(10)

time.sleep(9)

pi2go.stop()

**Sample Answer 4:** Note speeds may vary!

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.forward(10)

time.sleep(10)

pi2go.forward(20)

time.sleep(10)

pi2go.forward(30)

time.sleep(10)

pi2go.stop()

**Sample Answer 5:** Note LED values may vary

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.setLED(0, 0, 1000, 2000)

pi2go.setLED(1, 1000, 2000, 0)

pi2go.setLED(2, 2000, 0, 1000)

pi2go.setLED(3, 2000, 2000, 2000)

pi2go.forward(10)

time.sleep(10)

pi2go.setLED(0, 1000, 2000, 0)

pi2go.setLED(1, 2000, 0, 1000)

pi2go.setLED(2, 2000, 2000, 2000)

pi2go.setLED(3, 0, 1000, 2000)

pi2go.spinLeft(10)

time.sleep(10)

pi2go.reverse(10)

pi2go.setLED(0, 2000, 0, 1000)

pi2go.setLED(1, 2000, 2000, 2000)

pi2go.setLED(2, 0, 1000, 2000)

pi2go.setLED(3, 1000, 2000, 0)

time.sleep(10)

pi2go.stop()

**Sample Answer 6:** Note answers will vary depending upon the speed and how much the students are prepared to have the robot hit walls as it travels round.

import simclient.simrobot as pi2go, time

pi2go.init()

pi2go.spinRight(10)

time.sleep(5)

pi2go.forward(10)

time.sleep(25)

pi2go.spinLeft(10)

time.sleep(9)

pi2go.forward(10)

time.sleep(15)

pi2go.spinRight(10)

time.sleep(8)

pi2go.forward(10)

time.sleep(25)

pi2go.spinLeft(10)

time.sleep(9)

pi2go.forward(10)

time.sleep(20)

pi2go.stop()

**Chapter 9**

**Sample Answer 1:** The robot moves forward for 10 seconds or prints “Obstacle!” and moves backwards for 10 seconds (answer depends upon whether the student had something in front of the robot or not - in the simulation probably not)

**Sample Answer 2:** To test the robot was working you need to test its behaviour both when there is an obstacle (it should move backward) and when there isn’t (it should move forward).

**Sample Answer 3:**

import simclient.simrobot as pi2go, time

pi2go.init()

if (pi2go.irLeft()):

 pi2go.spinRight(10)

elif (pi2go.irRight()):

 pi2go.spinLeft(10)

time.sleep(10)

pi2go.stop()

Three cases are needed to test this (something on the left, something on the right and nothing on either side).

The description of the exercise doesn’t say what to do if there is an obstacle on both sides of the robot.

**Sample Answer 4:**

import simclient.simrobot as pi2go, time

pi2go.init()

if (pi2go.irCentre()):

 pi2go.reverse(10)

 time.sleep(10)

 pi2go.spinRight(10)

 time.sleep(10)

 pi2go.stop()

else:

 pi2go.forward(10)

**Chapter 10**

**Sample Answer 1:** If there is nothing in front of the robot it does nothing. Otherwise it reverses and keeps printing out “reversing” until it is far enough away from the obstacle to stop.

**Sample Answer 2:** You would need to test it when there is an obstacle in front of it and when there isn’t an obstacle in front of it.

**Chapter 11**

**Sample Answer 1:** The program prints “Waiting” until the switch is pressed. Then it flashes its LEDs.

**Sample Answer 2:** Notice that three while loops are needed - wait for the switch to be switched on, wait for the switch to be switched off and then wait for the switch to be switched on again. Students may need some help thinking through this.

import simclient.simrobot as pi2go, time

pi2go.init()

while not (pi2go.getSwitch()):

 print("Waiting")

pi2go.setAllLEDs(4095, 4095, 4095)

while (pi2go.getSwitch()):

 print(“Waiting for Switch Off”)

while not (pi2go.getSwitch()):

 print(“Still Waiting”)

pi2go.setAllLEDs(0, 0, 0)

**Sample Answer 3:** pi2go.getSwitch() or pi2go.irCentre()

**Program:**

import simclient.simrobot as pi2go, time

pi2go.init()

if (pi2go.getSwitch() or pi2go.irCentre()):

 pi2go.reverse(10)

while (pi2go.getSwitch()):

 print(“Waiting for Switch Off”)

while not (pi2go.getSwitch()):

 print(“Still Waiting”)

pi2go.stop()

**Sample Answer 4:** pi2go.getSwitch() and not (pi2go.irCentre())

**Program:**

import simclient.simrobot as pi2go, time

pi2go.init()

if (pi2go.getSwitch() and not (pi2go.irCentre())):

 pi2go.forward(10)

while (pi2go.getSwitch()):

 print(“Waiting for Switch Off”)

while not (pi2go.getSwitch()):

 print(“Still Waiting”)

pi2go.stop()

**Chapter 12**

**Sample Answer 1:** While neither infra-red sensor detects an obstacle it moves forward and then sleeps for 10 seconds. If either sensor detects an obstacle then it stops.

**Sample Answer 2:** It does nothing until the left infra-red sensor detects something. Then it moves forward until the right infra-red sensor detects something. Then it stops.

**Sample Answer Exercise 1:**

import simclient.simrobot as pi2go

pi2go.init()

pi2go.forward(10)

while True:

 if (pi2go.irLeft()):

 break

pi2go.stop()

**Sample Answer Exercise 2:**

import simclient.simrobot as pi2go

import time

pi2go.init()

while True:

 if (pi2go.irLeft() and pi2go.irRight()):

 break

 if (pi2go.irLeft()):

 pi2go.spinRight(10)

 elif (pi2go.irRight()):

 pi2go.spinLeft(10)

 else:

 pi2go.forward(10)

 continue

 print("Spinning to find a clear route")

pi2go.stop()

**Chapter 13**

**Sample Answer Exercise 1:**

**Program:**

import simclient.simrobot as pi2go

pi2go.init()

if (pi2go.irCentre()):

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

if (pi2go.irLeft()):

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

if (pi2go.irRight()):

 pi2go.setLED(1, 1000, 1000, 1000)

**Sample Answer Exercise 2:**

import simclient.simrobot as pi2go

import time

pi2go.init()

while not (pi2go.getSwitch()):

 continue

time.sleep(2)

while not (pi2go.getSwitch()):

 print(pi2go.getDistance())

**Sample Answer Exercise 3:**

import simclient.simrobot as pi2go

pi2go.init()

while not (pi2go.getSwitch()):

 if (pi2go.irCentre()):

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

 else:

 pi2go.setLED(0, 0, 0, 0)

 if (pi2go.irLeft()):

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

 else:

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

 if (pi2go.irRight()):

 pi2go.setLED(1, 1000, 1000, 1000)

 else:

 pi2go.setLED(1, 0, 0, 0)

**Sample Answer Exercise 4:**

import simclient.simrobot as pi2go

pi2go.init()

if (pi2go.getSwitch()):

 while (pi2go.irCentre()):

 pi2go.reverse(10)

pi2go.stop()

**Sample Answer Exercise 5:**

import simclient.simrobot as pi2go

pi2go.init()

if (pi2go.getSwitch()):

 while not (pi2go.irCentre()):

 pi2go.forward(10)

pi2go.stop()

**Sample Answer Exercise 6:**

import simclient.simrobot as pi2go

pi2go.init()

while (pi2go.getSwitch()):

 if not (pi2go.irCentre()):

 pi2go.forward(10)

 else:

 pi2go.reverse(10)

pi2go.stop()

**Chapter 14**

**Sample Answer 1:**

Sensor: irCentre(), Motor: forward(10) and spinLeft(10) or spinRight(10)(speed values may vary)

**The Program:**

import simclient.simrobot as pi2go, time

pi2go.init()

time.sleep(30)

while True:

 while not (pi2go.irCentre()):

 pi2go.forward(10)

 while (pi2go.irCentre()):

 pi2go.spinLeft(30)

**Exercise 1:**

import simclient.simrobot as pi2go, time

pi2go.init()

while not (pi2go.getSwitch()):

 continue

while (pi2go.getSwitch()):

 continue

while not (pi2go.getSwitch()):

 while not (pi2go.irCentre()):

 pi2go.forward(10)

 while (pi2go.irCentre()):

 pi2go.spinLeft(30)

pi2go.stop()

**Exercise 2:**

import simclient.simrobot as pi2go, time

pi2go.init()

while not (pi2go.getSwitch()):

 continue

while (pi2go.getSwitch()):

 continue

while not (pi2go.getSwitch()):

 while (pi2go.irLeftLine()):

 pi2go.spinLeft(10)

 while (not pi2go.irLeftLine() and not pi2go.irRightLine()):

 pi2go.forward(30)

 while (pi2go.irRightLine()):

 pi2go.spinRight(30)

pi2go.stop()



 University of Liverpool, 2020

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