app.theconstructsim.com/Desktop

```
>>> def f():
...     lv = "Local Variable"
...
>>> f()
>>> print(lv)
Traceback (most recent call last):
  File "", line 1, in
NameError: name 'lv' is not defin
>>>
```

- End Expected Output -

As you can see, we are getting an error. Specifically, it says: **"name 'lv' is not defined"**. As we have previously said, the variable **lv** has been defined inside the method, so it is local to the method. This means that it CANNOT be used outside the method.

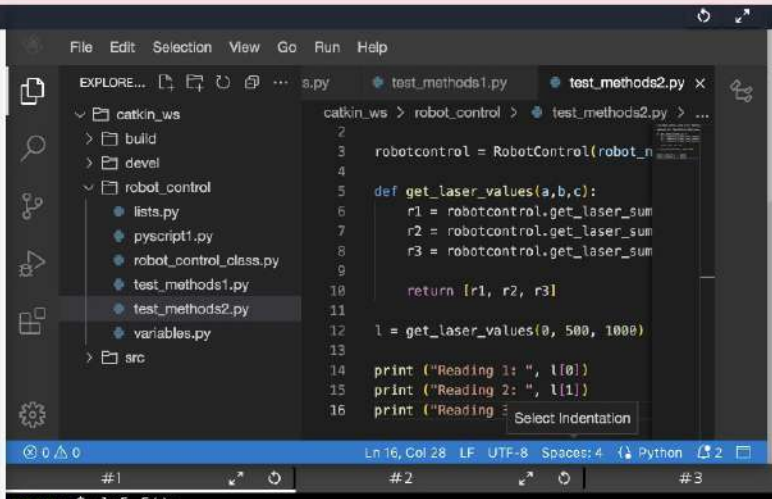
For a case like this, Python provides **global variables**. Let's check out an example:

In [ ]:

```
def f():
    global gv
    gv = "Global Variable"

gv = "Empty"
print (gv)
f()
print(gv)
```

As you can see, now we have defined the variable gv before the method is called. Then, inside the



#1 #2 #3 #4

```
user:~$ def f():
bash: syntax error near unexpected token `('
user:~$     global gv
bash: global: command not found
user:~$     gv = "Global Variable"
bash: gv: command not found
user:~$
user:~$ gv = "Empty"
bash: gv: command not found
user:~$ print (gv)
bash: syntax error near unexpected token `gv'
user:~$ f()
> print(gv)from robot_control_class import RobotControl
bash: syntax error near unexpected token `print'
user:~$
```

app.theconstructsim.com/Desktop

Update

- Expected Output -

```
>>> def add(a=2,b=2):
...     res = a + b
...     return res
...
>>> r = add(3,4)
>>> print (r)
7
>>>
```

- End Expected Output -

A method can return exactly one value, or we should say, one object. An object can be a numerical value, like an integer or a float, but it can also be a list or a dictionary (Remember the **get_laser_full()** method). So, if we have to return, for example, three integer values, we can return a list or a tuple with these three integer values.

In [ ]:

```
def return_list():

    return [1, 2, 3]

l = return_list()
print (l[0])
print (l[1])
print (l[2])
```

- Expected Output -

File Edit Selection View Go Run Help

EXPLORE... variables.py lists.py test_methods1.py

catkin_ws
- build
- devel
- robot_control
  - lists.py
  - pyscript1.py
  - robot_control_class.py
  - test_methods1.py
  - variables.py
- src

catkin_ws > robot_control > test_methods1.py > ...

```
1  from robot_control_class import Rob
2  import time
3
4  robotcontrol = RobotControl(robot_n
5
6  def move_x_seconds(secs):
7      robotcontrol.move_straight()
8      time.sleep(secs)
9      robotcontrol.stop_robot()
10
11
12 move_x_seconds(5)
```

No Problems

⊗ 0 ⚠ 0 Ln 12, Col 18 LF UTF-8 Spaces: 4 Python 2

#1 #2 #3 #4

```
bash: syntax error near unexpected token `('
user:~$
user:~$     return [1, 2, 3]
bash: return: [1,: numeric argument required
bash: return: can only `return' from a function or sourced script
user:~$
user:~$ l = return_list()
bash: syntax error near unexpected token `('
user:~$ print (l[0])
bash: syntax error near unexpected token `l[0]'
user:~$ print (l[1])
bash: syntax error near unexpected token `l[1]'
user:~$ print (l[2])
bash: syntax error near unexpected token `l[2]'
user:~$
```

app.theconstructsim.com/Desktop

So, when called, this method will return a **LIST** containing all the 720 different readings from the laser beams.

- 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:  2.4758487739691123
Position 719:  inf
(.py3venv) user:~/catkin_ws/src/robot_control$
```

- End Expected Output -

- Solution for Exercise 2.3 -

The Construct

File Edit Selection View Go Run Help

EXPLORE...

- catkin_ws
  - build
  - devel
  - robot_control
    - pyscript1.py
    - robot_control_class.py
    - variables.py
  - src

pyscript1.py | variables.py

catkin_ws > robot_control > variables.py > ...

```
1  trol_class import RobotControl
2
3   RobotControl()
4
5  control.get_laser(0)
6  ser value received is: ", laser1)
7
8  control.get_laser(360)
9  ser value received is: ", laser2)
10
11 control.get_laser(719)
12 ser value received is: ", laser2)
```

Ln 12, Col 48 LF UTF-8 Spaces: 4 Python 2

#1 #2 #3 #4

```
user:~$
user:~$ print (t)
bash: syntax error near unexpected token `t'
user:~$ dict = {"Jon": 25, "Daenerys": 22, "Cersei": 31, "Night King": 35}
bash: dict: command not found
user:~$
user:~$ print (dict["Jon"])
bash: syntax error near unexpected token `dict["Jon"]'
user:~$ print (dict["Night King"])
bash: syntax error near unexpected token `dict["Night King"]'
user:~$ dict = {"Jon": 25, "Daenerys": 22, "Cersei": 31, "Night King": 35}
bash: dict: command not found
user:~$
user:~$ dict["Jon"] = 10
bash: dict[Jon]: command not found
```

2 - Python Essentials

Python 3 for Robotics

100%

app.theconstructsim.com/Desktop

Update

END Python File: jedi_class.py

The code above has two parts:

1. Where we define the class
2. Where we use the definition to create and use variables of type *Jedi* (*j1* and *j2*)

5.3.1 Definition of a class

Let's quickly analyze the above class. The first thing that we see is the tag *class* and the name *Jedi*.

```
In [ ]:
class Jedi:
```

This means, let's define a class named *Jedi*. Then, everything that goes beneath that line and that contains an indentation belongs to the code of that class.

Let's see the different parts that a class can contain.

5.3.2 methods of the class

Then, we continue with the tag *def*. As you know from the previous chapter, that means that what follows is the definition of a method. However, in this case, the method is just defined for the class

File Edit Selection View Go Run Help

EXPLORE... hods3.py test_methods4.py test_class.py

catkin_ws
- build
- devel
- robot_control
  - lists.py
  - pyscript1.py
  - robot_control_class.py
  - test_class.py
  - test_methods1.py
  - test_methods2.py
  - test_methods3.py
  - test_methods4.py
  - variables.py
- src

catkin_ws > robot_control > test_class.py > ...

```
17                  self.move_straight()
18                  self.turn()
19                  i+=1
20
21      def move_straight(self):
22          self.robotcontrol.move_stra
23
24      def turn(self):
25          self.robotcontrol.turn(self
26
27
28  mr1 = MoveRobot('forward', 'clockwi
29  mr1.do_square()
30  mr2 = MoveRobot('forward', 'clockwi
31  mr2.do_square()
```

⊗ 0 ⚠ 0 Ln 31, Col 16 LF UTF-8 Spaces: 4 Python 2

#1 #2 #3 #4

```
user:~$ my_object = Jedi("Qui-Gong-Jin")
bash: syntax error near unexpected token `('
user:~$ rc = RobotControl()
bash: syntax error near unexpected token `('
user:~$ class Jedi:
bash: class: command not found
user:~$ def __init__(self,name):
bash: syntax error near unexpected token `('
user:~$ def say_hi(self):
bash: syntax error near unexpected token `('
user:~$ say_hi()
> j = Jedi()
bash: syntax error near unexpected token `j'
user:~$ j.say_hi()
>
```

5 - Python Classes & OOP

Python 3 for Robotics

100%

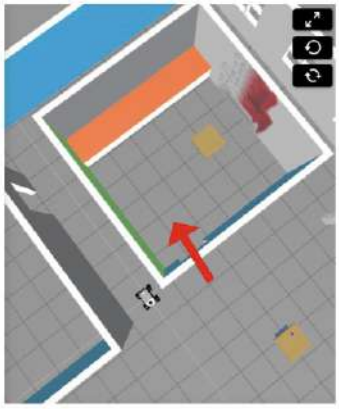
app.theconstructsim.com/Desktop

Update

Note that in this last exercise, we have introduced a new concept, **Python classes**. And this is exactly 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 -

- Solution for Exercise 4.4 -

File Edit Selection View Go Run Help

EXPLORE...

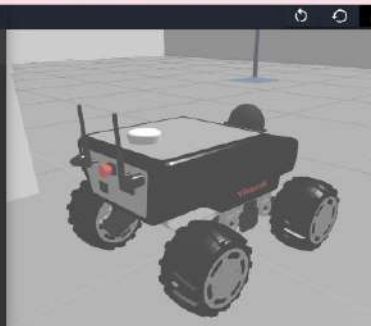
catkin_ws
- build
- devel
- robot_control
  - lists.py
  - pyscript1.py
  - robot_control_class.py
  - test_methods1.py
  - test_methods2.py
  - test_methods3.py
  - test_methods4.py
  - variables.py
- src

2.py | test_methods3.py | test_methods4.py

catkin_ws > robot_control > test_methods4.py > ...

```
1  l_class import RobotControl
2
3  botControl(robot_name="summit")
4
5  ("counter-clockwise", 0.3, 4)
6  _straight_time("forward", 0.3, 6)
7  ("counter-clockwise", 0.3, 4)
8  _straight_time("forward", 0.3, 7)
```

Ln 8, Col 51 LF UTF-8 Spaces: 4 Python 2



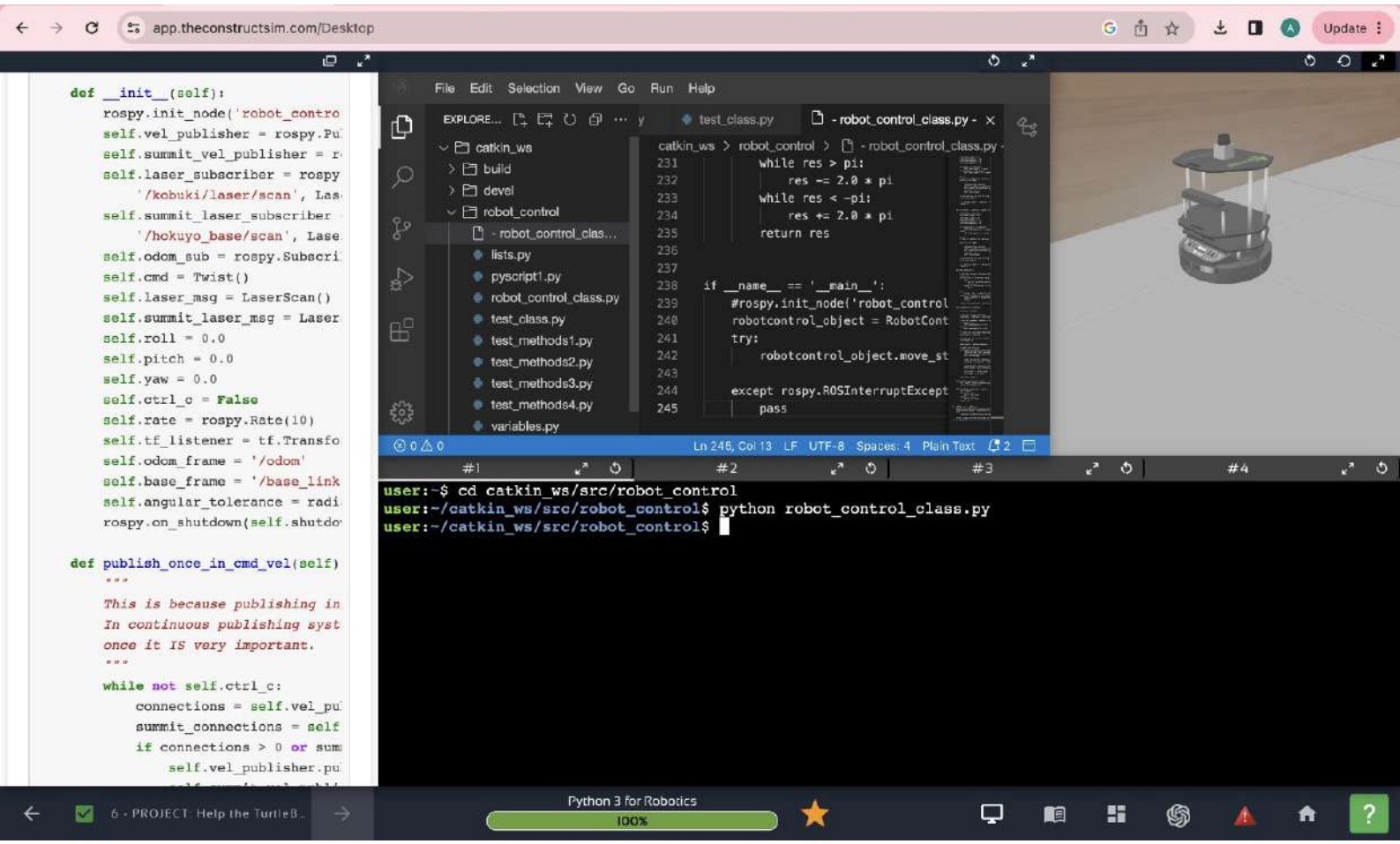
#1 #2 #3 #4

```
user:~$
```

4 - Methods

Python 3 for Robotics

100%


app.theconstructsim.com/Desktop
Update
def __init__(self):
rospy.init_node('robot_contro
self.vel_publisher = rospy.Pu
self.summit_vel_publisher = r
self.laser_subscriber = rospy
'/kobuki/laser/scan', Las
self.summit_laser_subscriber
'/hokuyo_base/scan', Lase
self.odom_sub = rospy.Subscri
self.cmd = Twist()
self.laser_msg = LaserScan()
self.summit_laser_msg = Laser
self.roll = 0.0
self.pitch = 0.0
self.yaw = 0.0
self.ctrl_c = False
self.rate = rospy.Rate(10)
self.tf_listener = tf.Transfo
self.odom_frame = '/odom'
self.base_frame = '/base_link
self.angular_tolerance = radi
rospy.on_shutdown(self.shutdo
def publish_once_in_cmd_vel(self)
"""
This is because publishing in
In continuous publishing syst
once it IS very important.
"""
while not self.ctrl_c:
connections = self.vel_pu
summit_connections = self
if connections > 0 or sum
self.vel_publisher.pu
File Edit Selection View Go Run Help
EXPLORE...
catkin_ws
build
devel
robot_control
- robot_control_clas...
lists.py
pyscript1.py
robot_control_class.py
test_class.py
test_methods1.py
test_methods2.py
test_methods3.py
test_methods4.py
variables.py
test_class.py
- robot_control_class.py -
catkin_ws > robot_control > - robot_control_class.py
231 while res > pi:
232 res -= 2.0 * pi
233 while res < -pi:
234 res += 2.0 * pi
235 return res
236
237
238 if __name__ == '__main__':
239 #rospy.init_node('robot_control
240 robotcontrol_object = RobotCont
241 try:
242 robotcontrol_object.move_st
243
244 except rospy.ROSInterruptExcept
245 pass
Ln 246, Col 13 LF UTF-8 Spaces: 4 Plain Text
#1
#2
#3
#4
user:~$ cd catkin_ws/src/robot_control
user:~/catkin_ws/src/robot_control$ python robot_control_class.py
user:~/catkin_ws/src/robot_control$
6 - PROJECT: Help the TurtleB...
Python 3 for Robotics
100%

app.theconstructsim.com/Desktop

Update

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 Robot
```

- You will also have to create an object of the class in order to be able to call the methods. You can do this with the following line:

In [ ]:

```
robotcontrol = RobotControl(robot_nam
```

- Now, you will be able to call the methods from this object, like this:

In [ ]:

```
robotcontrol.move_straight_time(neces
```

- End of Exercise 4.3 -

- End Solution for Exercise 4.3 -

File Edit Selection View Go Run Help

EXPLORE...

catkin_ws
- build
- devel
- robot_control
  - lists.py
  - pyscript1.py
  - robot_control_class.py
  - test_methods1.py
  - test_methods2.py
  - test_methods3.py
  - variables.py
- src

1.py | test_methods2.py | test_method | Outline

catkin_ws > robot_control > test_methods3.py > ...

```
1  robot_control_class import RobotCon
2
3  control = RobotControl(robot_name="
4
5  control.move_straight_time("forward
6  control.turn("clockwise", 0.3, 7)
```

⊗ 0 △ 0 Ln 6, Col 39 LF UTF-8 Spaces: 4 Python 2

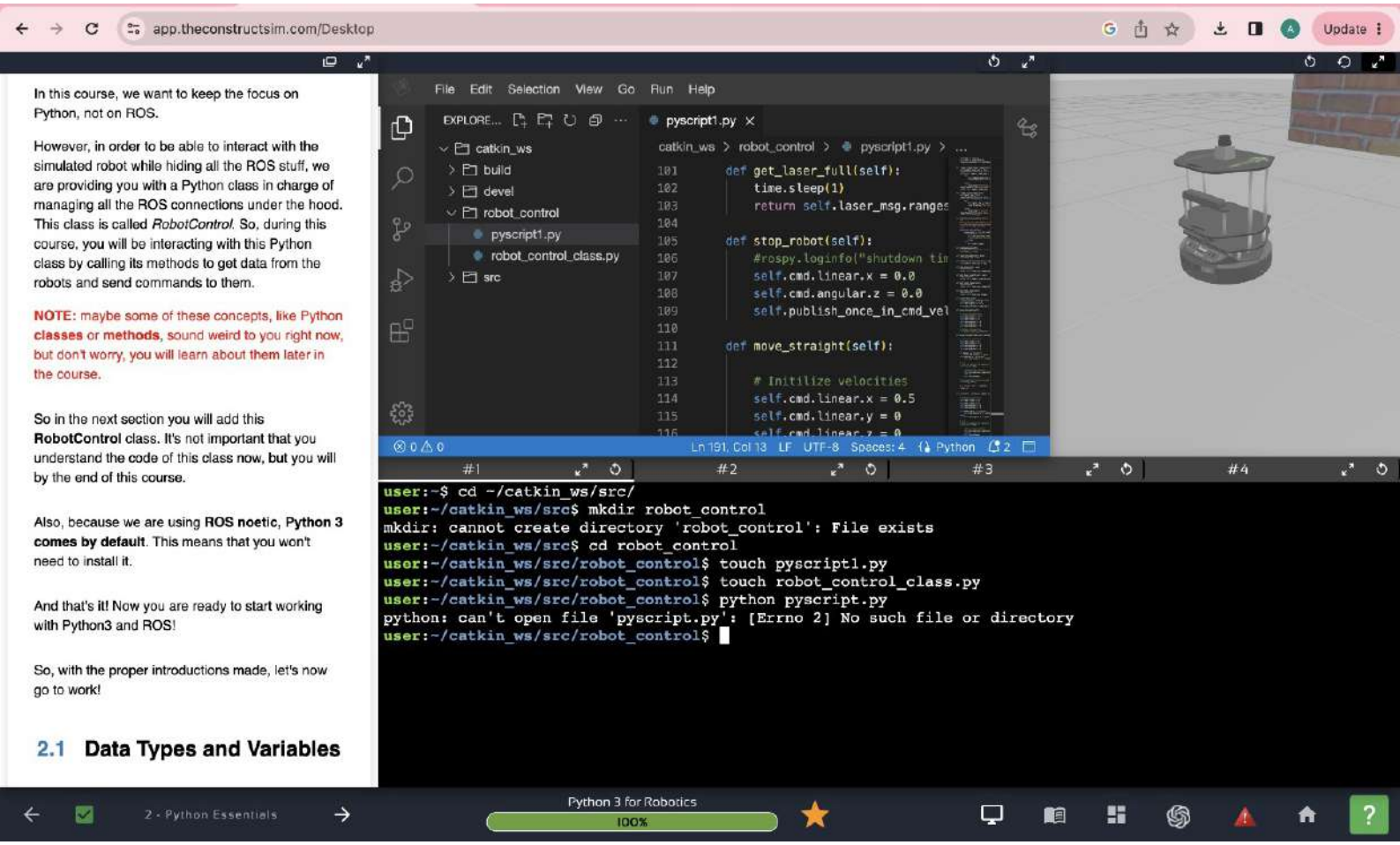
#1 #2 #3 #4

```
user:~$
user:~$ gv = "Empty"
bash: gv: command not found
user:~$ print (gv)
bash: syntax error near unexpected token `gv'
user:~$ f()
> print(gv)
bash: syntax error near unexpected token `print'
user:~$ from robot_control_class import RobotControl
from: can't read /var/mail/robot_control_class
user:~$ from robot_control_class import RobotControl
from: can't read /var/mail/robot_control_class
user:~$ robotcontrol = RobotControl(robot_name="summit")
bash: syntax error near unexpected token `('
user:~$
```

4 - Methods

Python 3 for Robotics

100%


app.theconstructsim.com/Desktop
Update
In this course, we want to keep the focus on Python, not on ROS.
However, in order to be able to interact with the simulated robot while hiding all the ROS stuff, we are providing you with a Python class in charge of managing all the ROS connections under the hood. This class is called *RobotControl*. So, during this course, you will be interacting with this Python class by calling its methods to get data from the robots and send commands to them.
NOTE: maybe some of these concepts, like Python classes or methods, sound weird to you right now, but don't worry, you will learn about them later in the course.
So in the next section you will add this **RobotControl** class. It's not important that you understand the code of this class now, but you will by the end of this course.
Also, because we are using **ROS noetic**, **Python 3 comes by default**. This means that you won't need to install it.
And that's it! Now you are ready to start working with Python3 and ROS!
So, with the proper introductions made, let's now go to work!
2.1 Data Types and Variables
File Edit Selection View Go Run Help
EXPLORE...
catkin_ws
build
devel
robot_control
pyscript1.py
robot_control_class.py
src
pyscript1.py
catkin_ws > robot_control > pyscript1.py > ...
def get_laser_full(self):
time.sleep(1)
return self.laser_msg.ranges
def stop_robot(self):
#rospy.loginfo("shutdown tim
self.cmd.linear.x = 0.0
self.cmd.angular.z = 0.0
self.publish_once_in_cmd_vel
def move_straight(self):
# Initilize velocities
self.cmd.linear.x = 0.5
self.cmd.linear.y = 0
Ln 191, Col 13 LF UTF-8 Spaces: 4 Python
#1
#2
#3
#4
user:~$ cd ~/catkin_ws/src/
user:~/catkin_ws/src$ mkdir robot_control
mkdir: cannot create directory 'robot_control': File exists
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 pyscript.py
python: can't open file 'pyscript.py': [Errno 2] No such file or directory
user:~/catkin_ws/src/robot_control$
2 - Python Essentials
Python 3 for Robotics
100%