**Programming in Python with the Initio 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 initio.stop()are not necessary to successfully complete the exercise, but they do make a nicer program.

import simclient.simrobot as initio, time

initio.init()

reading1 = initio.getDistance()

time.sleep(10)

reading2 = initio.getDistance()

if (reading1 < reading2):

initio.forward(10)

time.sleep(10)

initio.stop()

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

import simclient.simrobot as initio, time

initio.init()

reading1 = initio.getDistance()

time.sleep(10)

reading2 = initio.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 initio, time

initio.init()

while (not initio.irLeft()):

reading1 = initio.getDistance()

time.sleep(3)

reading2 = initio.getDistance()

if (reading1 < reading2):

initio.forward(10)

else:

initio.stop()

initio.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 initio

initio.init()

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

start = 0

while (start < int(num)):

start = start + 1

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

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

import simclient.simrobot as initio

initio.init()

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

if (input == 'Y'):

print(initio.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 initio

import time

initio.init()

count = 0

total\_distance = 0

while (count < 10):

total\_distance = total\_distance + initio.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 initio

import time

initio.init()

count = 0

total\_distance = 0

while (total\_distance < 1000):

total\_distance = total\_distance + initio.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 initio

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 initio

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 initio

import time

initio.init()

d1 = initio.getDistance()

time.sleep(1)

d2 = initio.getDistance()

if (d1 != d2):

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

**Exercise 2**

import simclient.simrobot as initio

import time

initio.init()

o1 = initio.irLeft()

o2 = initio.irRight()

if (o1 == o2):

initio.reverse(10)

elif (o1):

initio.spinRight(10)

else:

initio.spinLeft(10)

time.sleep(10)

initio.stop()

**Exercise 3**:

import simclient.simrobot as initio

import time

initio.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'):

initio.forward(10)

else:

initio.reverse(10)

time.sleep(int(seconds))

initio.stop()

**Exercise 4**:

import simclient.simrobot as initio

import time

initio.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'):

initio.forward(10)

else:

initio.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')

initio.stop()

**Exercise 5:**

import simclient.simrobot as initio

import time

initio.init()

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

if (direction == "F"):

initio.forward(10)

elif (direction == "B"):

initio.reverse(10)

elif (direction == "L"):

initio.spinLeft(10)

elif (direction == "R"):

initio.spinRight(10)

while(True):

print(initio.getDistance())

time.sleep(5)

initio.stop()

**Exercise 6:**

import simclient.simrobot as initio

initio.init()

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

while(direction != "S"):

if (direction == "F"):

initio.forward(10)

elif (direction == "B"):

initio.reverse(10)

elif (direction == "L"):

initio.spinLeft(10)

elif (direction == "R"):

initio.spinRight(10)

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

initio.stop()

**Exercise 7:**

import simclient.simrobot as initio

import time

initio.init()

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

initio.forward(10)

count = 0

d = 0

while (count < int(seconds)):

time.sleep(1)

d = d+initio.getDistance()

count = count + 1

initio.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:**

initio.init()

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

while(direction != "S"):

if (direction == "F"):

initio.forward(10)

elif (direction == "B"):

initio.reverse(10)

elif (direction == "L"):

initio.spinLeft(10)

elif (direction == "R"):

initio.spinRight(10)

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

initio.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:**  Note that the students won’t see much movement here since it doesn’t turn very far before it loses the obstacle. They might want to alter the distances so they seem more activity.

import simclient.simrobot as initio

import time

def shake\_head():

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

initio.init()

while (not initio.getDistance() < 50):

initio.forward(10)

initio.stop()

shake\_head()

while (initio.getDistance() < 50):

initio.spinLeft(10)

time.sleep(5)

initio.stop()

shake\_head()

while (not initio.getDistance() < 50):

initio.forward(10)

initio.stop()

shake\_head()

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

import simclient.simrobot as initio

import time

def shake\_head():

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

def while\_no\_obstacle():

while (not initio.getDistance() < 50):

initio.forward(10)

initio.stop();

initio.init()

while\_no\_obstacle()

shake\_head()

while (initio.getDistance() < 50):

initio.spinLeft(10)

time.sleep(5)

initio.stop()

shake\_head()

while\_no\_obstacle()

shake\_head()

**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'):

initio.spinLeft(10)

while(initio.irRight()):

continue

else:

initio.spinRight(10)

while(initio.irLeft()):

continue

initio.stop()

initio.init()

turn\_obstacle('left')

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

**Exercise 3:**

import simclient.simrobot as initio

import time

def obstacle(side):

if (side == 'left'):

return initio.irLeft()

else:

return initio.irRight()

initio.init()

print(obstacle('left'))

print(obstacle('left'))

**Exercise 4:**

def opposite(side):

if (side == 'left'):

return 'right'

else:

return 'left'

**Exercise 5:**

import simclient.simrobot as initio

import time

def obstacle(side):

if (side == 'left'):

return initio.irLeft()

else:

return initio.irRight()

def turn(side):

if (side == 'left'):

initio.spinLeft(10)

else:

initio.spinRight(10)

def turn\_until(side):

turn(side)

while(obstacle(opposite(side))):

continue

initio.stop();

def opposite(side):

if (side == 'left'):

return 'right'

else:

return 'left'

initio.init()

turn\_until('left')

**Chapter 8**

**Exercise 1:**

import simclient.simrobot as initio

import time

def when\_obstacle\_close():

while (not initio.getDistance() < 15):

continue

time.sleep(3)

initio.init()

when\_obstacle\_close()

initio.forward(10)

when\_obstacle\_close()

initio.stop()

**Exercise 2:**

import simclient.simrobot as initio

import time

def forward\_for(t):

initio.forward(10)

time.sleep(t)

initio.stop()

initio.init()

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

forward\_for(int(t1))

**Exercise 3:**

import simclient.simrobot as initio

import time

def average\_distance():

count = 0

total = 0

while (count < 10):

total = total + initio.getDistance()

time.sleep(1)

count = count + 1

return total/10

initio.init()

initio.forward(10)

average = average\_distance()

initio.stop()

print(str(average))

**Exercise 4:**

def follow\_line():

while True:

while (initio.irLeftLine()):

initio.spinLeft(10)

while (initio.irRightLine()):

initio.spinRight(10)

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

initio.forward(10)

**Exercise 5:**

import simclient.simrobot as initio

import time

def avoid\_obstacle():

while (not initio.getDistance() < 15):

while (initio.getDistance () < 50 and initio.getDistance() > 15):

initio.spinLeft(10)

while (initio.getDistance() > 50):

initio.forward(10)

def follow\_line():

while (not initio.getDistance() < 15):

while (initio.irLeftLine() and not initio.getDistance() < 15):

initio.spinLeft(10)

while (initio.irRightLine() and not initio.getDistance() < 15):

initio.spinRight(10)

while (not initio.irLeftLine() and not initio.irRightLine() and not initio.getDistance() < 15):

initio.forward(10)

initio.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

initio.stop()

**Exercise 6:** Note the use of a sleep command to give the servo time to turn. In the simulator this program may never end.

import simclient.simrobot as initio

import time

def furthest():

initio.setServo(1, 20)

time.sleep(1)

left = initio.getDistance()

initio.setServo(1, -20)

time.sleep(1)

right = initio.getDistance()

initio.setServo(1, 0)

print(str(left))

print(str(right))

if (left > right):

return 'left'

elif (left < right):

return 'right'

else:

return 'neither'

def turn(side):

if (side == 'left'):

initio.spinLeft(10)

elif (side == 'right'):

initio.spinRight(10)

else:

initio.stop()

initio.init()

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

turn(furthest())

initio.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.initio.init()

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

>>> my\_turning.initio.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 initio

import time

def turn(side):

print("message")

if (side == 'left'):

initio.spinLeft(10)

else:

initio.spinRight(10)

def obstacle(side):

if (side == 'left'):

return initio.irLeft()

else:

return initio.irRight()

initio.init()

**Chapter 10**

**Challenge Problem Sample Answer:**

import simclient.simrobot as initio

import time

initio.init()

def drive\_to\_wall():

while (not initio.getDistance() < 20 and not initio.irLeft() and not initio.irRight()):

initio.forward(10)

initio.stop()

def spin(direction):

if (direction == 'right'):

initio.spinRight(10)

else:

initio.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return initio.irRight()

else:

return initio.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

if (initio.getDistance() < 20):

while(initio.getDistance() < 20 or obstacle\_to(direction)):

spin(opposite\_direction(direction))

elif (obstacle\_to(direction)):

initio.forward(10)

time.sleep(2)

elif (not obstacle\_to(direction)):

initio.forward(10)

time.sleep(1)

spin(direction)

time.sleep(1)

drive\_to\_wall()

while (not initio.irLeftLine()):

follow\_wall('right')

initio.forward(10)

time.sleep(3)

initio.reverse(10)

time.sleep(10)

initio.spinLeft(10)

time.sleep(25)

while (not initio.irLeftLine()):

follow\_wall('left')

initio.stop()

**Troubleshooting:** It may take some experimentation to get sleep times correct. This above version of wall following can end 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. 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 2**. Pretty much the same as above just without program commands at the end.

import simclient.simrobot as initio

import time

def drive\_to\_wall():

while (not initio.getDistance() < 20 and not initio.irLeft() and not initio.irRight()):

initio.forward(10)

initio.stop()

def spin(direction):

if (direction == 'right'):

initio.spinRight(10)

else:

initio.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return initio.irRight()

else:

return initio.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

if (initio.getDistance() < 20):

while(initio.getDistance() < 20 or obstacle\_to(direction)):

spin(opposite\_direction(direction))

elif (obstacle\_to(direction)):

initio.forward(10)

time.sleep(2)

elif (not obstacle\_to(direction)):

initio.forward(10)

time.sleep(1)

spin(direction)

time.sleep(1)



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/).