**Virtual Pi2Go Programming: Functions**

**AIM:** After completing this worksheet you should be able to describe what a function and a function argument is in programming and use functions (with and without arguments) and functions which return values in your Python Programs.

**You Need:** To complete this worksheet you need to have a virtual Pi2Go simulator (see WS1), and to be able to use files to store Programs (WS5). You also need to know the commands to operate the Pi2Go motors and sensors (WS3 & WS4). You should be able to use If statements (WS7), while loops (WS8 & WS10), variables, numbers (WS12) and strings (WS13) in Python programs.

**If the simulator isn’t already running: Start the Simulator, Select the Pi2Go Simulation and house.xml, then start IDLE (open a *new IDLE window* if you have used IDLE to start the simulator).**

Sometimes we have parts of a program that perform the same sequence of commands.

Consider the following program that drives the robot forward for 5 seconds and flashes its LEDs and then drives it backwards for five seconds and flashes its LEDs.

import simclient.simrobot as pi2go

import time

pi2go.init()

pi2go.forward(10)

time.sleep(5)

pi2go.stop()

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.reverse(10)

time.sleep(5)

pi2go.stop()

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)

It would be good to separate out the code for flashing the LEDs, so we didn’t have to type it all twice. We can do this using a *function*. Functions in programs are like mini-programs that can be executed to perform some small task. You can *call* functions from within larger programs. All the commands you have been using with the virtual Pi2Go such as **forward, getDistance, init**  and so on are functions.

To create a function, you use the keyword def followed by the name of the function and then open and close brackets and a colon. You then write the code for the function indented (like with **if**  and **while**) on the lines below. So, we can write a “flash LED” functions as follows:

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)

The program becomes:

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()

pi2go.forward(10)

time.sleep(5)

pi2go.stop()

flash\_LEDs()

pi2go.reverse(10)

time.sleep(5)

pi2go.stop()

flash\_LEDs()

**Exercise 1:** Write a program that will move forward until it detects an obstacle. At that point it stops and flashes its LEDs. Then it turns until it no longer detects an obstacle (at which point it flashes its LEDs again). Then it moves forward until it detects another obstacle, stops and flashes its LEDs. Use the flash\_LEDs() function above for all the flashing.

**Functions with Arguments**

We don’t have to just use functions when we want to run the *exact same* piece of code again. We can use them if we want to run similar, but slightly different pieces of code. To do this we provide the functions with *arguments.* These appear in between the brackets after the function name. The arguments are names of variables that can then be used inside the function.

Consider the following function:

def turn(side):

 if (side == 'left'):

 pi2go.spinLeft(10)

 else:

 pi2go.spinRight(10)

**Question 1:** What does it do?

**Question 2:** What does the following program do?

import simclient.simrobot as pi2go

import time

def turn(side):

 if (side == 'left'):

 pi2go.spinLeft(10)

 else:

 pi2go.spinRight(10)

pi2go.init()

turn('right')

time.sleep(10)

pi2go.stop()

**Exercise 2:** Write a function that takes either **‘left’** or **‘right’** as an argument and, if it detects an obstacle on that side, turns away from it until it no longer detects an obstacle. Show the use of this function in a program.

**Functions that Calculate Values**

Suppose we want a function that will calculate a value for a variable? We can use the return keyword to return the value

 Consider the following function:

def obstacle(side):

 if (side == 'left'):

 return pi2go.irLeft()

 elif (side == 'right'):

 return pi2go.irRight()

 else:

 return pi2go.irCentre()

**Question 3:** What does it do?

**Exercise 3:** Write a program that uses the obstacle function to print out the value from each of the three infra-red distance sensors

**Exercise 4:** Write a function, opposite(side), that returns ‘left’ if its argument is ‘right’ and vice versa.

**Exercise 5:** Using your opposite(side) function, the obstacle(side) function and the turn(side) function write a function turn\_until(side) that turns away from an obstacle until it is no longer picked up by the infra-red sensor on that side. Illustrate its use in a program.



 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/).