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

**Chapter 1**

**Answers:** The answers to the questions 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.

**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.
* 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 initio.cleanup() and initio.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.

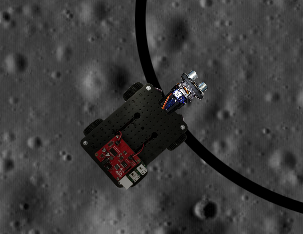
**Sample Answer Exercise 1:** For the first question the answers should be 90.0 (for the distance sensor) and 0 for both the IR sensors.

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 2:** When the sensor is facing directly to the wall, setServo(1, 0), the distance is 30 as the servo rotates this gradually increases as the distance between the wall and the sensor increases (to 40, 50, 60) until it suddenly increases a lot in the negative direction where the sensor is pointing at the gap before reducing as the sensor detects the wall on the right of the robot. On the left the values increase until the sensor is pointing to the corner and then decrease as it points to the edge of the world. (NB. The exact numbers may vary depending on where the children placed the robot, but the important thing is that they understand how the sensor values are changing depending upon where it is pointing).

**Sample Answer Exercise 3:** Both sensors should return 0.

**Potential Problems:** It can be very fiddly to get the robot correctly positioned over the black line, particularly if the user isn’t entirely sure where the sensor are. The picture below shows one correct placement.



**Chapter 6**

**Troubleshooting:** The biggest cause of bugs in the exercises in chapter 6 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 pans its ultrasonic sensor 45 degrees. Bright students may realise that although the program contains initio.forward(10) they don’t see this execute because it is interrupted by initio.stop() soon after.

**Sample Answer 2:**

import simclient.simrobot as initio

initio.init()

initio.forward(10)

initio.setServo(0, -45)

initio.stop()

initio.setServo(0, 0)

**Potential Problems:** If the ultrasonic servo starts at 0 in the new program (or -45 in the first program) then the program will appear to do nothing at all. Until WS6, there are no commands available to delay the execution of the next command in an Initio 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. initio.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 initio, time

initio.init()

time.sleep(30)

initio.forward(10)

time.sleep(10)

initio.spinRight(10)

time.sleep(10)

initio.stop()

**Chapter 8**

**Sample Answer 1:**

import simclient.simrobot as initio, time

initio.init()

initio.spinRight(10)

time.sleep(2)

print(initio.getDistance())

initio.stop()

**Sample Answer 2:**

import simclient.simrobot as initio, time

initio.init()

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

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

This answer pauses each time readings are taken to help the programmer tell if a quarter circle has been turned.

import simclient.simrobot as initio, time

initio.init()

print(initio.irLeft())

print(initio.irRight())

initio.spinLeft(60)

time.sleep(1.5)

initio.stop()

time.sleep(1)

print(initio.irLeft())

print(initio.irRight())

initio.spinLeft(60)

time.sleep(1.5)

initio.stop()

time.sleep(1)

print(initio.irLeft())

print(initio.irRight())

initio.spinLeft(60)

time.sleep(1.5)

initio.stop()

time.sleep(1)

print(initio.irLeft())

print(initio.irRight())

initio.spinLeft(60)

time.sleep(1.5)

initio.stop()

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

import simclient.simrobot as initio, time

initio.init()

initio.forward(10)

time.sleep(10)

initio.forward(20)

time.sleep(10)

initio.forward(30)

time.sleep(10)

initio.stop()

**Sample Answer 5:**

import simclient.simrobot as initio, time

inito.init()

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

inito.forward(10)

time.sleep(10)

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

inito.spinLeft(10)

time.sleep(10)

initio.setServo(1, 20)

time.sleep(5)

initio.setServo(1, -20)

time.sleep(5)

initio.setServo(1, 0)

initio.reverse(10)

time.sleep(10)

initio.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 initio, time

initio.init()

initio.spinRight(10)

time.sleep(5)

initio.forward(10)

time.sleep(25)

initio.spinLeft(10)

time.sleep(9)

initio.forward(10)

time.sleep(15)

initio.spinRight(10)

time.sleep(8)

initio.forward(10)

time.sleep(25)

initio.spinLeft(10)

time.sleep(9)

initio.forward(10)

time.sleep(20)

initio.stop()

**Chapter 9**

**Sample Answer 1:** The robot moves forward for 10 seconds or print the value from the distance sensor and move 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 initio, time

initio.init()

if (initio.irLeft()):

initio.spinRight(10)

elif (initio.irRight()):

initio.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 initio, time

initio.init()

if (initio.getDistance() < 30):

initio.reverse(10)

time.sleep(10)

initio.spinRight(10)

time.sleep(10)

initio.stop()

else:

initio.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 something comes closer than 5cm. Then it “waves its head”.

**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 initio, time

initio.init()

while (initio.getDistance() > 50):

print(“Waiting”)

initio.setServo(1, 20)

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

print(“Waiting for Obstacle to Move”)

while (initio.getDistance() > 50):

print(“Still Waiting”)

initio.setServo(1, 0)

**Sample Answer 3:** not (initio.getDistance() > 5)

**Program:**

import simclient.simrobot as initio, time

initio.init()

if not (initio.getDistance() > 50):

initio.forward(10)

while (initio.getDistance() > 50):

print(“Waiting for Obstacle”)

initio.stop()

**Sample Answer 4:** not (initio.irLeft() or initio.irRight())

or alternatively (not (initio.irLeft()) and not (initio.irRight())

**Program:**

import simclient.simrobot as initio, time

initio.init()

if (not (initio.irLeft() or initio.irRight())):

initio.forward(10)

while (not (initio.irLeft() or initio.irRight())):

print(“Still Advancing”)

initio.stop()

**Chapter 12**

**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 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: getDistance(), Motor: forward(10) and spinLeft(10) or spinRight(10)(speed values may vary)

**The Program:**

import simclient.simrobot as initio, time

initio.init()

time.sleep(30)

while True:

while (initio.getDistance() > 50):

initio.forward(10)

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

initio.spinLeft(30)

**Program Modified to Stop Cleanly**

import simclient.simrobot as initio, time

initio.init()

while (initio.getDistance() > 5):

continue

while not (initio.getDistance() > 5):

continue

while (initio.getDistance() > 5):

while (initio.getDistance() > 50):

initio.forward(10)

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

initio.spinLeft(30)

initio.stop()

**Exercise 2:**

import simclient.simrobot as initio, time

initio.init()

while (initio.getDistance() > 5):

continue

while not (initio.getDistance() > 5):

continue

while (initio.getDistance() > 5):

while (initio.irLeftLine()):

initio.spinLeft(10)

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

initio.forward(30)

while (initio.irRightLine()):

initio.spinRight(30)

initio.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/).



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/).