**Programming in Python with the Pi2Go Simulator: Part 2 Answer Booklet**

**Chapter 1**

**Sample Answer: 1** The program will take two readings from the ultrasonic (distance) sensor at a 10 second interval. If the first reading is less than the second reading it will print out “Object is moving away”. If the first reading is greater than the second reading it will print out “Object is moving closer”.

**Sample Answer 2:** There are three things to test. Firstly I can put a block in front of the robot, not move it, and run the program. It should print out “Object is not moving”. Secondly I can move the block after the program has started running. It should print out either “Object is moving away” or “Object is moving closer” depending upon whether I move the block closer or further away. I should test both these options.

**Potential Issues/Trouble Shooting:**

* Cut and paste of program from the work sheet may create syntax errors (particularly to do with the use of “ and indentation inside if statements)
* The simulator initialisation doesn’t complete until after initialisation complete is printed. At this point the first reading is immediately taken. Students who move the block to eagerly may find they get “Object is not moving” - they need to wait for the initialisation complete message before moving the block.
* Students have 10 seconds to move the block after initialisation is complete. This ought to be plenty of time but students do need to be aware of it.

**Sample Answer Exercise 1:** Note the use of time.sleep(10) and pi2go.stop()are not necessary to successfully complete the exercise, but they do make a nicer program.

import simclient.simrobot as pi2go, time

pi2go.init()

reading1 = pi2go.getDistance()

time.sleep(10)

reading2 = pi2go.getDistance()

if (reading1 < reading2):

pi2go.forward(10)

time.sleep(10)

pi2go.stop()

**Sample Answer Exercise 2:** The elif isn’t necessary but does showcase the use of !=

import simclient.simrobot as pi2go, time

pi2go.init()

reading1 = pi2go.getDistance()

time.sleep(10)

reading2 = pi2go.getDistance()

if (reading1 == reading2):

print("Object Stopped!")

elif (reading2 != reading1):

print("Object Moving!")

**Sample Answer Exercise 3:** Note I’ve reduced the sleep time in order to make the robot a bit more responsive.

Students may find it useful to use print statements to see the values of reading1 and reading2 in order to debug their programs.

import simclient.simrobot as pi2go, time

pi2go.init()

while (not pi2go.getSwitch()):

reading1 = pi2go.getDistance()

time.sleep(3)

reading2 = pi2go.getDistance()

if (reading1 < reading2):

pi2go.forward(10)

else:

pi2go.stop()

pi2go.stop()

**Chapter 2**

**Sample Answer 1:** When any key is pressed the program prints out the value from the ultrasonic distance sensor.

**Sample Answer 2:** Note that parts of the answer in italics will depend upon what happens when the program runs

Hello

*Name that was entered*

The Distance reading is

*Value from Ultrasonic Sensor*

**Sample Answer 3:** Y

**Sample Answer 4:** N

**Sample Answer Exercise 1:**

import simclient.simrobot as pi2go

pi2go.init()

num = input(['How many distance readings would you like?'])

start = 0

while (start < int(num)):

start = start + 1

print("Distance: " + str(pi2go.getDistance()))

**Sample Answer** **Exercise 2:**

import simclient.simrobot as pi2go

pi2go.init()

input = input(['Would you like a distance reading? [Y/N]'])

if (input == 'Y'):

print(pi2go.getDistance())

**Chapter 3**

What is printed out when you run the program? (It takes 1 minute to run)

***Probably 620***

**Exercise:** Modify the program so that it prints out the total distance measured over 10 measurements.

import simclient.simrobot as pi2go

import time

pi2go.init()

count = 0

total\_distance = 0

while (count < 10):

total\_distance = total\_distance + pi2go.getDistance()

time.sleep(3)

count = count + 1

print("The Total Distance is: " + str(total\_distance))

**Exercise:** Write a program that will take readings from the distance sensor until a total distance of over 1000 has been measured and then prints out the average distance per reading.

import simclient.simrobot as pi2go

import time

pi2go.init()

count = 0

total\_distance = 0

while (total\_distance < 1000):

total\_distance = total\_distance + pi2go.getDistance()

time.sleep(3)

count = count + 1

print("The Average Distance is: " + str(total\_distance/count))

**Working with Strings**

Try running the following program:

import simclient.simrobot as pi2go

name = input(['Please enter your name'])

print("Hello \"" + name + "\"")

What happens?

*It prompts the user to enter their name and then prints* ***Hello*** *followed by their name.*

**Exercise:** Write a program that uses new line and tab to ask someone their first name and then their surname then prints Hello followed by a tab then their first name and then prints their second name on a new line.

import simclient.simrobot as pi2go

fname = input(['Please enter your first name'])

sname = input(['Please enter your second name'])

print("Hello \t" + fname + "\n" + sname)

**Chapter 4**

**Exercise 1**:

import simclient.simrobot as pi2go

import time

pi2go.init()

d1 = pi2go.getDistance()

time.sleep(1)

d2 = pi2go.getDistance()

if (d1 != d2):

print("The readings were: " + str(d1) + " and " + str(d2))

**Exercise 2**

import simclient.simrobot as pi2go

import time

pi2go.init()

o1 = pi2go.irLeft()

o2 = pi2go.irRight()

if (o1 == o2):

pi2go.reverse(10)

elif (o1):

pi2go.spinRight(10)

else:

pi2go.spinLeft(10)

time.sleep(10)

pi2go.stop()

**Exercise 3**:

import simclient.simrobot as pi2go

import time

pi2go.init()

fb = input(["Please enter forward or backward"])

if (fb == 'forward' or fb == 'backward'):

seconds = input(["How long would you like the robot to move (enter a number under 10)?"])

if (int(seconds) < 10):

if (fb == 'forward'):

pi2go.forward(10)

else:

pi2go.reverse(10)

time.sleep(int(seconds))

pi2go.stop()

**Exercise 4**:

import simclient.simrobot as pi2go

import time

pi2go.init()

fb\_incorrect = True

while(fb\_incorrect):

fb = input(["Please enter forward or backward"])

if (fb == 'forward' or fb == 'backward'):

fb\_incorrect = False

s\_incorrect = True

while(s\_incorrect):

seconds = input(["How long would you like the robot to move (enter a number under 10)?"])

if (int(seconds) < 10):

s\_incorrect = False

if (fb == 'forward'):

pi2go.forward(10)

else:

pi2go.reverse(10)

time.sleep(int(seconds))

else:

print('You did not enter a number under 10')

else:

print('You did not enter forward or backward')

pi2go.stop()

**Exercise 5:**

import simclient.simrobot as pi2go

import time

pi2go.init()

direction = input(["Which way would you like the robot to move? (F, B, L, R)"])

if (direction == "F"):

pi2go.forward(10)

elif (direction == "B"):

pi2go.reverse(10)

elif (direction == "L"):

pi2go.spinLeft(10)

elif (direction == "R"):

pi2go.spinRight(10)

while(True):

print(pi2go.getDistance())

time.sleep(5)

pi2go.stop()

**Exercise 6:**

import simclient.simrobot as pi2go

pi2go.init()

direction = input(["Which way would you like the robot to move? (F, B, L, R)"])

while(direction != "S"):

if (direction == "F"):

pi2go.forward(10)

elif (direction == "B"):

pi2go.reverse(10)

elif (direction == "L"):

pi2go.spinLeft(10)

elif (direction == "R"):

pi2go.spinRight(10)

direction = input(["Which way would you like the robot to move next? (F, B, L, R, S)"])

pi2go.stop()

**Exercise 7:**

import simclient.simrobot as pi2go

import time

pi2go.init()

seconds = input(["Please enter a time in seconds"])

pi2go.forward(10)

count = 0

d = 0

while (count < int(seconds)):

time.sleep(1)

d = d+pi2go.getDistance()

count = count + 1

pi2go.stop()

print("Average Distance was " + str(d/count))

**Chapter 5**

**Question 1:** On what line has the error occurred? 12

**Question 2:** What does the error message say is missing from forward() ?

*1 required positional argument: ‘speed’*

**Corrected Program:**

pi2go.init()

direction = input(["Which way would you like the robot to move? (F, B, L, R)"])

while(direction != "S"):

if (direction == "F"):

pi2go.forward(10)

elif (direction == "B"):

pi2go.reverse(10)

elif (direction == "L"):

pi2go.spinLeft(10)

elif (direction == "R"):

pi2go.spinRight(10)

direction = input(["Which way would you like the robot to move next? (F, B, L, R, S)"])

pi2go.stop()

**Question 3:** Click on **Go**. What happens?

*The program executes as normal to the end – prompting for user input as it goes.*

**Question 4:** How many times do you have to click **Over**?

*This should be 27 but may vary depending on how accurately they input responses when prompted.*

**Question 5:** Run the program again and click **Over** a couple of times and then click **Go.** What happens?

*Once* ***Go*** *is clicked the program executes to the end as normal.*

**Question 6:** Now run the program and click **Go.** Enter R when prompted by the program. What happens?

*The program starts executing, prompts for input and then stops.*

**Potential Issue:** Sometimes, particularly if code has been cut and paste from a worksheet IDLE doesn’t display the lines of code exactly as they appear to the debugger so the program won’t stop at the breakpoints as expected – or may even not stop at any breakpoint. If this happens it is best to exit the file and then reopen.

**Question 7:** What line have you stopped at?

*This will be line 15 if they cut-and-pasted from the worksheet – otherwise it will depend a bit upon their line space.*

**Question 8:** What is the value of direction?

R

**Chapter 6**

**Exercise 1:** The two errors are failing to case average to a string and calculating the average using multiplication rather than division.

**Exercise 2:** The exercise loops infinitely because total\_distance never increases. total\_distance should be calculated during the loop using the ulta-sonic sensor (like in Exercise 1).

**Chapter 7**

**Exercise 1:**

import simclient.simrobot as pi2go

import time

def flash\_LEDs():

pi2go.setAllLEDs(500, 500, 500)

time.sleep(1)

pi2go.setAllLEDs(0, 0, 0)

time.sleep(1)

pi2go.setAllLEDs(500, 500, 500)

time.sleep(1)

pi2go.setAllLEDs(0, 0, 0)

pi2go.init()

while (not pi2go.irCentre()):

pi2go.forward(10)

pi2go.stop()

flash\_LEDs()

while (pi2go.irCentre()):

pi2go.spinLeft(10)

time.sleep(5)

pi2go.stop()

flash\_LEDs()

while (not pi2go.irCentre()):

pi2go.forward(10)

pi2go.stop()

flash\_LEDs()

A cleverer answer to Exercise 1 uses a second function as follows:

import simclient.simrobot as pi2go

import time

def flash\_LEDs():

pi2go.setAllLEDs(500, 500, 500)

time.sleep(1)

pi2go.setAllLEDs(0, 0, 0)

time.sleep(1)

pi2go.setAllLEDs(500, 500, 500)

time.sleep(1)

pi2go.setAllLEDs(0, 0, 0)

def while\_no\_obstacle():

while (not pi2go.irCentre()):

pi2go.forward(10)

pi2go.stop();

pi2go.init()

while\_no\_obstacle()

flash\_LEDs()

while (pi2go.irCentre()):

pi2go.spinLeft(10)

time.sleep(5)

pi2go.stop()

flash\_LEDs()

while\_no\_obstacle()

flash\_LEDs()

**Question 1&2:** The turn(side) function turns left or right depending upon its argument. The following program turns the robot right for 10 seconds.

**Exercise 2:**

def turn\_obstacle(side):

if (side == 'right'):

pi2go.spinLeft(10)

while(pi2go.irRight()):

continue

else:

pi2go.spinRight(10)

while(pi2go.irLeft()):

continue

pi2go.stop()

pi2go.init()

turn\_obstacle('left')

**Question 3:** returns the value of the obstacle sensor on the left, right or centre depending upon its argument.

**Exercise 3:**

import simclient.simrobot as pi2go

import time

def obstacle(side):

if (side == 'left'):

return pi2go.irLeft()

elif (side == 'right'):

return pi2go.irRight()

else:

return pi2go.irCentre()

pi2go.init()

print(obstacle('left'))

print(obstacle('centre'))

print(obstacle('left'))

**Exercise 4:**

def opposite(side):

if (side == 'left'):

return 'right'

else:

return 'left'

**Exercise 5:**

import simclient.simrobot as pi2go

import time

def obstacle(side):

if (side == 'left'):

return pi2go.irLeft()

elif (side == 'right'):

return pi2go.irRight()

else:

return pi2go.irCentre()

def turn(side):

if (side == 'left'):

pi2go.spinLeft(10)

else:

pi2go.spinRight(10)

def turn\_until(side):

turn(side)

while(obstacle(opposite(side))):

continue

pi2go.stop();

def opposite(side):

if (side == 'left'):

return 'right'

else:

return 'left'

pi2go.init()

turn\_until('left')

**Chapter 8**

**Exercise 1:**

import simclient.simrobot as pi2go

import time

def when\_switch\_pressed():

while (not pi2go.getSwitch() == 1):

continue

time.sleep(3)

pi2go.init()

when\_switch\_pressed()

pi2go.forward(10)

when\_switch\_pressed()

pi2go.stop()

**Exercise 2:**

import simclient.simrobot as pi2go

import time

def forward\_for(t):

pi2go.forward(10)

time.sleep(t)

pi2go.stop()

pi2go.init()

t1 = input("Enter a time in seconds")

forward\_for(int(t1))

**Exercise 3:**

import simclient.simrobot as pi2go

import time

def average\_distance():

count = 0

total = 0

while (count < 10):

total = total + pi2go.getDistance()

time.sleep(1)

count = count + 1

return total/10

pi2go.init()

pi2go.forward(10)

average = average\_distance()

pi2go.stop()

print(str(average))

**Exercise 4:**

def follow\_line():

while True:

while (pi2go.irLeftLine()):

pi2go.spinLeft(10)

while (pi2go.irRightLine()):

pi2go.spinRight(10)

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

pi2go.forward(10)

**Exercise 5:**

import simclient.simrobot as pi2go

import time

def avoid\_obstacle():

while (not pi2go.getSwitch()):

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

pi2go.spinLeft(10)

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

pi2go.forward(10)

def follow\_line():

while (not pi2go.getSwitch()):

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

pi2go.spinLeft(10)

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

pi2go.spinRight(10)

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

pi2go.forward(10)

pi2go.init()

while True:

todo = input("Obstacle or Line or Stop? O/L/S")

if (todo == "O"):

avoid\_obstacle()

elif (todo == "L"):

follow\_line()

else:

break

pi2go.stop()

**Exercise 6:**

import simclient.simrobot as pi2go

import time

def brightest():

if (pi2go.getLightFL() > pi2go.getLightFR()):

return 'left'

elif (pi2go.getLightFL() < pi2go.getLightFR()):

return 'right'

else:

return 'neither'

def turn(side):

if (side == 'left'):

pi2go.spinLeft(10)

elif (side == 'right'):

pi2go.spinRight(10)

else:

pi2go.stop()

pi2go.init()

while (not brightest() == 'neither'):

turn(brightest())

pi2go.stop()

**Chapter 9**

**Question 1:** The robot should be initialised (first command), turn left (second command) and then stop (third command).

**Troubleshooting Note:** If the students don’t type cleanup between reimports of the module then there is a good chance they will get error messages talking about socket connections. If this happens it may be necessary to close down both IDLE and the simulator and start again.

**Question 2:**

>>> import turning as my\_turning

>>> my\_turning.pi2go.init()

>>> my\_turning.turn('right')

>>> my\_turning.pi2go.stop()

**Question 3:**

The module should behave just as it did previously – in particular it *won’t* print out the new message in the turn function. This is because the module hasn’t actually been reloaded.

**Question 4:** When **importlib** is used the module correctly reloads and this time the message is printed out when the turn function is executed.

**Exercise:**

import simclient.simrobot as pi2go

import time

def turn(side):

print("message")

if (side == 'left'):

pi2go.spinLeft(10)

else:

pi2go.spinRight(10)

def obstacle(side):

if (side == 'left'):

return pi2go.irLeft()

elif (side == 'right'):

return pi2go.irRight()

else:

return pi2go.irCentre()

pi2go.init()

**Chapter 10**

**Challenge Problem Sample Answer:**

**import** simclient.simrobot **as** pi2go

**import** time

pi2go.init()

**def** drive\_to\_wall():

**while** (**not** pi2go.irCentre() **and** **not** pi2go.irLeft() **and** **not** pi2go.irRight()):

pi2go.forward(10)

pi2go.stop()

**def** spin(direction):

**if** (direction == 'right'):

pi2go.spinRight(10)

**else**:

pi2go.spinLeft(10)

**def** obstacle\_to(direction):

**if** (direction == 'right'):

**return** pi2go.irRight()

**else**:

**return** pi2go.irLeft()

**def** opposite\_direction(direction):

**if** (direction == 'right'):

**return** 'left'

**else**:

**return** 'right'

**def** follow\_wall(direction):

**if** (pi2go.irCentre()):

**while**(pi2go.irCentre() **or** obstacle\_to(direction)):

spin(opposite\_direction(direction))

**elif** (obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

**elif** (**not** obstacle\_to(direction)):

spin(direction)

time.sleep(1)

drive\_to\_wall()

**while** (**not** pi2go.irLeftLine()):

follow\_wall('right')

pi2go.forward(10)

time.sleep(3)

pi2go.reverse(10)

time.sleep(10)

pi2go.spinLeft(10)

time.sleep(5)

**while** (**not** pi2go.irLeftLine()):

follow\_wall('left')

pi2go.stop()

**Troubleshooting:** It may take some experimentation to get sleep times correct. This above version of wall following often ends up “bouncing along” the side of the wall since the robot becomes angled slightly towards it. Some fine tuning of timings might help with this, as might using the ultra-sonic sensor as well as the irCentre sensor – but, to be honest, these steps make the program a lot more fiddly and don’t necessarily improve performance. Really the robot needs more sensors at the side in order to help it determine the angle it is at with respect to the wall. So the above is probably as good as can reasonably be expected.

**Exercise Sample Answer**: This has been tweaked a bit from the above in an attempt to reduce the “bouncing” effect. There’s no real right answer here since perfect performance isn’t possible without using quite sophisticated mathematical and statistical techniques in order to map the space.

import simclient.simrobot as pi2go

import time

def drive\_to\_wall():

while (not pi2go.irCentre() and not pi2go.irLeft() and not pi2go.irRight()):

pi2go.forward(10)

pi2go.stop()

def spin(direction):

if (direction == 'right'):

pi2go.spinRight(10)

else:

pi2go.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return pi2go.irRight()

else:

return pi2go.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

while (not pi2go.getSwitch()):

if (pi2go.irCentre()):

while(pi2go.irCentre() or obstacle\_to(direction)):

spin(opposite\_direction(direction))

pi2go.forward(10)

time.sleep(2)

elif (obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

spin(opposite\_direction(direction))

time.sleep(2)

elif (not obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

spin(direction)

time.sleep(2)

pi2go.stop()



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/).