

Nama : Muhammad Ismail Ibadurrahman
NIM : 1103204239
Mata Kuliah : Robotika dan Sistem Cerdas

1. Screenshoot command-command
2. Screenshoot Course Python
- 2.1. Introduction (*Maaf yang 100% lupa di terdokumentasikan)

Knowledge Check for [Introduction](#)

Passing score is 100%

1. A robotics developer does NOT need to learn Python.

- ☒ False
☐ True

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

- ☐ False
☒ True

3. Python files have the extension _____

- ☐ .python
☒ .py
☐ .txt
☐ .file

4. Where is a Python program executed?

- ☐ none of the above
☐ simulation
☒ IDE
☐ terminal

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

- ☒ False
☐ True

Your score for this knowledge check: **80%**

CLOSE

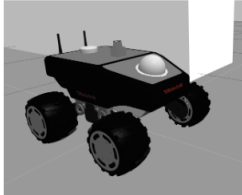
- How to change behavior based on conditions
- How to create methods that can be called from other places of the code
- How to encapsulate the code into classes so you can have clean and robust code

1.4 How will you learn all this?

You will learn through hands-on experience from day one! In the Robot Ignite Academy, we strongly believe that the best way to learn is by practicing, practicing, and then... practicing some more!

Since this is a Python Course, we are going to focus on the programming language, leaving robotics aside. And during the course, you are going to be able to interact with the simulation environment as well. Specifically, you are going to work with a Summit XL robot.

Summit XL:



Also, for the Course Project, you'll be using a Turtlebot robot.

Turtlebot:

```
File Edit Selection View Go Run Help
catkin_ws > src > arm_control.py > ...
1 from smart_grasping_sandbox.smart_grasper import SmartGrasper
2 from tf.transformations import quaternion_from_euler
3 from math import pi
4 import time
5
6 sgs = SmartGrasper()
7
8 sgs.pick()
9
10 sgs.reset_world()
```

Ln 10, Col 18 LF UTF-8 Spaces: 4 Python 3.2

```
#1 #2 #3 #4
[WARN] [1696047517.269716251, 70.855000000]: Link H1_F3_palm_link has visual geometry but no collision geometry. Collision geometry will be left empty. Fix your URDF file by explicitly specifying collision geometry.
[WARN] [1696047517.392857518, 70.929000000]: Kinematics solver doesn't support #attempts anymore, but only a timeout.
Please remove the parameter '/robot_description_kinematics/arm/kinematics_solver_attempts' from your configuration.
[INFO] [1696047518.419598802, 71.662000000]: Ready to take commands for planning group arm.
[INFO] [1696047518.933699962, 72.044000000]: Ready to take commands for planning group hand.
[INFO] [1696047519.380375, 72.316000]: STARTING CONTROLLERS
[INFO] [1696047519.493362, 72.318000]: Moving to Pregrasp
[INFO] [1696047519.729345534, 72.487000000]: ABORTED: Solution found but controller failed during execution
[INFO] [1696047521.816004, 73.844000]: Grasping
[INFO] [1696047533.692504693, 81.807000000]: ABORTED: Solution found but controller failed during execution
[INFO] [1696047533.793001, 81.868000]: Lifting
[INFO] [1696047538.780366, 84.919000]: STARTING CONTROLLERS
user:~/catkin_ws/src$
```

Python 3 for Robotics

1 - Introduction

100%

2.2. Python Essentials

2.2.1. Quiz

Knowledge Check for Python Essentials

Passing score is 100%

1. The following are data types in Python, EXCEPT:

☐ Dictionaries

☐ Tuples

☒ All are data types

☐ Lists

2. Which of the following statements is false about data if data = {'a': 1, 'b': 2, 'c': 3}?

☐ data is a dictionary

☐ data has three keys

☐ data['b'] is equal to 2

☒ data is a list

3. Let i = "i" and got_it = "got it". What is the value of the expression 'i + got_it'?

☐ igotit

☐ igot it

☒ i got it

☐ got i

4. If alphabet = "ABCDE", what is alphabet[2]?

☒ S

☐ B

☐ ABCDE

☐ ASCDE

5. A _____ is a container that stores some data. It can be a number, text, or more complex data types.

☐ class

☐ method


☐ type

☒ variable

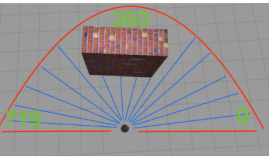
✓ Your score for this knowledge check: 100%

CLOSE

2.2.2. Exercise 2.1



these beams are projected from the laser in all directions, covering range of 180 degrees in front of the robot, more or less. Check out the image below:



this image represents the laser beams (although in reality there are many more, we cannot draw the 720!!) being projected from the laser in a range of 180 degrees. Also, keep in mind the numbers that appear in this image, since they will be very important in just a moment.

what the numbers in this image mean are the following:

- If we pass the number 0 to the `get_laser()` method, we will get the reading of the first laser beam at the right side of the robot.
- If we pass the number 360 to the `get_laser()` method, we will get the reading of the laser beam in front of the robot.
- If we pass the number 719 to the `get_laser()` method, we will get the reading of the last laser beam at the left side of the robot.
- And the same applies to all numbers in-between those.

```
File Edit Selection View Go Run Help
catkin_ws > src > robot_control > pyscript1.py > ...
1 from robot_control_class import RobotControl
2
3 rc = RobotControl()
4
5 a = rc.get_laser(360)
6
7 print("The distance measured is: ", a, " m.")
```

Ln 7, Col 47 1F UTF-8 Spaces: 4 Python 3.7.2

```
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$ ls
pyscript1.py robot_control_class.py
user:~/catkin_ws/src/robot_control$ python pyscript1.py
[INFO] [1696048452.780237, 0.000000]: Robot Turtlebot...
[INFO] [1696048452.782025, 0.000000]: Checking Laser...
[INFO] [1696048452.967025, 799.315000]: Checking Laser...DONE
The distance measured is: 2.4815642833709717 m.
user:~/catkin_ws/src/robot_control$ python pyscript1.py
[INFO] [1696048591.339989, 0.000000]: Robot Turtlebot...
[INFO] [1696048591.342098, 0.000000]: Checking Laser...
[INFO] [1696048591.402799, 921.169000]: Checking Laser...DONE
The distance measured is: 2.4948785305023193 m.
user:~/catkin_ws/src/robot_control$
```

2 - Python Essentials Python 3 for Robotics 100%

Source Code :

2.2.3. Exercise 2.2

Source Code :

#Muhammad Ismail Ibadurrahman_1103204239

```
from robot_control_class import RobotControl
```

```
robotcontrol = RobotControl()
```

#Refer Fungsi RobotControl dari librarynya

```
laser1 = robotcontrol.get_laser(0)
```

#deklarasi variabel laser 1 dengan fungsi yang ada didalam library robotcontrol dengan parameter 0 *derajat

```
print ("The laser value received is: ", laser1)
```

#print output laser1

```
laser2 = robotcontrol.get_laser(360)
```

#deklarasi variabel laser 1 dengan fungsi yang ada didalam library robotcontrol 360 *derajat

```
print ("The laser value received is: ", laser2)
```

#print output laser2

```
laser3 = robotcontrol.get_laser(719)
```

#deklarasi variabel laser 1 dengan fungsi yang ada didalam library robotcontrol 719 *derajat

```
print ("The laser value received is: ", laser3)
```

#print output variabel laser3

Output :

```
AttributeError: 'int' object has no attribute 'laser_msg'
user:~/catkin_ws/src/robot_control$ python variables.py
[INFO] [1696051038.740639, 0.000000]: Robot Turtlebot...
[INFO] [1696051038.742312, 0.000000]: Checking Laser...
[INFO] [1696051038.778418, 3318.629000]: Checking Laser...DONE
The laser value received is: inf
The laser value received is: 2.4792051315307617
The laser value received is: inf
user:~/catkin_ws/src/robot_control$
```

2.2.4. Exercise 2.3

Source Code :

#Muhammad Ismail Ibadurrahman_1103204239

```
from robot_control_class import RobotControl
```

```
rc = RobotControl()
```

#Refer Fungsi RobotControl dari librarynya

```
l = rc.get_laser_full()
```

#deklarasi variabel l dengan fungsi yang ada didalam library robotcontrol

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

#print output l parameter 0 *derajat

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

#print output l parameter 360 *derajat

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

#print output l parameter 719 *derajat

Output :

```
user:~/catkin_ws/src/robot_control$ python lists.py
[INFO] [1696054231.786484, 0.000000]: Robot Turtlebot...
[INFO] [1696054231.787784, 0.000000]: Checking Laser...
[INFO] [1696054231.823422, 6501.381000]: Checking Laser...DONE
Position 0: inf
Position 360: 2.4521312713623047
Position 719: inf
user:~/catkin_ws/src/robot_control$
```

2.2.5. Exercise 2.4

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
from robot_control_class import RobotControl
#import library, fungsi RobotControl dari class robot control
rc = RobotControl()
#Deklarasi variabel rc, refer Fungsi RobotControl dari librarynya
l = rc.get_laser_full()
#deklarasi variabel l dengan fungsi yang ada didalam library robotcontrol
dict = {"P0": l[0], "P100": l[100], "P200": l[200], "P300": l[300], "P400": l[400], "P500":
l[500], "P600": l[600], "P719": l[719]}
#deklarasi variabel lists dgn nama dict dengan parameter variasi derajatnya
print (dict)
#output saat variabel lists dict dipanggil
```

Output :

```
user:~/catkin_ws/src/robot_control$ python dictionaries.py
[INFO] [1696054434.569709, 0.000000]: Robot Turtlebot...
[INFO] [1696054434.570881, 0.000000]: Checking Laser...
[INFO] [1696054434.606191, 6703.528000]: Checking Laser...DONE
{'P0': inf, 'P100': inf, 'P200': inf, 'P300': 2.5229811668395996, 'P400': 2.4800662994384766, 'P500': inf, 'P600': inf, 'P719': inf}
user:~/catkin_ws/src/robot_control$
```

2.2.6. Exercise 2.5

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
from robot_control_class import RobotControl

num = int(input("Select a number between 0 and 719: "))
#deklarasi input integer
rc = RobotControl()
#deklarasi variabel rc dgn fungsi dari library robotcontrol
a = rc.get_laser(num)
#deklarasi variabel a dgn fungsi dr library dan parameter num
print ("The laser value received is: ", a)
#output saat variabel a dipanggil
```

Output :

```

user:~$ ls
catkin_ws
user:~$ cd catkin_ws/
user:~/catkin_ws$ ls
build  devel  src
user:~/catkin_ws$ cd src
user:~/catkin_ws/src$ ls
CMakeLists.txt  arm_control.py  robot_control
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ ls
__pycache__  dictionaries.py  lists.py  pyscript1.py  robot_control_class.py  test_input.py  variables.py
user:~/catkin_ws/src/robot_control$ python test_input.py
Select a number between 0 and 719: 500
[INFO] [1696054796.749016, 0.000000]: Robot Turtlebot...
[INFO] [1696054796.750170, 0.000000]: Checking Laser...
[INFO] [1696054796.897725, 212.405000]: Checking Laser...DONE
The laser value received is: inf
user:~/catkin_ws/src/robot_control$

```

2.3. Conditional Statements & Loops

2.3.1. Quiz(*Maaf yang 100% lupa di terdokumentasikan)

Knowledge Check for [Conditional Statements & Loops](#) ✕

Passing score is 100%

- In Python, statements after a conditional keyword are _____
 - ☐ quoted with single quotes
 - ☐ bracketed in curly braces
 - ☐ quoted with double quotes
 - ☒ indented
- Which set of keywords are used to create conditional statements in Python?
 - ☒ if...elif...else
 - ☐ if...or...else
 - ☐ either...or...else
 - ☐ if...otherwise...else
- What happens when the condition of a while block in the middle of a Python script evaluates as False?
 - ☐ Execution of the script restarts from the top
 - ☐ It is not possible to predict what would happen
 - ☒ The next statement outside the block is executed
 - ☐ Execution of the script stops
- What is the value of the following conditional expression? "if 'Go' == 'go' or 4 > 7"
 - ☐ False
 - ☒ No
 - ☐ Gogo
 - ☐ True
- Python conditional statements evaluate to either _____
 - ☐ Stop or Continue
 - ☒ True or False
 - ☐ Yes or No
 - ☐ Yes or Yes

▲ Your score for this knowledge check: 80%

CLOSE

2.3.2. Exercise 3.1

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
```

```
from robot_control_class import RobotControl
```

```
#Refer Fungsi RobotControl dari librarynya
```

```
robotcontrol = RobotControl()
```

```
#Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya
```

```
a = robotcontrol.get_laser(360)
```

```
#Deklarasi variabel a, refer Fungsi get_laser dari librarynya
```

```
if a < 1: #looping kondisi a < 1
```

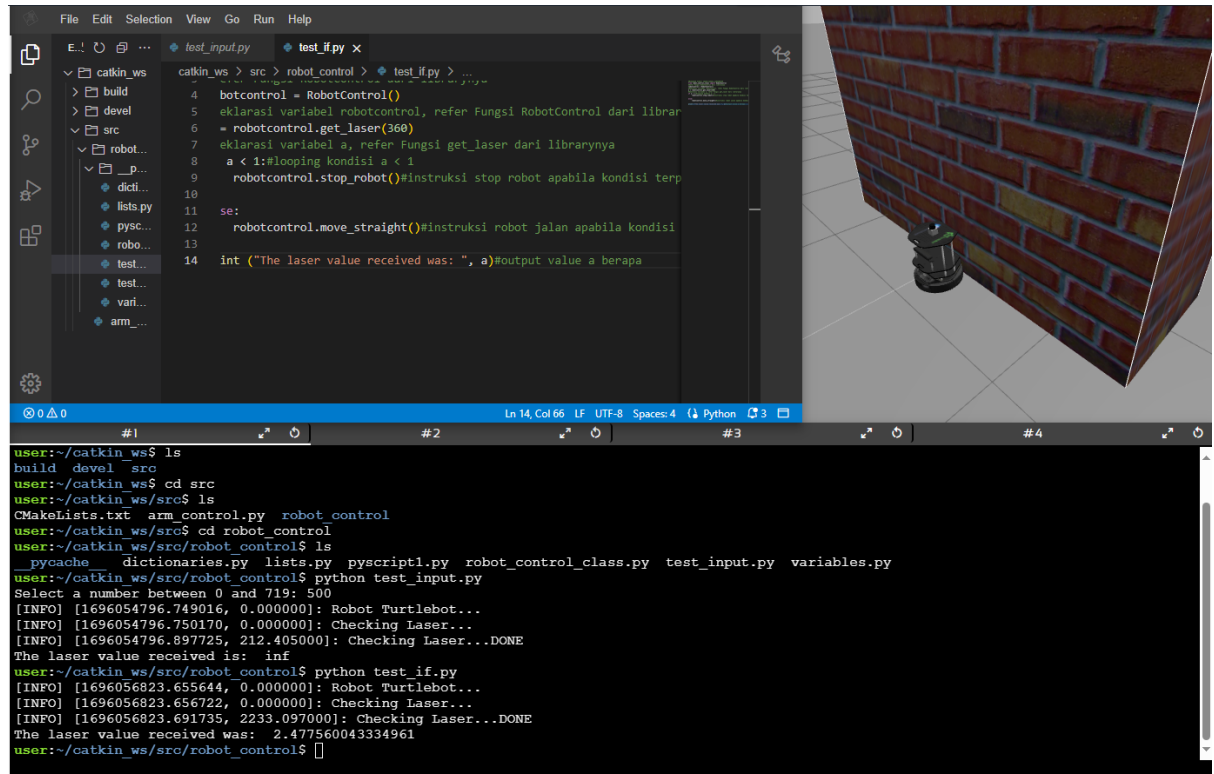
```
robotcontrol.stop_robot()#instruksi stop robot apabila kondisi terpenuhi
```

else:

```
robotcontrol.move_straight()#instruksi robot jalan apabila kondisi terpenuhi
```

```
print ("The laser value received was: ", a)#output value a berapa
```

Output :



```
File Edit Selection View Go Run Help
catkin_ws > src > robot_control > test_if.py x
4 botcontrol = RobotControl()
5 #eklarasi variabel robotcontrol, refer Fungsi RobotControl dari library
6 = robotcontrol.get_laser(360)
7 #eklarasi variabel a, refer Fungsi get_laser dari librarynya
8 a < 1:#looping kondisi a < 1
9 robotcontrol.stop_robot()#instruksi stop robot apabila kondisi terp
10
11 se:
12 robotcontrol.move_straight()#instruksi robot jalan apabila kondisi
13
14 int ("The laser value received was: ", a)#output value a berapa

user:~/catkin_ws$ ls
build devel src
user:~/catkin_ws$ cd src
user:~/catkin_ws/src$ ls
CMakeLists.txt arm_control.py robot_control
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ ls
__pycache__ dictionaries.py lists.py pyscript1.py robot_control_class.py test_input.py variables.py
user:~/catkin_ws/src/robot_control$ python test_input.py
Select a number between 0 and 719: 500
[INFO] [1696054796.749016, 0.000000]: Robot Turtlebot...
[INFO] [1696054796.750170, 0.000000]: Checking Laser...
[INFO] [1696054796.897725, 212.405000]: Checking Laser...DONE
The laser value received is: inf
user:~/catkin_ws/src/robot_control$ python test_if.py
[INFO] [1696056823.655644, 0.000000]: Robot Turtlebot...
[INFO] [1696056823.656722, 0.000000]: Checking Laser...
[INFO] [1696056823.691735, 2233.097000]: Checking Laser...DONE
The laser value received was: 2.477560043334961
user:~/catkin_ws/src/robot_control$
```

2.3.3. Exercise 3.2

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
```

```
from robot_control_class import RobotControl
```

```
#Refer Fungsi RobotControl dari librarynya
```

```
robotcontrol = RobotControl()
```

```
#Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya
```

```
a = robotcontrol.get_laser(360)
```

```
#Deklarasi variabel a, refer Fungsi get_laser dari librarynya
```

```
while a > 1:#looping kondisi a > 1
```

```
robotcontrol.move_straight()#instruksi maju lurus robot apabila kondisi terpenuhi
```

```
a = robotcontrol.get_laser(360)#instruksi deteksi dgn 360derajat robot apabila kondisi terpenuhi
```

```
print ("Current distance to wall: %f" % a)#cetak jarak robot ke dinding dgn variabel parameter a
```

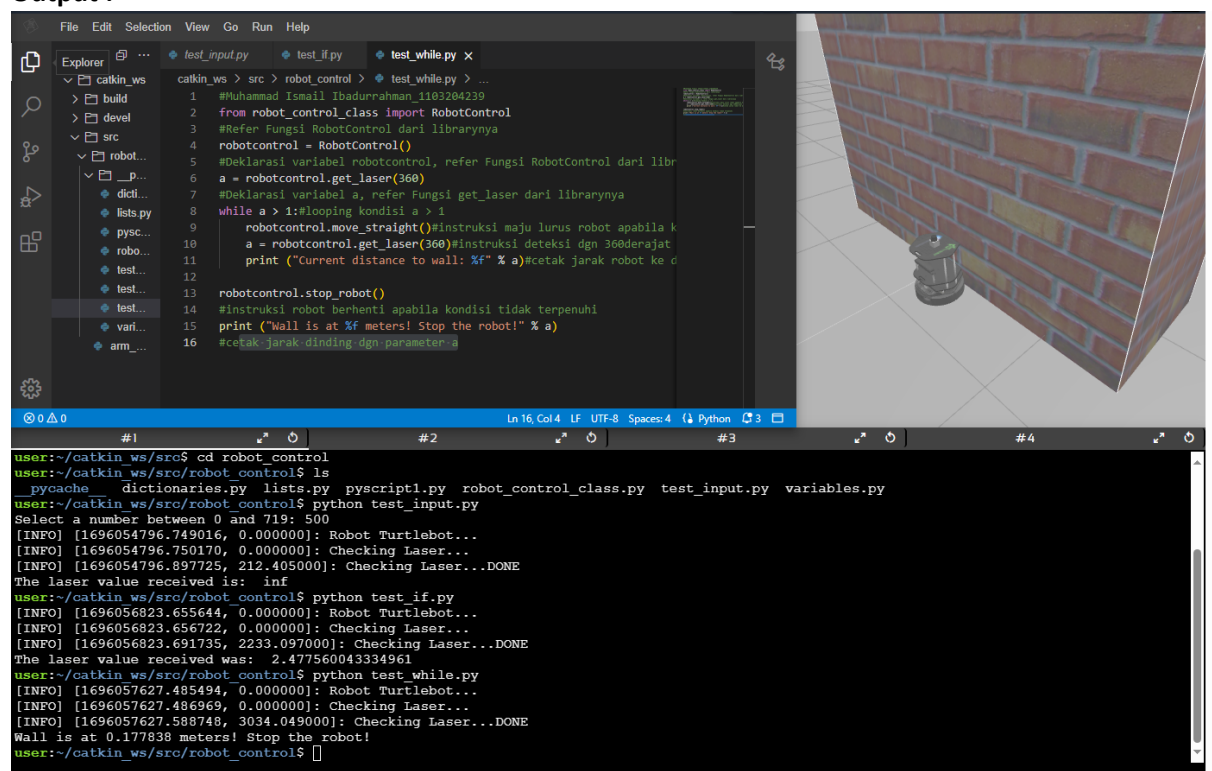
```
robotcontrol.stop_robot()
```

```
#instruksi robot berhenti apabila kondisi tidak terpenuhi
```

```
print ("Wall is at %f meters! Stop the robot!" % a)
```

```
#cetak jarak dinding dgn parameter a
```

Output :



The screenshot displays a ROS environment. On the right, a 3D simulation shows a robot (Turtlebot) in a corridor, approaching a brick wall. On the left, a code editor shows the Python script `test_while.py` with the following content:

```
1 #Muhammad Ismail Ibadurrahman_1103204239
2 from robot_control_class import RobotControl
3 #Refer Fungsi RobotControl dari librarynya
4 robotcontrol = RobotControl()
5 #Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya
6 a = robotcontrol.get_laser(360)
7 #Deklarasi variabel a, refer Fungsi get_laser dari librarynya
8 while a > 1: #looping kondisi a > 1
9     robotcontrol.move_straight() #instruksi maju lurus robot apabila k
10    a = robotcontrol.get_laser(360) #instruksi deteksi dgn 360derajat
11    print ("Current distance to wall: %f" % a) #cetak jarak robot ke d
12
13 robotcontrol.stop_robot()
14 #instruksi robot berhenti apabila kondisi tidak terpenuhi
15 print ("Wall is at %f meters! Stop the robot!" % a)
16 #cetak jarak dinding dgn parameter a
```

Below the code editor, a terminal window shows the execution of the script:

```
user:~/catkin_ws/src$ cd robot_control
user:~/catkin_ws/src/robot_control$ ls
__pycache__ dictionaries.py lists.py pyscript1.py robot_control_class.py test_input.py variables.py
user:~/catkin_ws/src/robot_control$ python test_input.py
Select a number between 0 and 719: 500
[INFO] [1696054796.749016, 0.000000]: Robot Turtlebot...
[INFO] [1696054796.750170, 0.000000]: Checking Laser...
[INFO] [1696054796.897725, 212.405000]: Checking Laser...DONE
The laser value received is: inf
user:~/catkin_ws/src/robot_control$ python test_if.py
[INFO] [1696056823.655644, 0.000000]: Robot Turtlebot...
[INFO] [1696056823.656722, 0.000000]: Checking Laser...
[INFO] [1696056823.691735, 2233.097000]: Checking Laser...DONE
The laser value received was: 2.477560043334961
user:~/catkin_ws/src/robot_control$ python test_while.py
[INFO] [1696057627.485494, 0.000000]: Robot Turtlebot...
[INFO] [1696057627.486969, 0.000000]: Checking Laser...
[INFO] [1696057627.588748, 3034.049000]: Checking Laser...DONE
Wall is at 0.177838 meters! Stop the robot!
user:~/catkin_ws/src/robot_control$
```

2.3.4. Exercise 3.3

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
```

```
from robot_control_class import RobotControl
```

```
#Refer Fungsi RobotControl dari librarynya
```

```
robotcontrol = RobotControl()
```

```
#Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya
```

```
l = robotcontrol.get_laser_full()
```

```
#Deklarasi variabel l, refer Fungsi get_laser_full dari librarynya
```

```
maxim = 0
```

```
#Deklarasi variabel maxim dengan value 0
```

```
for value in l:#pengulangan dengan iterasi bergantung pada variabel l
```

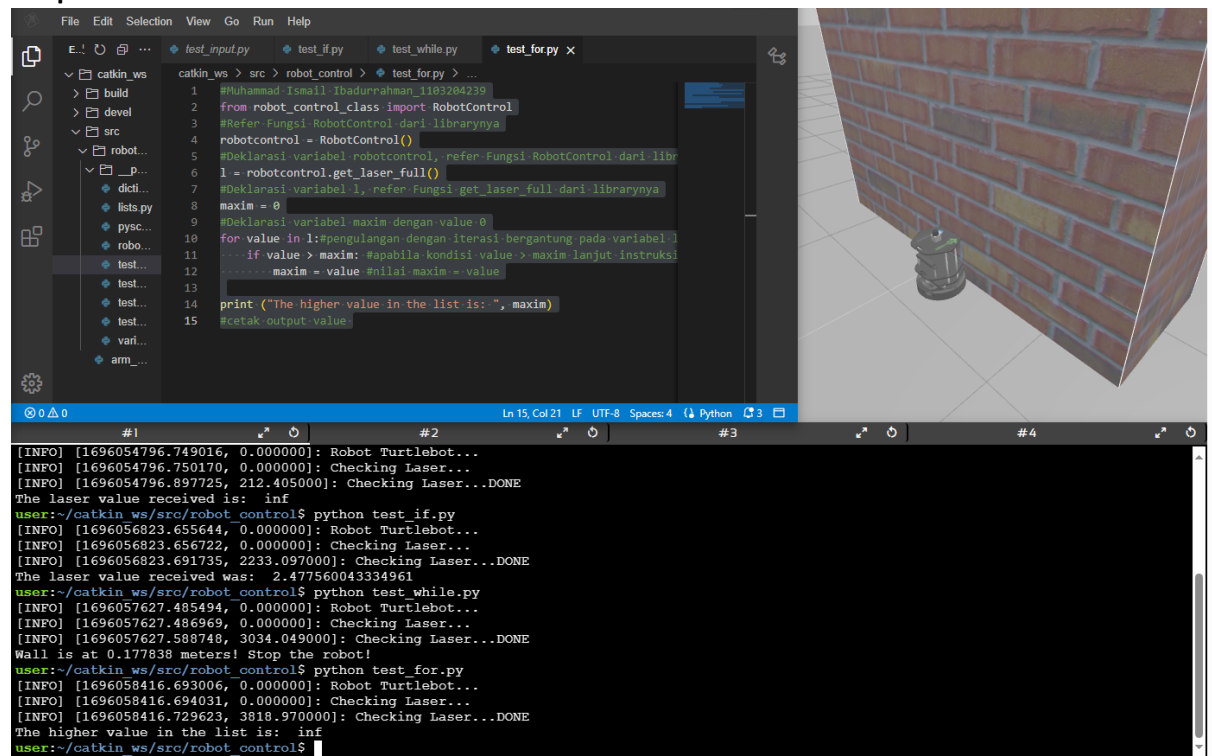
```
    if value > maxim: #apabila kondisi value > maxim lanjut instruksi
```

```
        maxim = value #nilai maxim = value
```

```
print ("The higher value in the list is: ", maxim)
```

```
#cetak output value
```

Output :



The screenshot displays a ROS2 workspace environment. On the left, a file explorer shows the directory structure of a catkin workspace. The main editor window shows a Python script named `test_for.py` with the following code:

```
1 #Muhammad-Ismail-Ibadurrahman_1183284239
2 from robot_control_class import RobotControl
3 #Refer Fungsi RobotControl dari librarynya
4 robotcontrol = RobotControl()
5 #Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari libr
6 l = robotcontrol.get_laser_full()
7 #Deklarasi variabel l, refer Fungsi get_laser_full dari librarynya
8 maxim = 0
9 #Deklarasi variabel maxim dengan value 0
10 for value in l:#pengulangan dengan iterasi bergantung pada variabel l
11     if value > maxim: #apabila kondisi value > maxim lanjut instruksi
12         maxim = value #nilai maxim = value
13
14 print ("The higher value in the list is: ", maxim)
15 #cetak output value
```

On the right, a 3D visualization shows a robot (Turtlebot) in a simulated environment, positioned near a brick wall. The bottom terminal window shows the execution output:

```
[INFO] [1696054796.749016, 0.000000]: Robot Turtlebot...
[INFO] [1696054796.750170, 0.000000]: Checking Laser...
[INFO] [1696054796.897725, 212.405000]: Checking Laser...DONE
The laser value received is: inf
user:~/catkin_ws/src/robot_control$ python test_if.py
[INFO] [1696056823.655644, 0.000000]: Robot Turtlebot...
[INFO] [1696056823.656722, 0.000000]: Checking Laser...
[INFO] [1696056823.691735, 2233.097000]: Checking Laser...DONE
The laser value received was: 2.477560043334961
user:~/catkin_ws/src/robot_control$ python test_while.py
[INFO] [1696057627.485494, 0.000000]: Robot Turtlebot...
[INFO] [1696057627.486969, 0.000000]: Checking Laser...
[INFO] [1696057627.588748, 3034.049000]: Checking Laser...DONE
Wall is at 0.177838 meters! Stop the robot!
user:~/catkin_ws/src/robot_control$ python test_for.py
[INFO] [1696058416.693006, 0.000000]: Robot Turtlebot...
[INFO] [1696058416.694031, 0.000000]: Checking Laser...
[INFO] [1696058416.729623, 3818.970000]: Checking Laser...DONE
The higher value in the list is: inf
user:~/catkin_ws/src/robot_control$
```

2.4. Methods

2.4.1. Quiz

Knowledge Check for **Methods**

Passing score is 100%

1. How many parameters can a method take?

☐ at most five
☒ zero or more
☐ at least one
☐ one or more

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

☐ loop
☐ class
☐ variable
☒ method

3. How should you call a method "go_home"?

☒ go_home()
☐ call go_home()
☐ go_home[]
☐ ! go_home()

4. Which of the following is true about variables defined in a method?

☐ they are only available in a script that calls the method
☐ they are available everywhere even in other scripts
☐ they are available for a brief moment after the method is called
☒ they cannot be used outside the method

5. A method is defined as "get_value(a, b, c=3)". Which of the following calls to the function is wrong?

☐ get_value(4, 5, 3)
☒ get_value[3]
☐ get_value(5, 4)

☒ Your score for this knowledge check: **100%**

CLOSE

2.4.2. Exercise 4.1

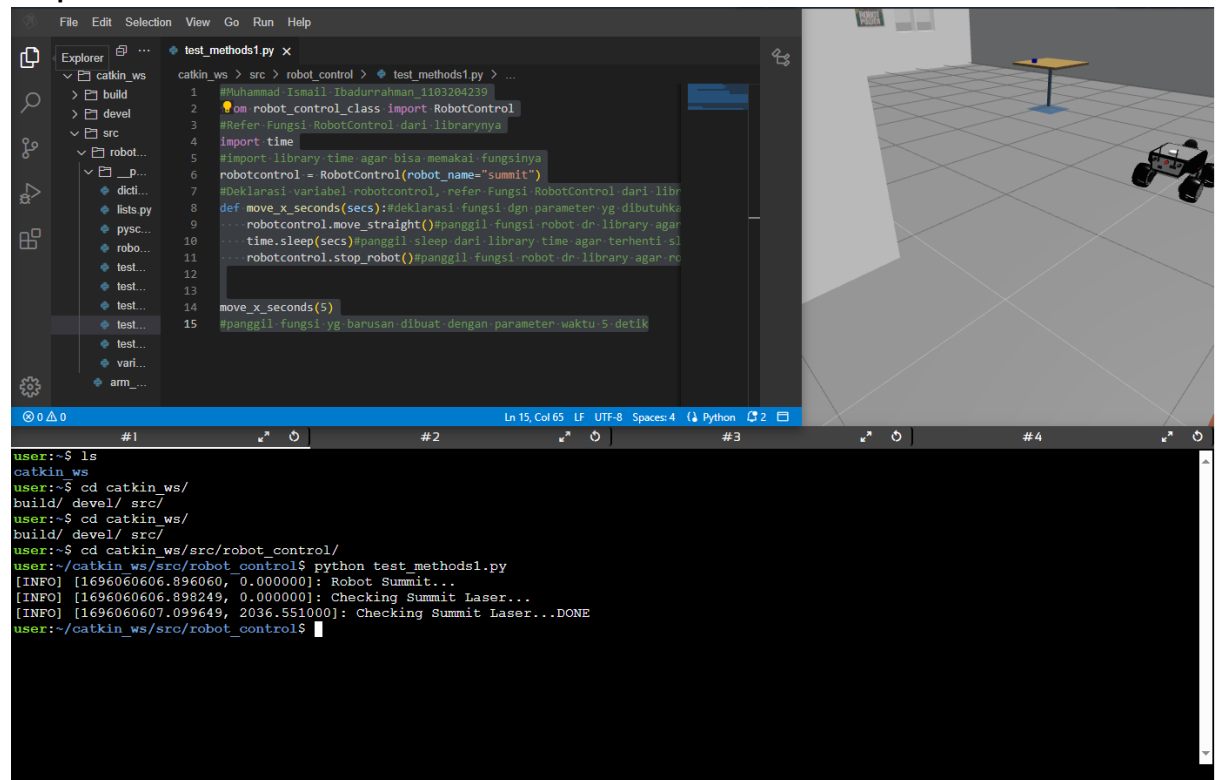
Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
from robot_control_class import RobotControl
#Refer Fungsi RobotControl dari librarynya
import time
#import library time agar bisa memakai fungsinya
robotcontrol = RobotControl(robot_name="summit")
#Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya dan memberi
nama robotnya summit
def move_x_seconds(secs):#deklarasi fungsi dgn parameter yg dibutuhkan detik
    robotcontrol.move_straight()#panggil fungsi robot dr library agar bergerak lurus
    time.sleep(secs)#panggil sleep dari library time agar terhenti slm waktu yg ditentukan
    robotcontrol.stop_robot()#panggil fungsi robot dr library agar robot stop

move_x_seconds(5)
```

#panggil fungsi yg barusan dibuat dengan parameter waktu 5 detik

Output :



The screenshot displays a ROS2 workspace environment. The top window shows a Python script named `test_methods1.py` in the `robot_control` package. The script imports `RobotControl` from `robot_control_class` and `time` from the `time` module. It initializes a `RobotControl` object with the name "summit". A function `move_x_seconds(secs)` is defined, which calls `robotcontrol.move_straight()`, sleeps for the specified seconds, and then calls `robotcontrol.stop_robot()`. The function is called with `move_x_seconds(5)`. The bottom window shows the terminal output of running `python test_methods1.py`, which includes status messages for the robot and laser checking.

```
catkin_ws > src > robot_control > test_methods1.py > ...
1 #Muhammad Ismail Ibadurrahman_1103204239
2 from robot_control_class import RobotControl
3 #Refer Fungsi RobotControl dari librarynya
4 import time
5 #import library time agar bisa memakai fungsinya
6 robotcontrol = RobotControl(robot_name="summit")
7 #Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya
8 def move_x_seconds(secs):#deklarasi fungsi dgn parameter yg dibutuhkan
9     robotcontrol.move_straight()#panggil fungsi robot dr library agar
10     time.sleep(secs)#panggil sleep dari library time agar terhenti sl
11     robotcontrol.stop_robot()#panggil fungsi robot dr library agar ro
12
13
14 move_x_seconds(5)
15 #panggil fungsi yg barusan dibuat dengan parameter waktu 5 detik

Ln 15, Col 65  LF  UTF-8  Spaces: 4  Python  2  [Python icon]
```

```
user:~$ ls
catkin_ws
user:~$ cd catkin_ws/
build/ devel/ src/
user:~$ cd catkin_ws/
build/ devel/ src/
user:~$ cd catkin_ws/src/robot_control/
user:~/catkin_ws/src/robot_control$ python test_methods1.py
[INFO] [1696060606.896060, 0.000000]: Robot Summit...
[INFO] [1696060606.898249, 0.000000]: Checking Summit Laser...
[INFO] [1696060607.099649, 2036.551000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$
```

2.4.3. Exercise 4.2

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
from robot_control_class import RobotControl
#Refer Fungsi RobotControl dari librarynya
robotcontrol = RobotControl(robot_name="summit")
#Deklarasi variabel robotcontrol, refer Fungsi RobotControl dari librarynya dan memberi
nama robotnya summit
def get_laser_values(a,b,c):#deklarasi fungsi dgn parameter yg dibutuhkan a,b,c
    r1 = robotcontrol.get_laser_summit(a)#Deklarasi variabel r1, refer fungsi
get_laser_control dari lib RobotControl dan memberi nama robotnya summit
    r2 = robotcontrol.get_laser_summit(b)#Deklarasi variabel r2, refer Fungsi
get_laser_control dari lib RobotControl dan memberi nama robotnya summit
    r3 = robotcontrol.get_laser_summit(c)#Deklarasi variabel r3, refer Fungsi
get_laser_control dari lib RobotControl dan memberi nama robotnya summit

    return [r1, r2, r3]#mengembalikan value r1, r2, r3
```

```
l = get_laser_values(0, 500, 1000)
```

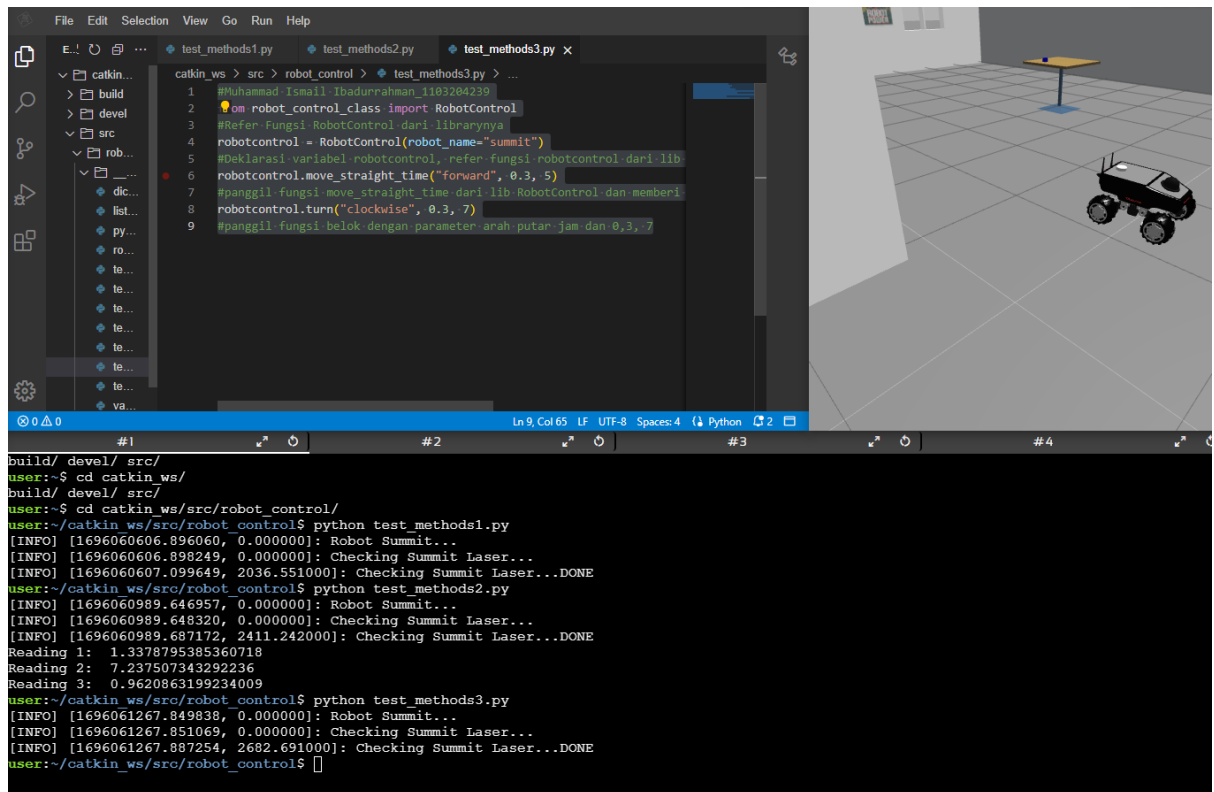
#Deklarasi variabel l dengan value fungsi `get_laser_values` yg dibuat barusan dengan parameter 0, 500, 1000

```
print ("Reading 1: ", l[0])#cetak value l dari data bergantung parameter 1
```

```
print ("Reading 2: ", l[1])#cetak value l dari data bergantung parameter 2
```

```
print ("Reading 3: ", l[2])#cetak value l dari data bergantung parameter 3
```

Output :



2.4.5. Exercise 4.4

Source Code :

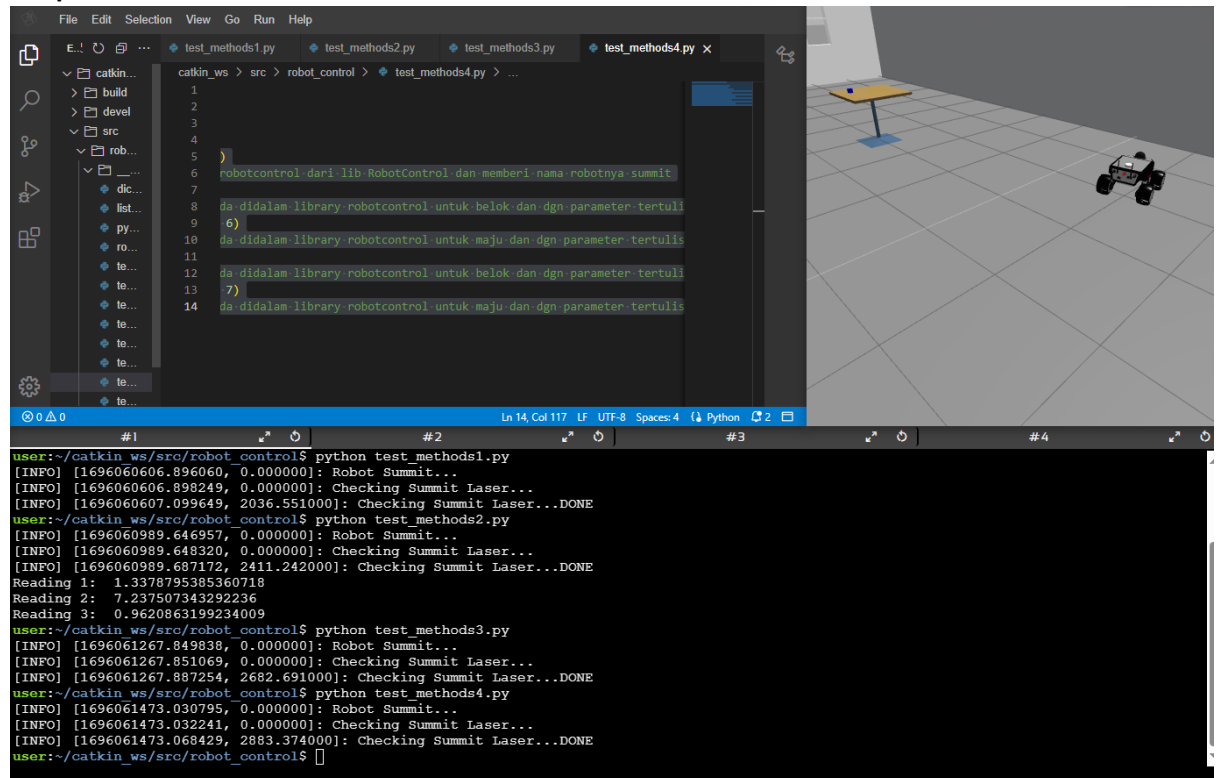
#Muhammad Ismail Ibadurrahman_1103204239

```

from robot_control_class import RobotControl
#Refer Fungsi RobotControl dari librarynya
robotcontrol = RobotControl(robot_name="summit")
#Deklarasi variabel robotcontrol, refer fungsi robotcontrol dari lib RobotControl dan
memberi nama robotnya summit
robotcontrol.turn("counter-clockwise", 0.3, 4)
#memberi perintah pada robot dari fungsi yang ada didalam library robotcontrol untuk
belok dan dgn parameter tertulis
robotcontrol.move_straight_time("forward", 0.3, 6)
#memberi perintah pada robot dari fungsi yang ada didalam library robotcontrol untuk
maju dan dgn parameter tertulis
robotcontrol.turn("counter-clockwise", 0.3, 4)
#memberi perintah pada robot dari fungsi yang ada didalam library robotcontrol untuk
belok dan dgn parameter tertulis
robotcontrol.move_straight_time("forward", 0.3, 7)
#memberi perintah pada robot dari fungsi yang ada didalam library robotcontrol untuk
maju dan dgn parameter tertulis

```

Output :



The screenshot displays a ROS2 workspace environment. On the left, a file explorer shows the directory structure: `catkin_ws` > `src` > `robot_control` > `test_methods4.py`. The code editor shows the following Python code in `test_methods4.py`:

```
1
2
3
4
5
6 robotcontrol dari lib RobotControl dan memberi nama robotnya summit
7
8 da didalam library robotcontrol untuk belok dan dgn parameter tertulis
9 6)
10 da didalam library robotcontrol untuk maju dan dgn parameter tertulis
11
12 da didalam library robotcontrol untuk belok dan dgn parameter tertulis
13 7)
14 da didalam library robotcontrol untuk maju dan dgn parameter tertulis
```

On the right, a 3D simulation shows a small robot (Summit) in a room with a table and a chair. The bottom terminal window shows the execution of four Python scripts:

```
user:~/catkin_ws/src/robot_control$ python test_methods1.py
[INFO] [1696060606.896060, 0.000000]: Robot Summit...
[INFO] [1696060606.898249, 0.000000]: Checking Summit Laser...
[INFO] [1696060607.099649, 2036.551000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python test_methods2.py
[INFO] [1696060989.646957, 0.000000]: Robot Summit...
[INFO] [1696060989.648320, 0.000000]: Checking Summit Laser...
[INFO] [1696060989.687172, 2411.242000]: Checking Summit Laser...DONE
Reading 1: 1.3378795385360718
Reading 2: 7.237507343292236
Reading 3: 0.9620863199234009
user:~/catkin_ws/src/robot_control$ python test_methods3.py
[INFO] [1696061267.849838, 0.000000]: Robot Summit...
[INFO] [1696061267.851069, 0.000000]: Checking Summit Laser...
[INFO] [1696061267.887254, 2682.691000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python test_methods4.py
[INFO] [1696061473.030795, 0.000000]: Robot Summit...
[INFO] [1696061473.032241, 0.000000]: Checking Summit Laser...
[INFO] [1696061473.068429, 2883.374000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$
```

2.5. Python Classes & OOP

2.5.1. Quiz

Knowledge Check for **Python Classes & OOP**

Passing score is 100%

- You want to define a Python class called "Manager", what are the first two words you should write?
☐ Manager class
☐ define Manager
☒ class Manager
☐ def Manager
- The methods of the class can modify any variable of the class.
☒ True
☐ False
- In OOP, objects are composed of what and what?
☐ classes and methods
☐ attributes and classes
☒ attributes and methods
☐ methods and functions
- When defining variables and methods of the class that will be used by instances of the class, what keyword must be used?
☐ global
☒ self
☐ class
☐ this
- Before you can use a class defined in another script in your code, what is the first thing you have to do?
☐ call one of its methods
☐ export it
☒ import it
☐ instantiate it

✓ Your score for this knowledge check: **100%**

CLOSE

2.5.2. Exercise 5.1

Source Code :

```
#Muhammad Ismail Ibadurrahman_1103204239
from robot_control_class import RobotControl
#Refer Fungsi RobotControl dari librarynya
class MoveRobot:
```

```
def __init__(self, motion, clockwise, speed, time): #deklarasi variabel class inisiasi dgn parameter seperti yg tertulis
```

```
    self.robotcontrol = RobotControl(robot_name="summit")#deklarasi variabel robotcontrol dengan value robotcontrol fungsi dari library RobotControl agar robot diberi nama jd summit
```

```
    self.motion = motion#deklarasi variabel motion dengan value motion fungsi dari library RobotControl
```

```
    self.clockwise = clockwise#deklarasi variabel clockwise dengan value clockwise fungsi dari library RobotControl
```

```
    self.speed = speed#deklarasi variabel speed dengan value speed fungsi dari library RobotControl
```

```
    self.time = time#deklarasi variabel time dengan value time fungsi dari library RobotControl
```

```
    self.time_turn = 7.0 # This is an estimate time in which the robot will rotate 90 degrees
```

```
def do_square(self):#deklarasi fungsi do_square
```

```
    i = 0#deklarasi i untuk iterasi dengan nilai 0
```

```
    while (i < 4):#deklarasi pengulangan dengan kondisi i<4
```

```
        self.move_straight()#pemanggilan fungsi maju dari class yang sudah dibuat
```

```
        self.turn()#pemanggilan fungsi belok dari class yang sudah dibuat
```

```
        i+=1#iterasi +1
```

```
def move_straight(self):#deklarasi fungsi maju
```

```
    self.robotcontrol.move_straight_time(self.motion, self.speed, self.time)#deklarasi variabel maju dengan parameter tertentu agar robot berjalan sesuai keinginan
```

```
def turn(self):#deklarasi fungsi belok
```

```
    self.robotcontrol.turn(self.clockwise, self.speed, self.time_turn)#deklarasi variabel belok dengan parameter tertentu agar robot berjalan sesuai keinginan
```

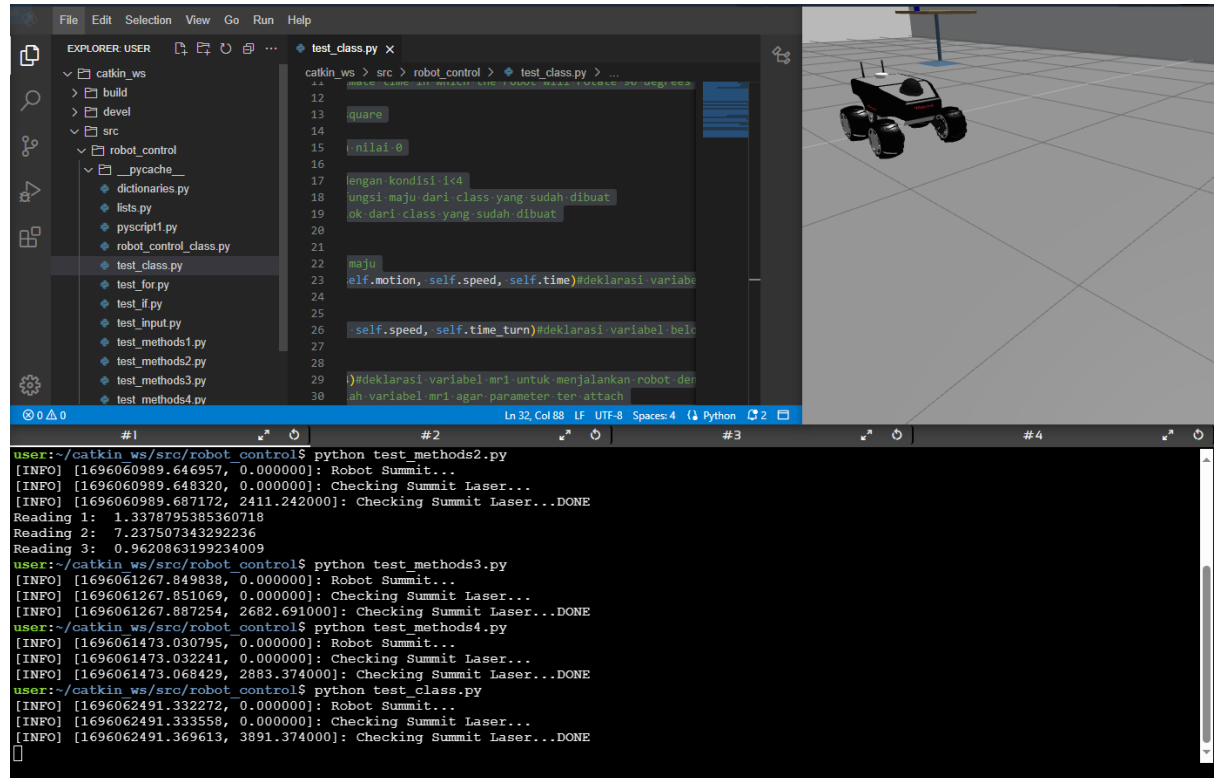
```
mr1 = MoveRobot('forward', 'clockwise', 0.3, 4)#deklarasi variabel mr1 untuk menjalankan robot dengan parameter tertentu
```

```
mr1.do_square()#panggil fungsi do_square setelah variabel mr1 agar parameter ter attach
```

```
mr2 = MoveRobot('forward', 'clockwise', 0.3, 8)#deklarasi variabel mr2 untuk menjalankan robot dengan parameter tertentu
```

```
mr2.do_square()#panggil fungsi do_square setelah variabel mr2 agar parameter ter attach
```

Output :



The screenshot displays a ROS2 development environment. The top section features a code editor with the file explorer on the left showing the project structure: `catkin_ws` containing `build`, `devel`, `src`, and `robot_control`. The `robot_control` package includes `__pycache__`, `dictionaries.py`, `lists.py`, `pyscript1.py`, `robot_control_class.py`, `test_class.py`, `test_for.py`, `test_if.py`, `test_input.py`, `test_methods1.py`, `test_methods2.py`, `test_methods3.py`, and `test_methods4.py`. The main editor window shows the content of `test_class.py`, which includes comments in Indonesian and Python code for a class method. The code defines a `square` variable, sets `nilai=0`, and includes logic for a robot's condition and movement, such as `self.motion, self.speed, self.time` and `self.speed, self.time_turn`. A 3D simulation of a robot is visible in the top right corner. The bottom section is a terminal window with four tabs. The active tab shows the execution of `python test_methods2.py`, followed by `python test_methods3.py`, `python test_methods4.py`, and `python test_class.py`. Each execution outputs information about the robot's summit and laser status, including timestamps and readings.

```
catkin_ws > src > robot_control > test_class.py > ...
11
12
13 square
14
15 nilai=0
16
17 lengan_kondisi<4
18 fungsi_maju_dari_class_yang_sudah_dibuat
19 ok_dari_class_yang_sudah_dibuat
20
21
22 maju
23 self.motion, self.speed, self.time)#deklarasi variabel
24
25
26 self.speed, self.time_turn)#deklarasi variabel belok
27
28
29 )#deklarasi variabel mr1 untuk menjalankan robot dan
30 ah variabel mr1 agar parameter ter attach
```

```
user:~/catkin_ws/src/robot_control$ python test_methods2.py
[INFO] [1696060989.646957, 0.000000]: Robot Summit...
[INFO] [1696060989.648320, 0.000000]: Checking Summit Laser...
[INFO] [1696060989.687172, 2411.242000]: Checking Summit Laser...DONE
Reading 1: 1.3378795385360718
Reading 2: 7.237507343292236
Reading 3: 0.9620863199234009
user:~/catkin_ws/src/robot_control$ python test_methods3.py
[INFO] [1696061267.849838, 0.000000]: Robot Summit...
[INFO] [1696061267.851069, 0.000000]: Checking Summit Laser...
[INFO] [1696061267.887254, 2682.691000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python test_methods4.py
[INFO] [1696061473.030795, 0.000000]: Robot Summit...
[INFO] [1696061473.032241, 0.000000]: Checking Summit Laser...
[INFO] [1696061473.068429, 2883.374000]: Checking Summit Laser...DONE
user:~/catkin_ws/src/robot_control$ python test_class.py
[INFO] [1696062491.332272, 0.000000]: Robot Summit...
[INFO] [1696062491.333558, 0.000000]: Checking Summit Laser...
[INFO] [1696062491.369613, 3891.374000]: Checking Summit Laser...DONE
```