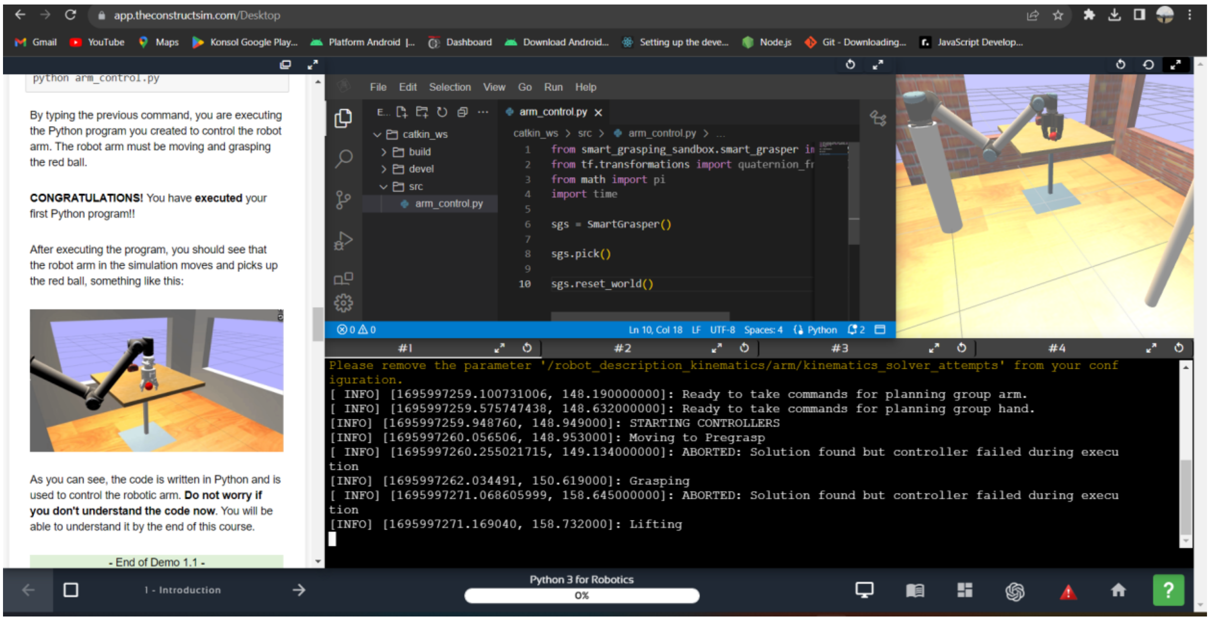
FILE SCREENSHOT PYTHON FOR ROBOTICS

FARIZ RAHMAN RAMADHAN

1103204046

- INTRODUCTION

DEMO 1.1



```
python arm_control.py
```

By typing the previous command, you are executing the Python program you created to control the robot arm. The robot arm must be moving and grasping the red ball.

CONGRATULATIONS! You have **executed** your first Python program!!

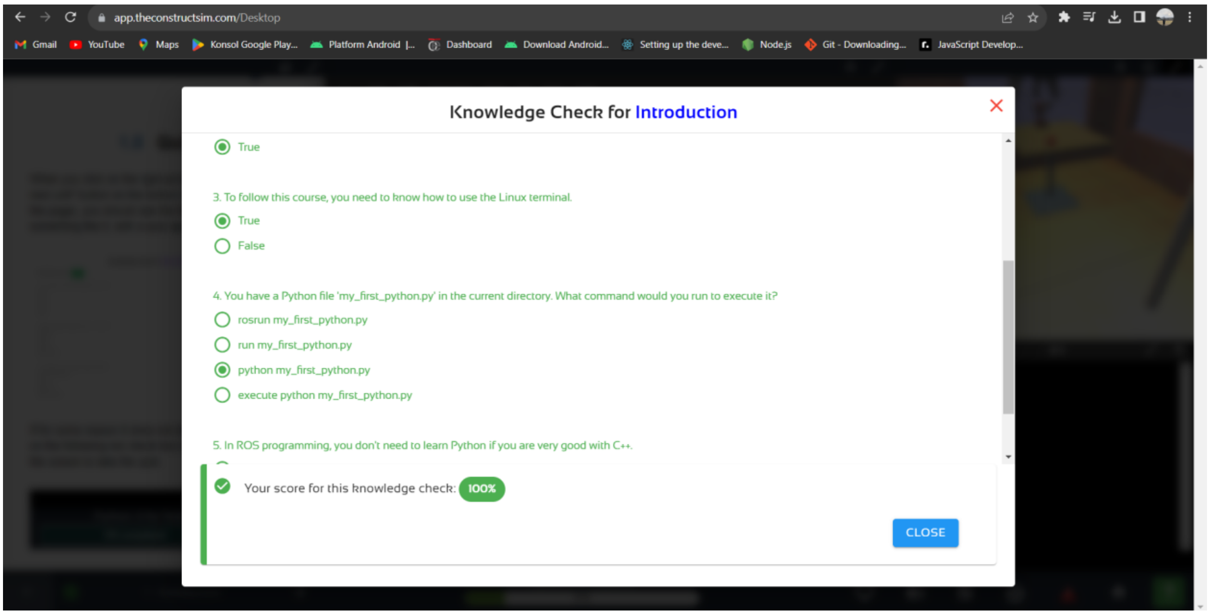
After executing the program, you should see that the robot arm in the simulation moves and picks up the red ball, something like this:

As you can see, the code is written in Python and is used to control the robotic arm. **Do not worry if you don't understand the code now**. You will be able to understand it by the end of this course.

- End of Demo 1.1 -

```
from smart_grasping_sandbox.smart_grasper in
from tf.transformations import quaternion_fr
from math import pi
import time

sgs = SmartGrasper()

sgs.pick()

sgs.reset_world()
```

```
Please remove the parameter '/robot_description_kinematics/arm/kinematics_solver_attempts' from your conf
iguration.
[ INFO] [1695997259.100731006, 148.190000000]: Ready to take commands for planning group arm.
[ INFO] [1695997259.575747438, 148.632000000]: Ready to take commands for planning group hand.
[INFO] [1695997259.948760, 148.949000]: STARTING CONTROLLERS
[INFO] [1695997260.056506, 148.953000]: Moving to Pregrasp
[ INFO] [1695997260.255021715, 149.134000000]: ABORTED: Solution found but controller failed during execu
tion
[INFO] [1695997262.034491, 150.619000]: Grasping
[ INFO] [1695997271.068605999, 158.645000000]: ABORTED: Solution found but controller failed during execu
tion
[INFO] [1695997271.169040, 158.732000]: Lifting
```



Knowledge Check for Introduction

● True

3. To follow this course, you need to know how to use the Linux terminal.

● True
○ False

4. You have a Python file 'my_first_python.py' in the current directory. What command would you run to execute it?

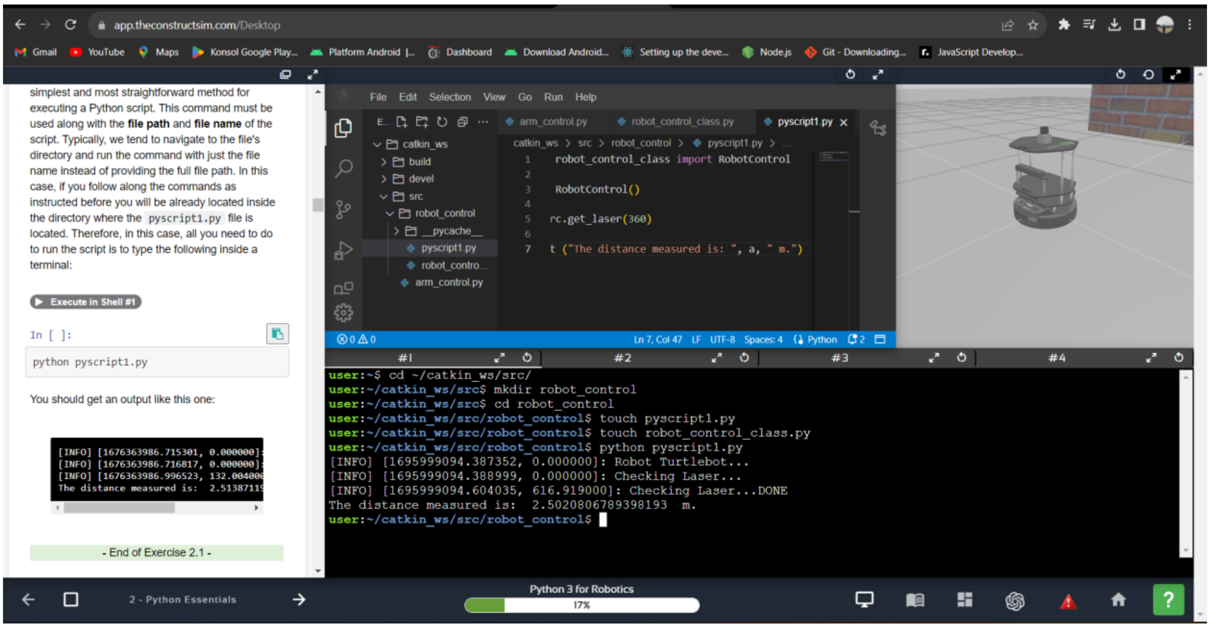
○ rosrun my_first_python.py
○ run my_first_python.py
● python my_first_python.py
○ execute python my_first_python.py

5. In ROS programming, you don't need to learn Python if you are very good with C++.

Your score for this knowledge check: 100%

- PYTHON ESSENTIALS

EXERCISE 2.1



simplest and most straightforward method for executing a Python script. This command must be used along with the **file path** and **file name** of the script. Typically, we tend to navigate to the file's directory and run the command with just the file name instead of providing the full file path. In this case, if you follow along the commands as instructed before you will be already located inside the directory where the `pyscript1.py` file is located. Therefore, in this case, all you need to do to run the script is to type the following inside a terminal:

▶ Execute in Shell #1

```
In [ ]:
python pyscript1.py
```

You should get an output like this one:

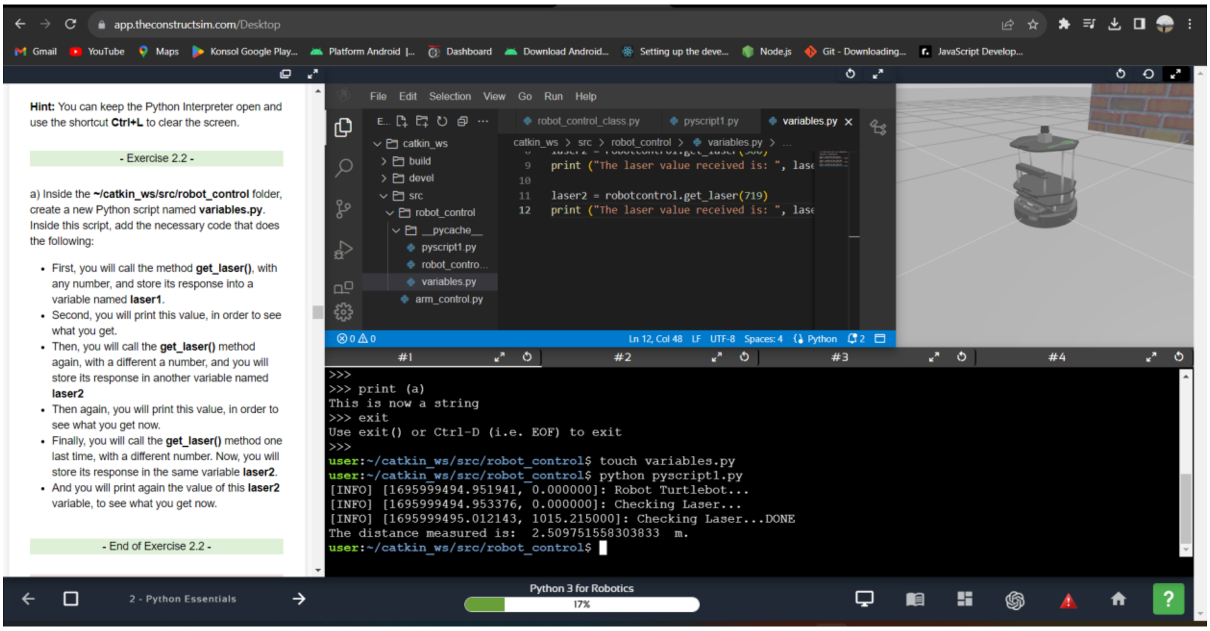
```
[INFO] [1676363986.715301, 0.000000]:
[INFO] [1676363986.716817, 0.000000]:
[INFO] [1676363986.996523, 132.00400
The distance measured is:  2.5138711
```

- End of Exercise 2.1 -

```
 robot_control_class import RobotControl

 RobotControl()

rc.get_laser(360)

t ("The distance measured is: ", a, " m.")
```

```
user:~$ cd ~/catkin_ws/src/
user:~/catkin_ws/src$ mkdir robot_control
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ touch pyscript1.py
user:~/catkin_ws/src/robot_control$ touch robot_control_class.py
user:~/catkin_ws/src/robot_control$ python pyscript1.py
[INFO] [1695999094.387352, 0.000000]: Robot Turtlebot...
[INFO] [1695999094.388999, 0.000000]: Checking Laser...
[INFO] [1695999094.604035, 616.919000]: Checking Laser...DONE
The distance measured is:  2.5020806789398193  m.
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 2.2



Hint: You can keep the Python Interpreter open and use the shortcut **Ctrl+L** to clear the screen.

- Exercise 2.2 -

a) Inside the **~/catkin_ws/src/robot_control** folder, create a new Python script named **variables.py**. Inside this script, add the necessary code that does the following:

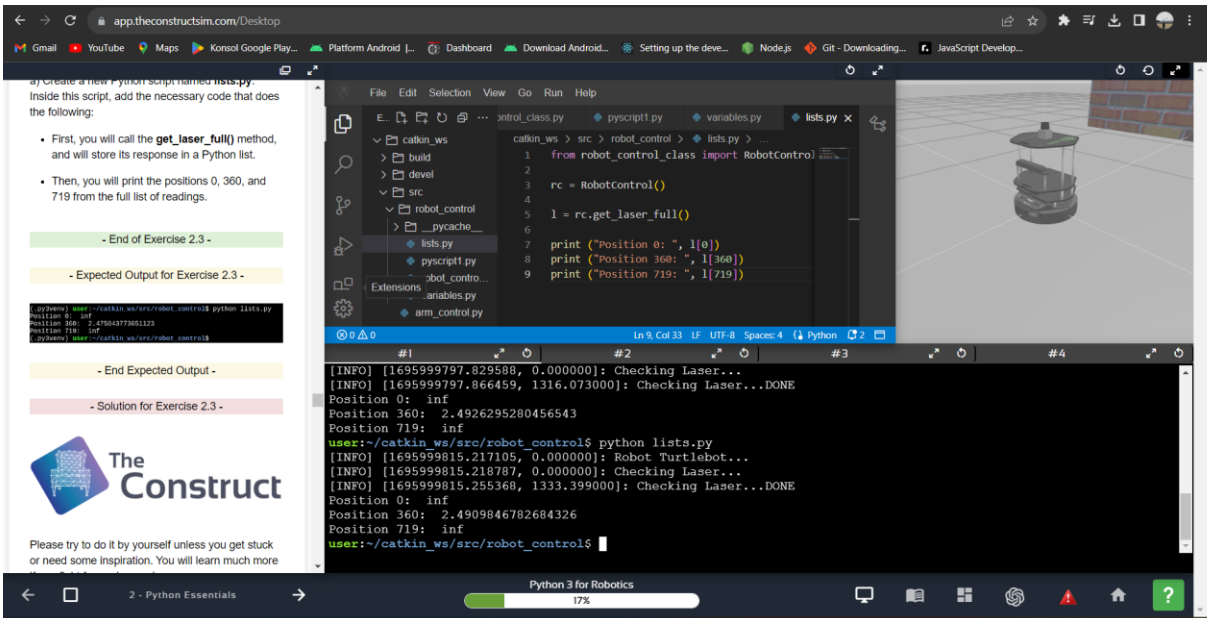
- First, you will call the method **get_laser()**, with any number, and store its response into a variable named **laser1**.
- Second, you will print this value, in order to see what you get.
- Then, you will call the **get_laser()** method again, with a different a number, and you will store its response in another variable named **laser2**
- Then again, you will print this value, in order to see what you get now.
- Finally, you will call the **get_laser()** method one last time, with a different number. Now, you will store its response in the same variable **laser2**.
- And you will print again the value of this **laser2** variable, to see what you get now.

- End of Exercise 2.2 -

```
print ("The laser value received is: ", lase

laser2 = robotcontrol.get_laser(719)
print ("The laser value received is: ", lase
```

```
>>>
>>> print (a)
This is now a string
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>>
user:~/catkin_ws/src/robot_control$ touch variables.py
user:~/catkin_ws/src/robot_control$ python pyscript1.py
[INFO] [1695999494.951941, 0.000000]: Robot Turtlebot...
[INFO] [1695999494.953376, 0.000000]: Checking Laser...
[INFO] [1695999495.012143, 1015.215000]: Checking Laser...DONE
The distance measured is:  2.509751558303833  m.
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 2.3



a) Create a new Python script named lists.py.
Inside this script, add the necessary code that does the following:

- First, you will call the **get_laser_full()** method, and will store its response in a Python list.
- Then, you will print the positions 0, 360, and 719 from the full list of readings.

- End of Exercise 2.3 -

- Expected Output for Exercise 2.3 -

```
(.py3venv) user:~/catkin_ws/src/robot_control$ python lists.py
Position 0:   inf
Position 360:   2.475043773651123
Position 719:   inf
(.py3venv) user:~/catkin_ws/src/robot_control$
```

- End Expected Output -

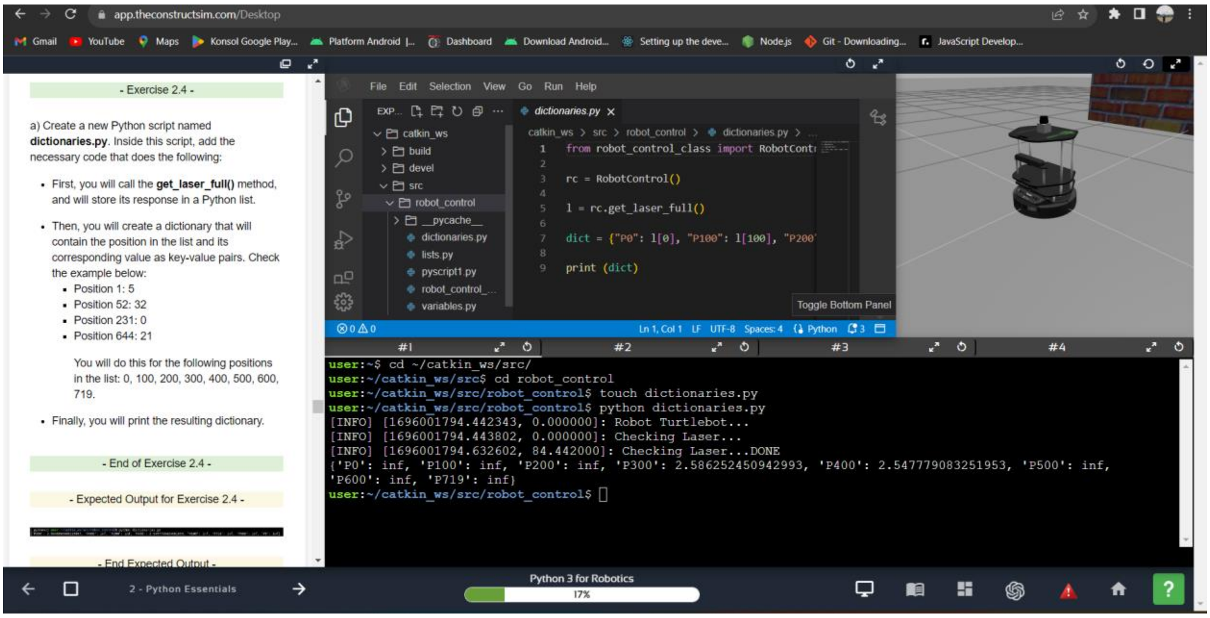
- Solution for Exercise 2.3 -

Please try to do it by yourself unless you get stuck or need some inspiration. You will learn much more

```python
from robot_control_class import RobotControl

rc = RobotControl()

l = rc.get_laser_full()

print ("Position 0: ", l[0])
print ("Position 360: ", l[360])
print ("Position 719: ", l[719])
```

```
[INFO] [1695999797.829588, 0.000000]: Checking Laser...
[INFO] [1695999797.866459, 1316.073000]: Checking Laser...DONE
Position 0:  inf
Position 360:  2.4926295280456543
Position 719:  inf
user:~/catkin_ws/src/robot_control$ python lists.py
[INFO] [1695999815.217105, 0.000000]: Robot Turtlebot...
[INFO] [1695999815.218787, 0.000000]: Checking Laser...
[INFO] [1695999815.255368, 1333.399000]: Checking Laser...DONE
Position 0:  inf
Position 360:  2.4909846782684326
Position 719:  inf
user:~/catkin_ws/src/robot_control$
```

EXERCISE 2.4



- Exercise 2.4 -

a) Create a new Python script named **dictionaries.py**. Inside this script, add the necessary code that does the following:

- First, you will call the **get_laser_full()** method, and will store its response in a Python list.
- Then, you will create a dictionary that will contain the position in the list and its corresponding value as key-value pairs. Check the example below:
    - Position 1: 5
    - Position 52: 32
    - Position 231: 0
    - Position 644: 21

  You will do this for the following positions in the list: 0, 100, 200, 300, 400, 500, 600, 719.
- Finally, you will print the resulting dictionary.

- End of Exercise 2.4 -

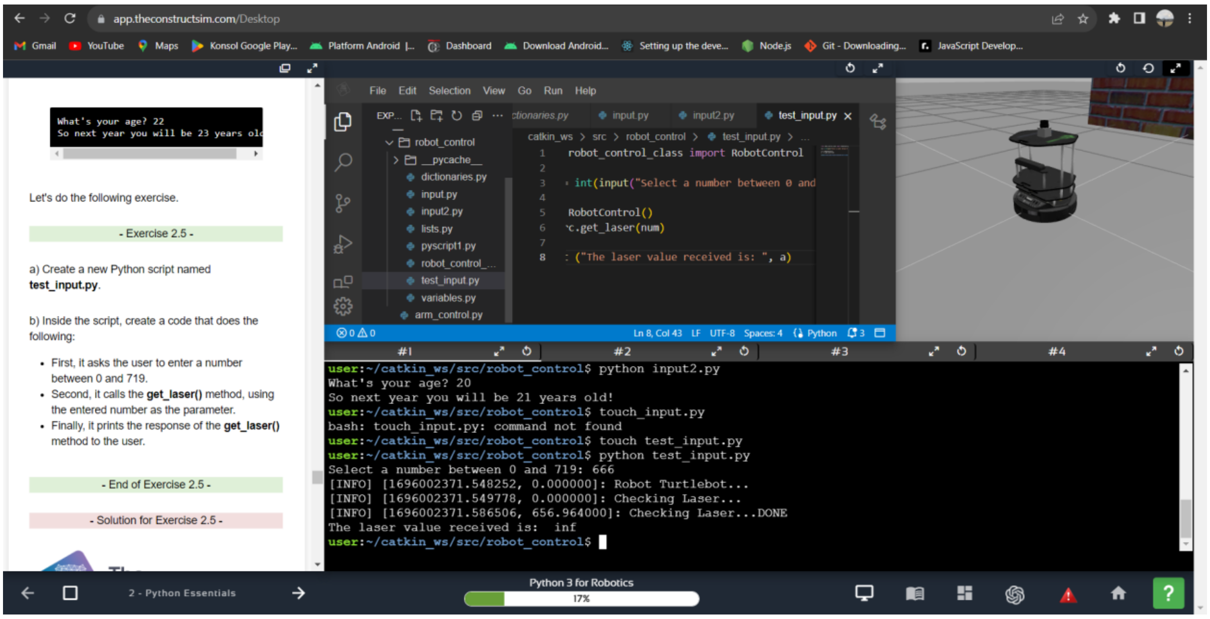
- Expected Output for Exercise 2.4 -

- End Expected Output -

```python
from robot_control_class import RobotCont

rc = RobotControl()

l = rc.get_laser_full()

dict = {"P0": l[0], "P100": l[100], "P200'

print (dict)
```

```
user:~$ cd ~/catkin_ws/src/
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ touch dictionaries.py
user:~/catkin_ws/src/robot_control$ python dictionaries.py
[INFO] [1696001794.442343, 0.000000]: Robot Turtlebot...
[INFO] [1696001794.443802, 0.000000]: Checking Laser...
[INFO] [1696001794.632602, 84.442000]: Checking Laser...DONE
{'P0': inf, 'P100': inf, 'P200': inf, 'P300': 2.586252450942993, 'P400': 2.547779083251953, 'P500': inf,
'P600': inf, 'P719': inf}
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 2.5



```
What's your age? 22
So next year you will be 23 years ol
```

Let's do the following exercise.

- Exercise 2.5 -

a) Create a new Python script named **test_input.py**.

b) Inside the script, create a code that does the following:

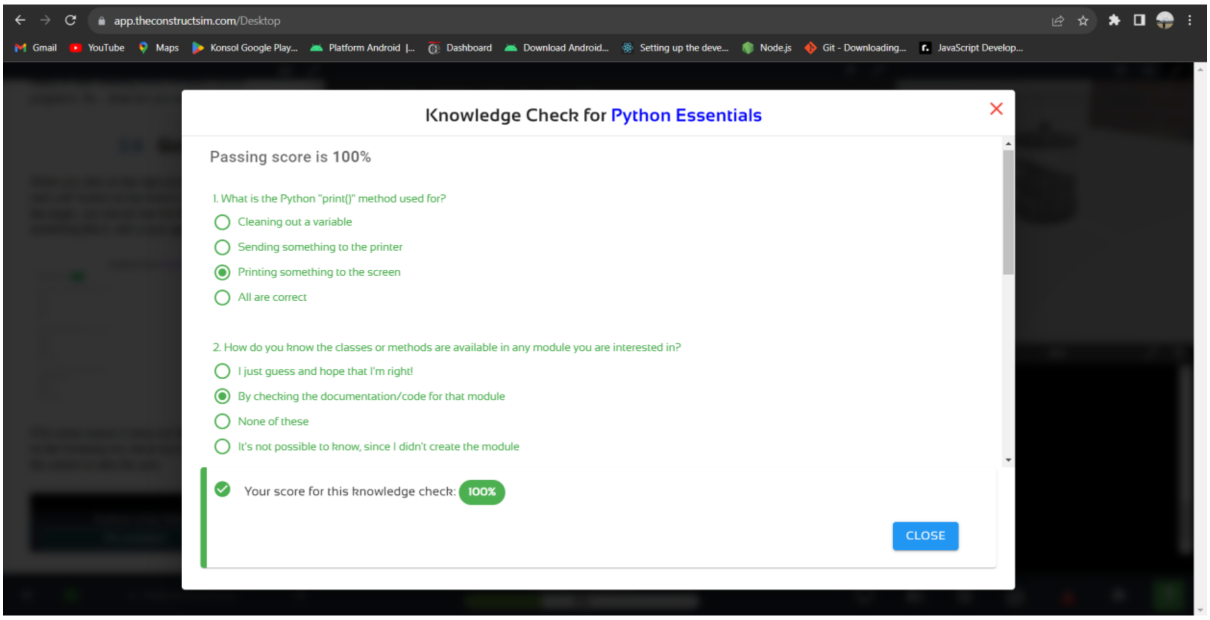
- First, it asks the user to enter a number between 0 and 719.
- Second, it calls the **get_laser()** method, using the entered number as the parameter.
- Finally, it prints the response of the **get_laser()** method to the user.

- End of Exercise 2.5 -

- Solution for Exercise 2.5 -

```
robot_control_class import RobotControl

: int(input("Select a number between 0 and

RobotControl()
rc.get_laser(num)

: ("The laser value received is: ", a)
```

```
user:~/catkin_ws/src/robot_control$ python input2.py
What's your age? 20
So next year you will be 21 years old!
user:~/catkin_ws/src/robot_control$ touch_input.py
bash: touch_input.py: command not found
user:~/catkin_ws/src/robot_control$ touch test_input.py
user:~/catkin_ws/src/robot_control$ python test_input.py
Select a number between 0 and 719: 666
[INFO] [1696002371.548252, 0.000000]: Robot Turtlebot...
[INFO] [1696002371.549778, 0.000000]: Checking Laser...
[INFO] [1696002371.586506, 656.964000]: Checking Laser...DONE
The laser value received is:  inf
user:~/catkin_ws/src/robot_control$
```



Knowledge Check for Python Essentials

Passing score is 100%

1. What is the Python "print()" method used for?

○ Cleaning out a variable
○ Sending something to the printer
● Printing something to the screen
○ All are correct

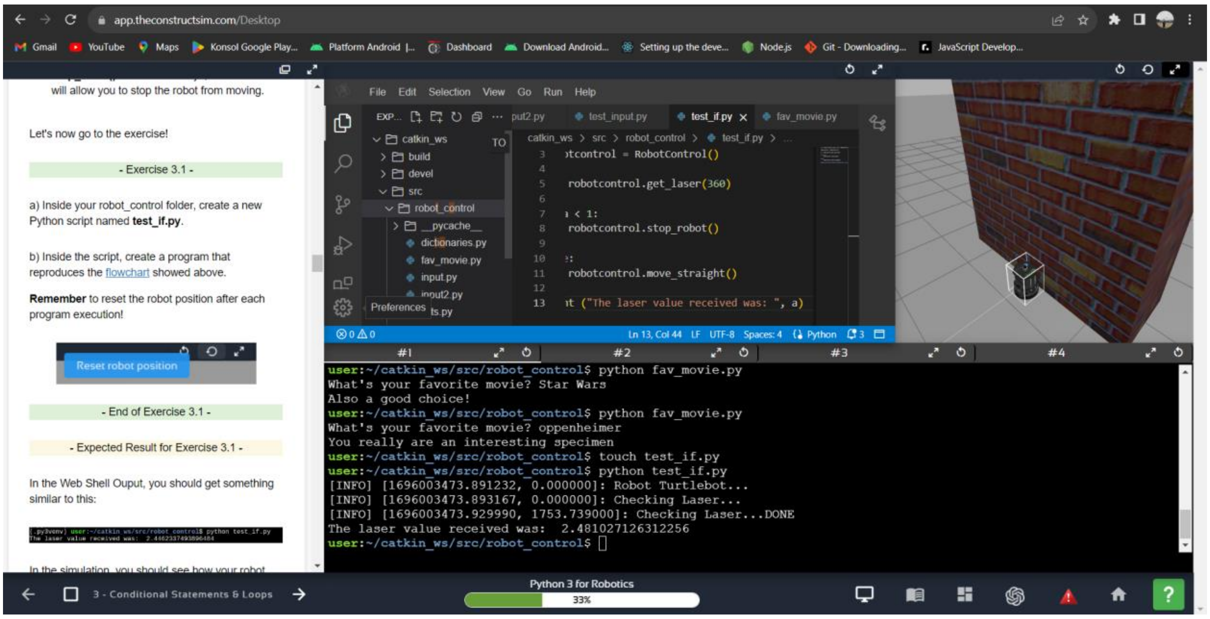
2. How do you know the classes or methods are available in any module you are interested in?

○ I just guess and hope that I'm right!
● By checking the documentation/code for that module
○ None of these
○ It's not possible to know, since I didn't create the module

✓ Your score for this knowledge check: 100%

- CONDITIONAL STATEMENTS AND LOOPS

EXERCISE 3.1



will allow you to stop the robot from moving.

Let's now go to the exercise!

- Exercise 3.1 -

a) Inside your robot_control folder, create a new Python script named **test_if.py**.

b) Inside the script, create a program that reproduces the flowchart showed above.

Remember to reset the robot position after each program execution!

- End of Exercise 3.1 -

- Expected Result for Exercise 3.1 -

In the Web Shell Ouput, you should get something similar to this:

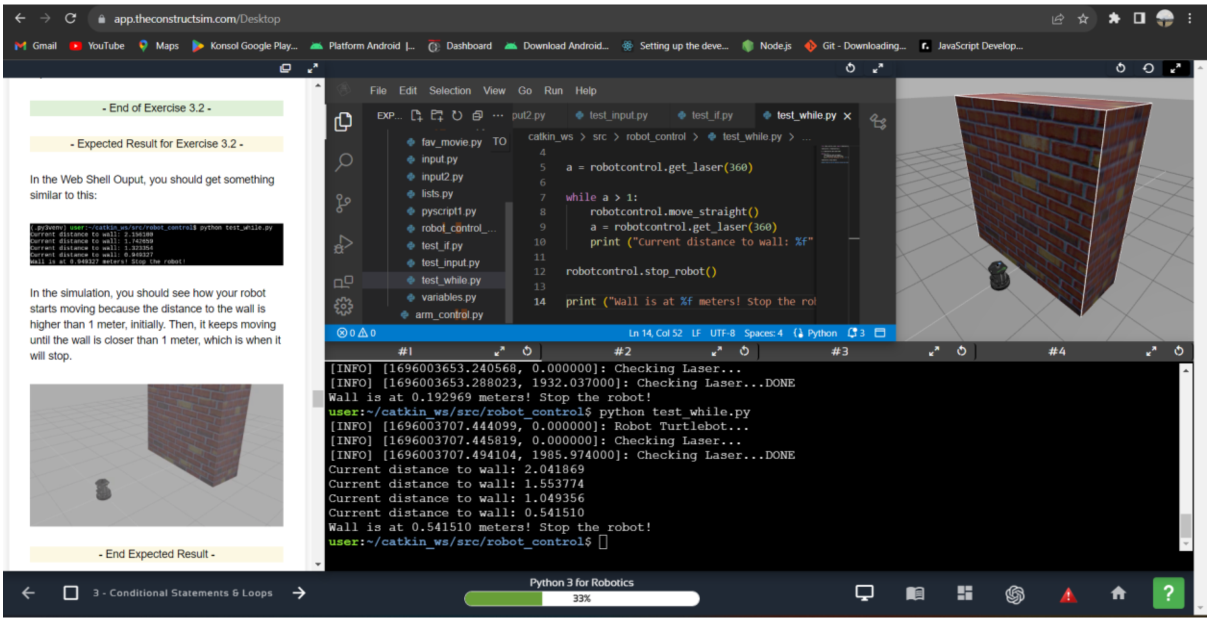
```
(.py3venv) user:~/catkin_ws/src/robot_control$ python test_if.py
The laser value received was:  2.4462337493896484
```

In the simulation, you should see how your robot

```
ɔtcontrol = RobotControl()

 robotcontrol.get_laser(360)

ı < 1:
 robotcontrol.stop_robot()

ɜ:
 robotcontrol.move_straight()

ıt ("The laser value received was: ", a)
```

```
user:~/catkin_ws/src/robot_control$ python fav_movie.py
What's your favorite movie? Star Wars
Also a good choice!
user:~/catkin_ws/src/robot_control$ python fav_movie.py
What's your favorite movie? oppenheimer
You really are an interesting specimen
user:~/catkin_ws/src/robot_control$ touch test_if.py
user:~/catkin_ws/src/robot_control$ python test_if.py
[INFO] [1696003473.891232, 0.000000]: Robot Turtlebot...
[INFO] [1696003473.893167, 0.000000]: Checking Laser...
[INFO] [1696003473.929990, 1753.739000]: Checking Laser...DONE
The laser value received was:  2.481027126312256
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 3.2



- End of Exercise 3.2 -

- Expected Result for Exercise 3.2 -

In the Web Shell Ouput, you should get something similar to this:

```
(.py3venv) user:~/catkin_ws/src/robot_control$ python test_while.py
Current distance to wall: 2.156188
Current distance to wall: 1.742659
Current distance to wall: 1.323354
Current distance to wall: 0.949327
Wall is at 0.949327 meters! Stop the robot!
```

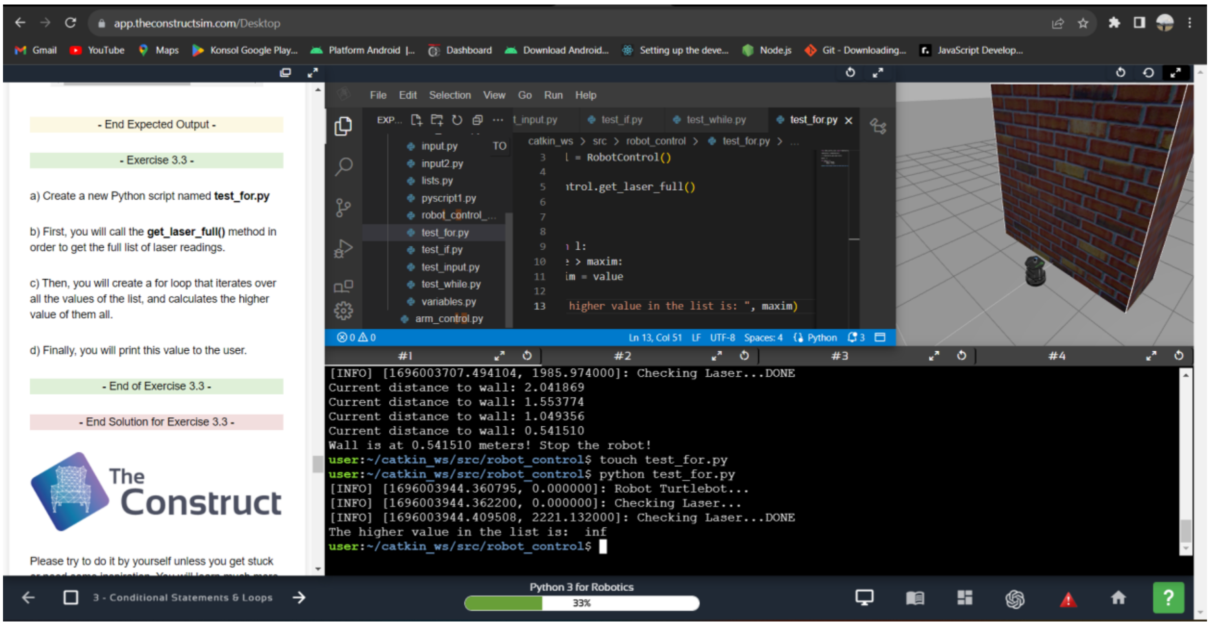
In the simulation, you should see how your robot starts moving because the distance to the wall is higher than 1 meter, initially. Then, it keeps moving until the wall is closer than 1 meter, which is when it will stop.

- End Expected Result -

```

a = robotcontrol.get_laser(360)

while a > 1:
    robotcontrol.move_straight()
    a = robotcontrol.get_laser(360)
    print ("Current distance to wall: %f"

robotcontrol.stop_robot()

print ("Wall is at %f meters! Stop the ro
```

```
[INFO] [1696003653.240568, 0.000000]: Checking Laser...
[INFO] [1696003653.288023, 1932.037000]: Checking Laser...DONE
Wall is at 0.192969 meters! Stop the robot!
user:~/catkin_ws/src/robot_control$ python test_while.py
[INFO] [1696003707.444099, 0.000000]: Robot Turtlebot...
[INFO] [1696003707.445819, 0.000000]: Checking Laser...
[INFO] [1696003707.494104, 1985.974000]: Checking Laser...DONE
Current distance to wall: 2.041869
Current distance to wall: 1.553774
Current distance to wall: 1.049356
Current distance to wall: 0.541510
Wall is at 0.541510 meters! Stop the robot!
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 3.3



- End Expected Output -

- Exercise 3.3 -

a) Create a new Python script named **test_for.py**

b) First, you will call the **get_laser_full()** method in order to get the full list of laser readings.

c) Then, you will create a for loop that iterates over all the values of the list, and calculates the higher value of them all.

d) Finally, you will print this value to the user.

- End of Exercise 3.3 -

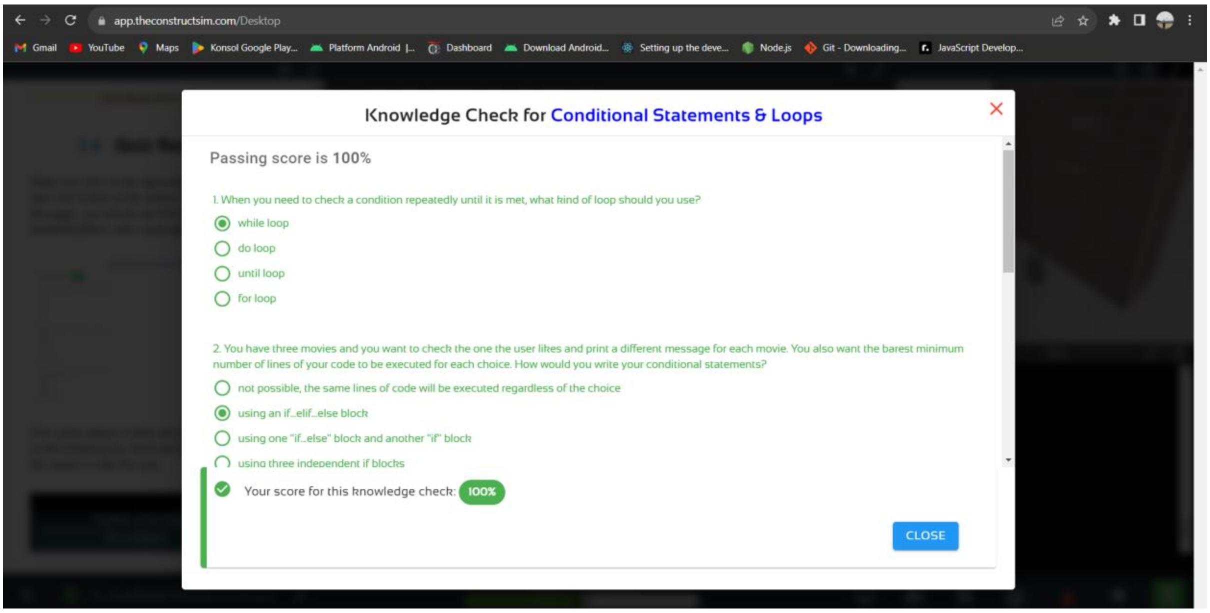
- End Solution for Exercise 3.3 -

Please try to do it by yourself unless you get stuck

```
3   l = RobotControl()
4
5   ıtrol.get_laser_full()
6
7
8
9   ı l:
10  ɛ > maxim:
11  im = value
12
13   higher value in the list is: ", maxim)
```

```
[INFO] [1696003707.494104, 1985.974000]: Checking Laser...DONE
Current distance to wall: 2.041869
Current distance to wall: 1.553774
Current distance to wall: 1.049356
Current distance to wall: 0.541510
Wall is at 0.541510 meters! Stop the robot!
user:~/catkin_ws/src/robot_control$ touch test_for.py
user:~/catkin_ws/src/robot_control$ python test_for.py
[INFO] [1696003944.360795, 0.000000]: Robot Turtlebot...
[INFO] [1696003944.362200, 0.000000]: Checking Laser...
[INFO] [1696003944.409508, 2221.132000]: Checking Laser...DONE
The higher value in the list is:  inf
user:~/catkin_ws/src/robot_control$
```

app.theconstructsim.com/Desktop
File Edit Selection View Go Run Help
input.py
input2.py
lists.py
pyscript1.py
robot_control_...
test_for.py
test_if.py
test_input.py
test_while.py
variables.py
arm_control.py
catkin_ws > src > robot_control > test_for.py
Ln 13, Col 51 LF UTF-8 Spaces: 4 Python
3 - Conditional Statements & Loops
Python 3 for Robotics
33%



Knowledge Check for Conditional Statements & Loops

Passing score is 100%

1. When you need to check a condition repeatedly until it is met, what kind of loop should you use?

● while loop
○ do loop
○ until loop
○ for loop

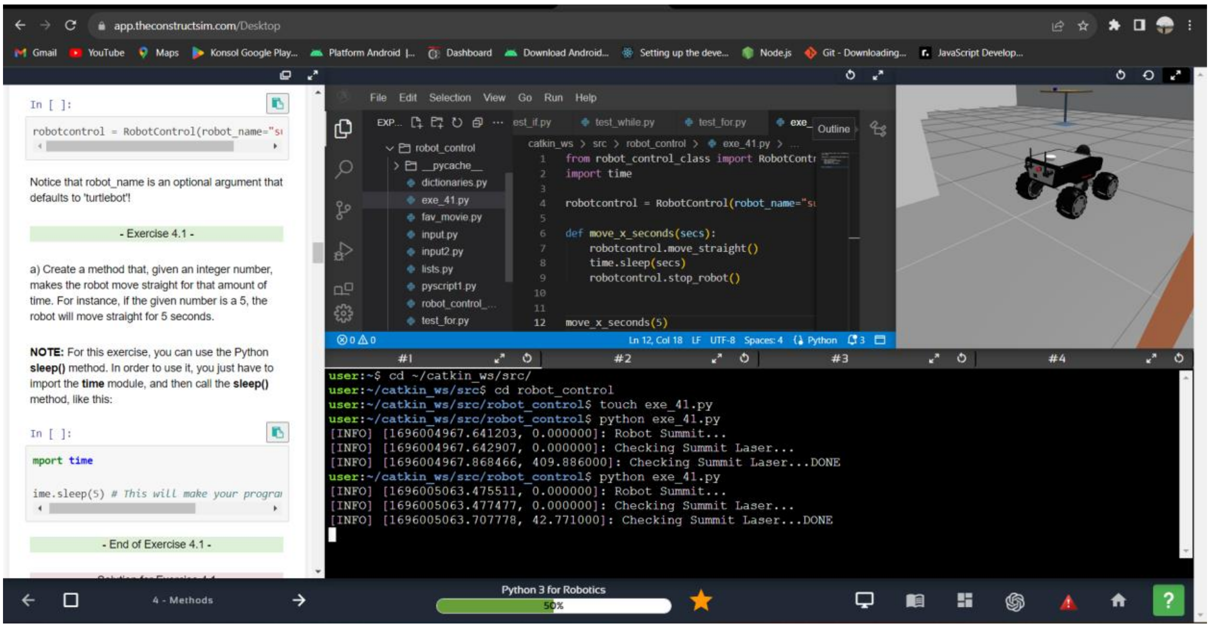
2. You have three movies and you want to check the one the user likes and print a different message for each movie. You also want the barest minimum number of lines of your code to be executed for each choice. How would you write your conditional statements?

○ not possible, the same lines of code will be executed regardless of the choice
● using an if...elif...else block
○ using one "if...else" block and another "if" block
○ using three independent if blocks

✓ Your score for this knowledge check: 100%

- METHODS

EXERCISE 4.1



```
In [ ]:
robotcontrol = RobotControl(robot_name="su
```

Notice that robot_name is an optional argument that defaults to 'turtlebot'!

- Exercise 4.1 -

a) Create a method that, given an integer number, makes the robot move straight for that amount of time. For instance, if the given number is a 5, the robot will move straight for 5 seconds.

NOTE: For this exercise, you can use the Python **sleep()** method. In order to use it, you just have to import the **time** module, and then call the **sleep()** method, like this:

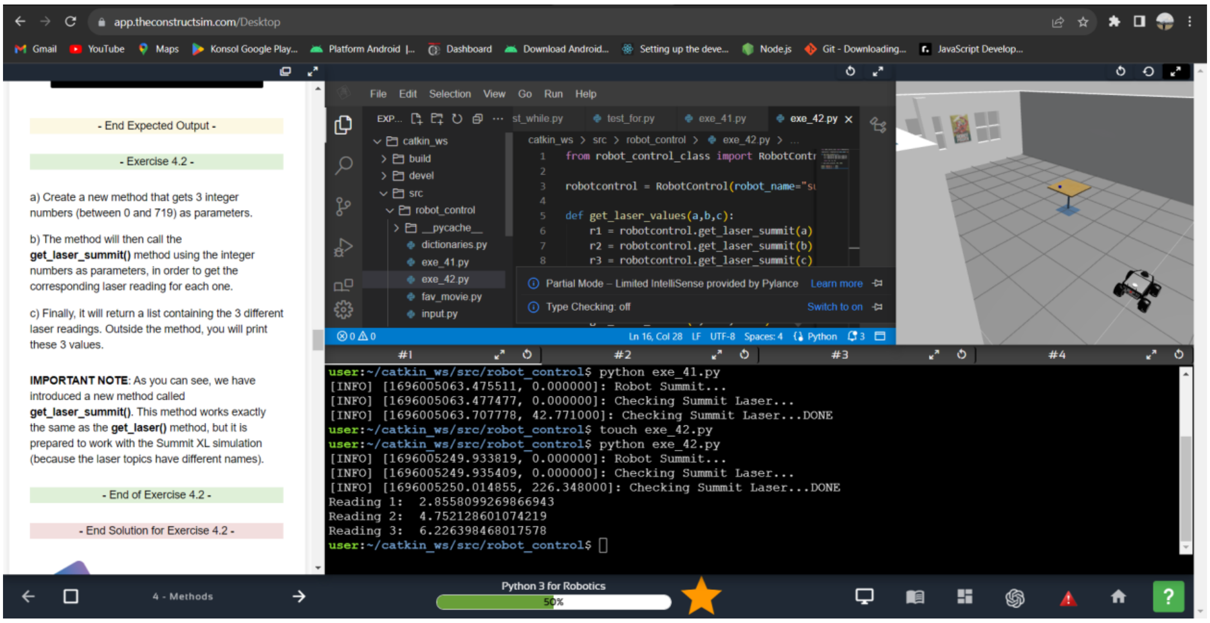
```
In [ ]:
mport time

ime.sleep(5) # This will make your progra
```

- End of Exercise 4.1 -

```
from robot_control_class import RobotContr
import time

robotcontrol = RobotControl(robot_name="su

def move_x_seconds(secs):
    robotcontrol.move_straight()
    time.sleep(secs)
    robotcontrol.stop_robot()


move_x_seconds(5)
```

```
user:~$ cd ~/catkin_ws/src/
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ touch exe_41.py
user:~/catkin_ws/src/robot_control$ python exe_41.py
[INFO] [1696004967.641203, 0.000000]: Robot Summit...
[INFO] [1696004967.642907, 0.000000]: Checking Summit Laser...
[INFO] [1696004967.868466, 409.886000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python exe_41.py
[INFO] [1696005063.475511, 0.000000]: Robot Summit...
[INFO] [1696005063.477477, 0.000000]: Checking Summit Laser...
[INFO] [1696005063.707778, 42.771000]: Checking Summit Laser...DONE
```

EXERCISE 4.2



- End Expected Output -

- Exercise 4.2 -

a) Create a new method that gets 3 integer numbers (between 0 and 719) as parameters.

b) The method will then call the **get_laser_summit()** method using the integer numbers as parameters, in order to get the corresponding laser reading for each one.

c) Finally, it will return a list containing the 3 different laser readings. Outside the method, you will print these 3 values.

IMPORTANT NOTE: As you can see, we have introduced a new method called **get_laser_summit()**. This method works exactly the same as the **get_laser()** method, but it is prepared to work with the Summit XL simulation (because the laser topics have different names).

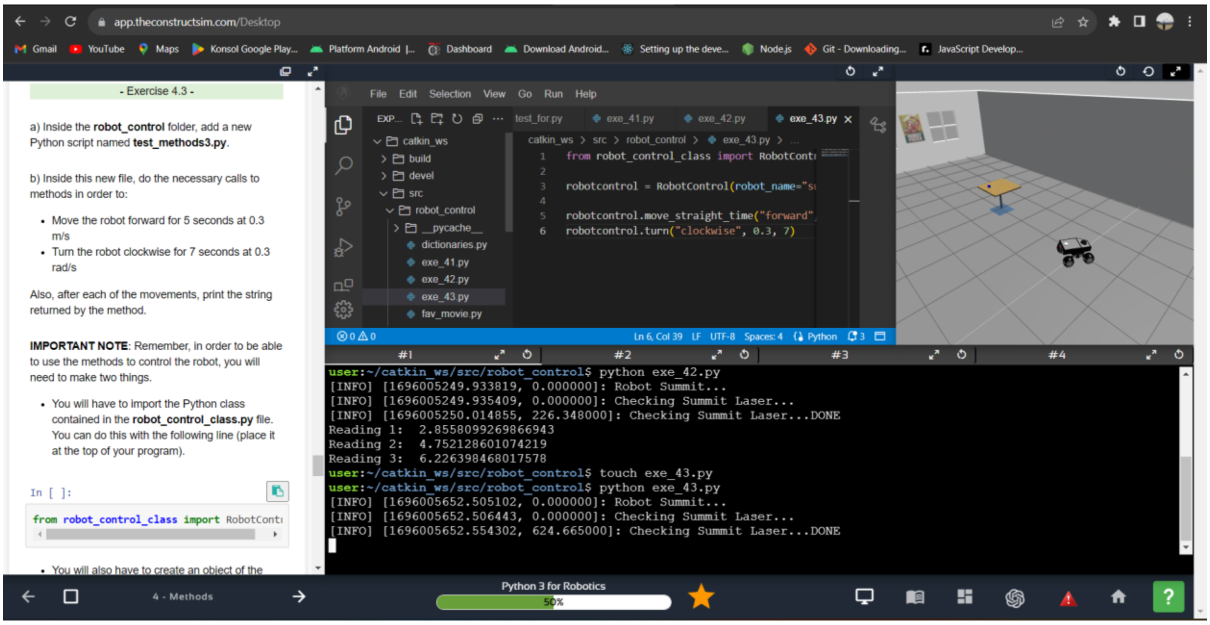
- End of Exercise 4.2 -

- End Solution for Exercise 4.2 -

```
from robot_control_class import RobotContr

robotcontrol = RobotControl(robot_name="su

def get_laser_values(a,b,c):
    r1 = robotcontrol.get_laser_summit(a)
    r2 = robotcontrol.get_laser_summit(b)
    r3 = robotcontrol.get_laser_summit(c)
```

```
user:~/catkin_ws/src/robot_control$ python exe_41.py
[INFO] [1696005063.475511, 0.000000]: Robot Summit...
[INFO] [1696005063.477477, 0.000000]: Checking Summit Laser...
[INFO] [1696005063.707778, 42.771000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ touch exe_42.py
user:~/catkin_ws/src/robot_control$ python exe_42.py
[INFO] [1696005249.933819, 0.000000]: Robot Summit...
[INFO] [1696005249.935409, 0.000000]: Checking Summit Laser...
[INFO] [1696005250.014855, 226.348000]: Checking Summit Laser...DONE
Reading 1:  2.8558099269866943
Reading 2:  4.752128601074219
Reading 3:  6.226398468017578
user:~/catkin_ws/src/robot_control$ 
```

EXERCISE 4.3



- Exercise 4.3 -

a) Inside the **robot_control** folder, add a new Python script named **test_methods3.py**.

b) Inside this new file, do the necessary calls to methods in order to:

- Move the robot forward for 5 seconds at 0.3 m/s
- Turn the robot clockwise for 7 seconds at 0.3 rad/s

Also, after each of the movements, print the string returned by the method.

IMPORTANT NOTE: Remember, in order to be able to use the methods to control the robot, you will need to make two things.

- You will have to import the Python class contained in the **robot_control_class.py** file. You can do this with the following line (place it at the top of your program).

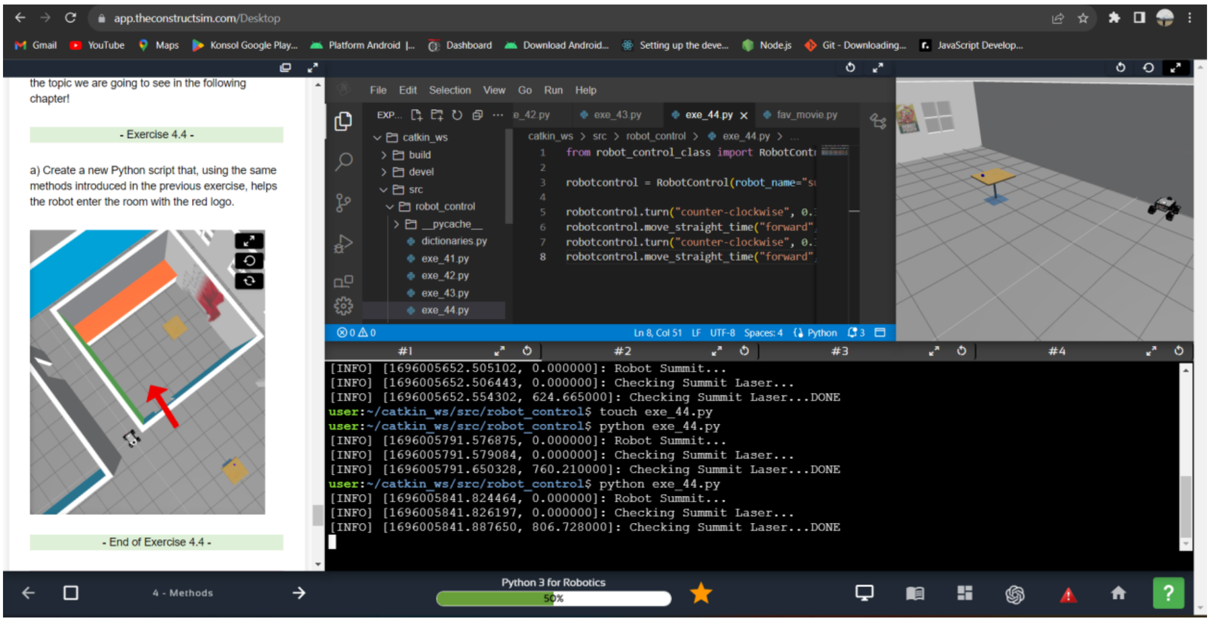
```
In [ ]:
from robot_control_class import RobotConti
```

- You will also have to create an object of the

```
from robot_control_class import RobotContr
robotcontrol = RobotControl(robot_name="su
robotcontrol.move_straight_time("forward",
robotcontrol.turn("clockwise", 0.3, 7)
```

```
user:~/catkin_ws/src/robot_control$ python exe_42.py
[INFO] [1696005249.933819, 0.000000]: Robot Summit...
[INFO] [1696005249.935409, 0.000000]: Checking Summit Laser...
[INFO] [1696005250.014855, 226.348000]: Checking Summit Laser...DONE
Reading 1:  2.8558099269866943
Reading 2:  4.752128601074219
Reading 3:  6.226398468017578
user:~/catkin_ws/src/robot_control$ touch exe_43.py
user:~/catkin_ws/src/robot_control$ python exe_43.py
[INFO] [1696005652.505102, 0.000000]: Robot Summit...
[INFO] [1696005652.506443, 0.000000]: Checking Summit Laser...
[INFO] [1696005652.554302, 624.665000]: Checking Summit Laser...DONE
```

EXERCISE 4.4



the topic we are going to see in the following chapter!

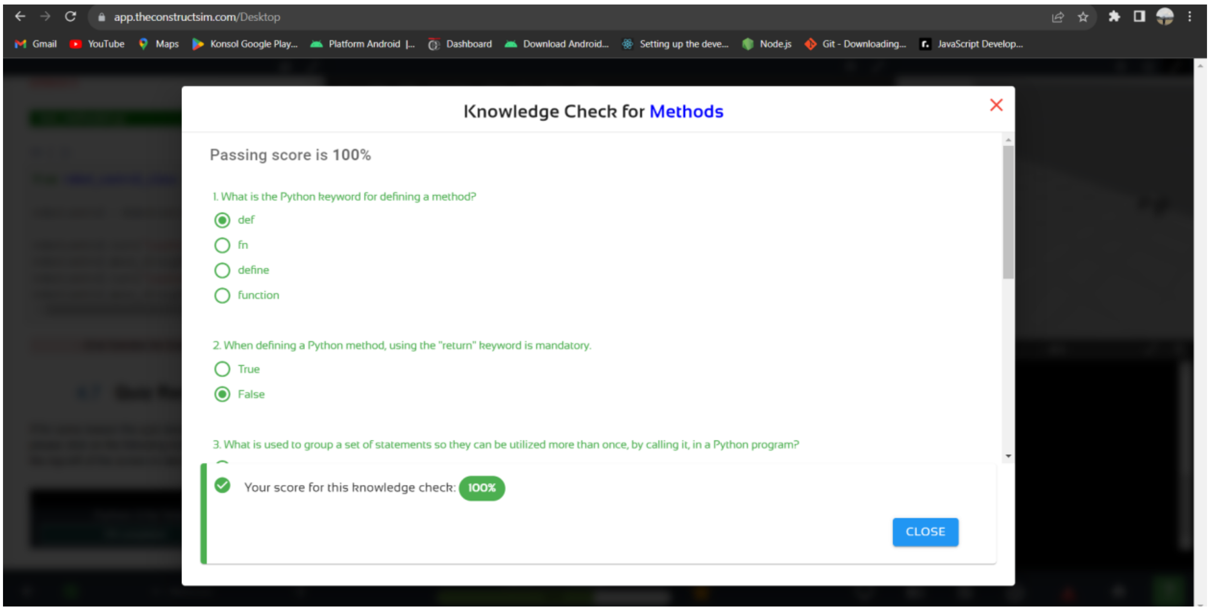
- Exercise 4.4 -

a) Create a new Python script that, using the same methods introduced in the previous exercise, helps the robot enter the room with the red logo.

- End of Exercise 4.4 -

```
from robot_control_class import RobotContr
robotcontrol = RobotControl(robot_name="su
robotcontrol.turn("counter-clockwise", 0.
robotcontrol.move_straight_time("forward",
robotcontrol.turn("counter-clockwise", 0.
robotcontrol.move_straight_time("forward",
```

```
[INFO] [1696005652.505102, 0.000000]: Robot Summit...
[INFO] [1696005652.506443, 0.000000]: Checking Summit Laser...
[INFO] [1696005652.554302, 624.665000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ touch exe_44.py
user:~/catkin_ws/src/robot_control$ python exe_44.py
[INFO] [1696005791.576875, 0.000000]: Robot Summit...
[INFO] [1696005791.579084, 0.000000]: Checking Summit Laser...
[INFO] [1696005791.650328, 760.210000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python exe_44.py
[INFO] [1696005841.824464, 0.000000]: Robot Summit...
[INFO] [1696005841.826197, 0.000000]: Checking Summit Laser...
[INFO] [1696005841.887650, 806.728000]: Checking Summit Laser...DONE
```



Knowledge Check for Methods

Passing score is 100%

1. What is the Python keyword for defining a method?

● def
○ fn
○ define
○ function

2. When defining a Python method, using the "return" keyword is mandatory.

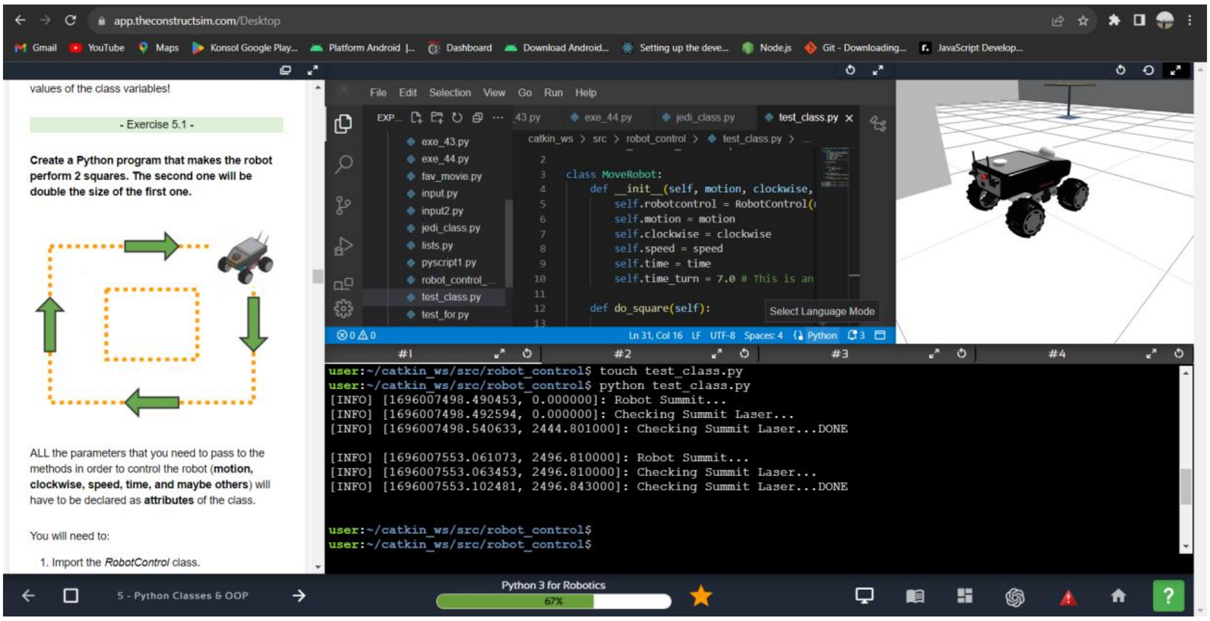
○ True
● False

3. What is used to group a set of statements so they can be utilized more than once, by calling it, in a Python program?

Your score for this knowledge check: 100%

- Classes and Object Oriented Programming

EXERCISE 5.1



values of the class variables!

- Exercise 5.1 -

Create a Python program that makes the robot perform 2 squares. The second one will be double the size of the first one.

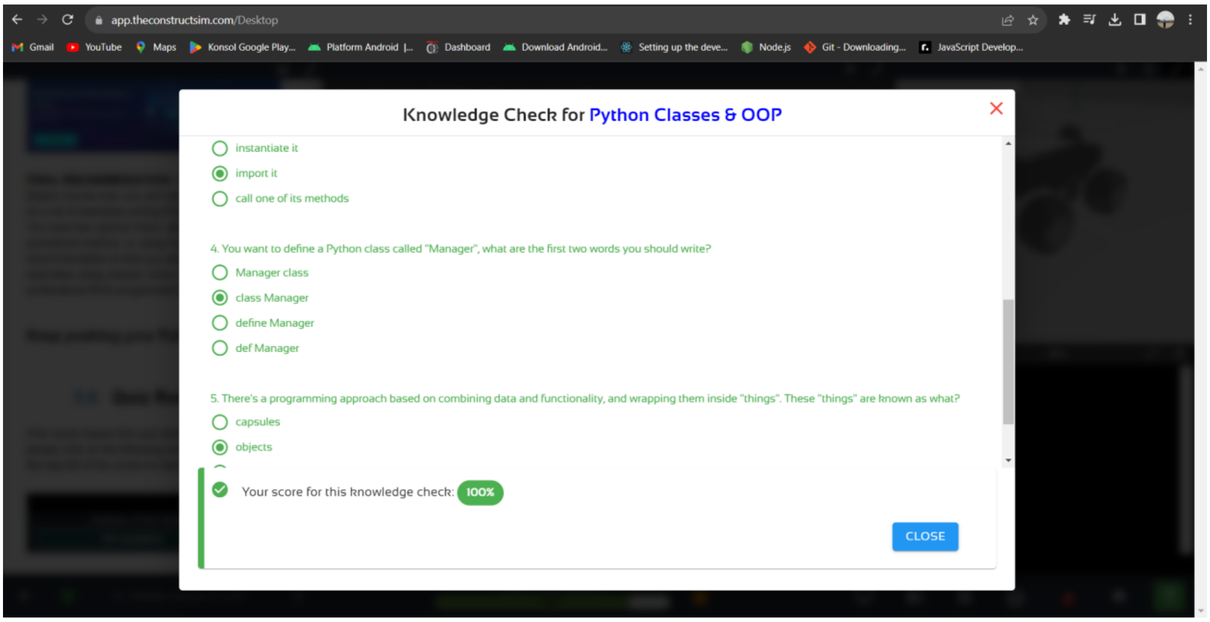
ALL the parameters that you need to pass to the methods in order to control the robot (**motion, clockwise, speed, time, and maybe others**) will have to be declared as **attributes** of the class.

You will need to:

1. Import the *RobotControl* class.

```
class MoveRobot:
    def __init__(self, motion, clockwise,
        self.robotcontrol = RobotControl(
        self.motion = motion
        self.clockwise = clockwise
        self.speed = speed
        self.time = time
        self.time_turn = 7.0 # This is an

    def do_square(self):
```

```
user:~/catkin_ws/src/robot_control$ touch test_class.py
user:~/catkin_ws/src/robot_control$ python test_class.py
[INFO] [1696007498.490453, 0.000000]: Robot Summit...
[INFO] [1696007498.492594, 0.000000]: Checking Summit Laser...
[INFO] [1696007498.540633, 2444.801000]: Checking Summit Laser...DONE

[INFO] [1696007553.061073, 2496.810000]: Robot Summit...
[INFO] [1696007553.063453, 2496.810000]: Checking Summit Laser...
[INFO] [1696007553.102481, 2496.843000]: Checking Summit Laser...DONE


user:~/catkin_ws/src/robot_control$
user:~/catkin_ws/src/robot_control$
```



Knowledge Check for Python Classes & OOP

○ instantiate it

● import it

○ call one of its methods

4. You want to define a Python class called "Manager", what are the first two words you should write?

○ Manager class

● class Manager

○ define Manager

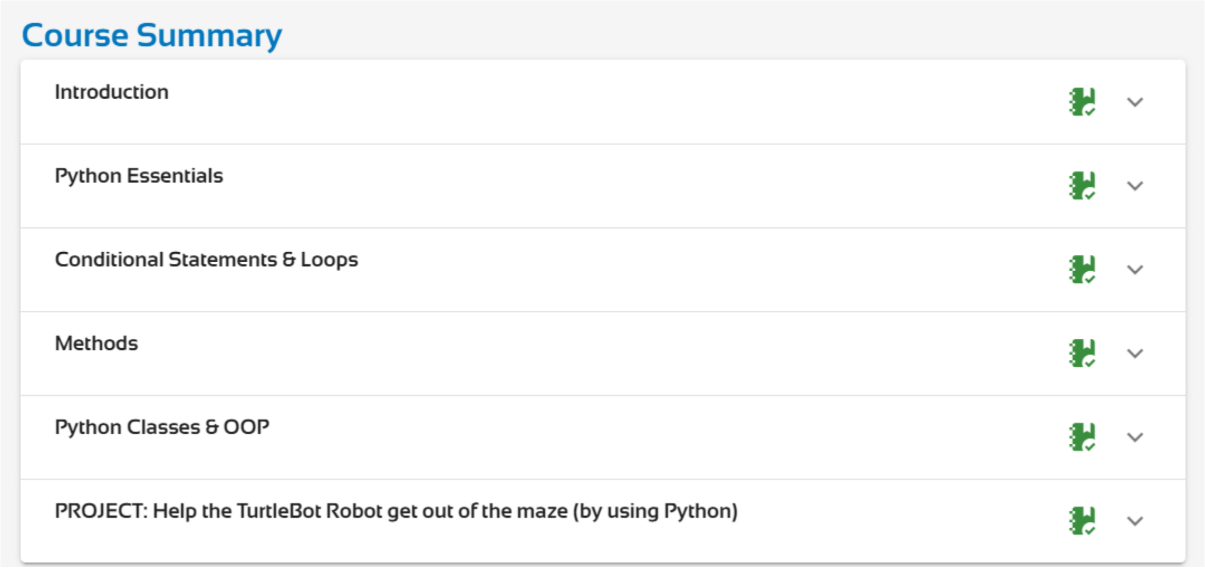
○ def Manager

5. There's a programming approach based on combining data and functionality, and wrapping them inside "things". These "things" are known as what?

○ capsules

● objects

✓ Your score for this knowledge check: 100%



Course Summary

- Introduction
- Python Essentials
- Conditional Statements & Loops
- Methods
- Python Classes & OOP
- PROJECT: Help the TurtleBot Robot get out of the maze (by using Python)