

# Team 1 – Task 1

(Computer vision, Navigation)

Akenat Loawkitpanich 6210552706

Nattanon Manotraitrat 6210554661

# Objectives

- Robot can detect a person
- Navigate to a person (skipped)
- Detect a raised hand
- Avoid obstacle (skipped)

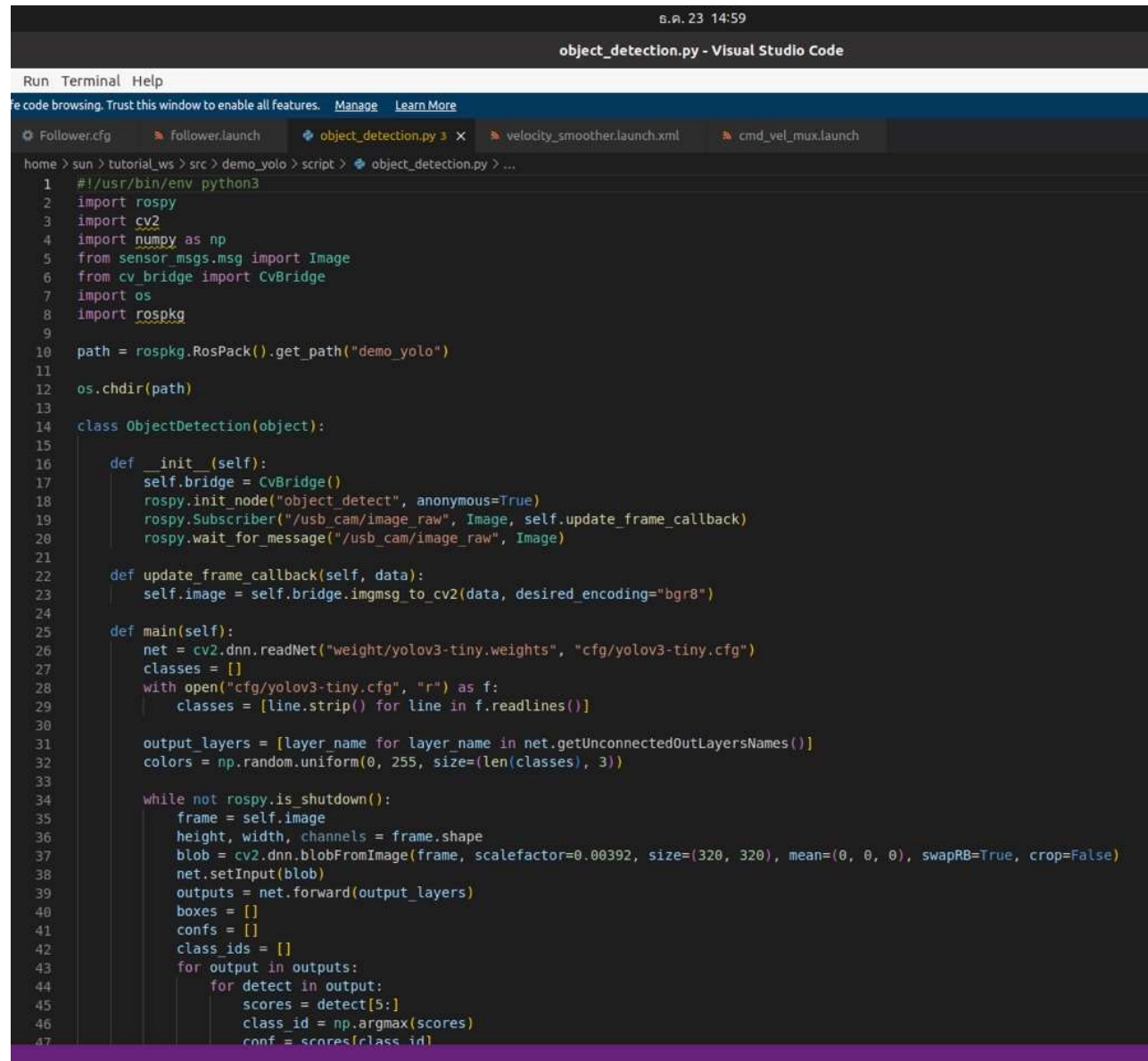
# Task 1.1

for person detection

- Using yolo\_v4

- Weight:

[https://github.com/AlexeyAB/darknet/releases/download/darknet\\_yolo\\_v3\\_optimal/yolov3.weights](https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov3.weights)



The screenshot shows a Visual Studio Code editor window titled "object\_detection.py - Visual Studio Code". The editor displays a Python script for object detection. The script imports necessary libraries like rospy, cv2, numpy, sensor\_msgs.msg, cv\_bridge, os, and rospkg. It sets the current directory to the path of the demo\_yolo package. A class named ObjectDetection is defined with an \_\_init\_\_ method that initializes a CvBridge, sets up a ROS node, and subscribes to a camera image topic. The update\_frame\_callback method processes the received image data. The main method reads the YOLOv3 weights and configuration files, sets up the neural network, and enters a loop to process frames. In the loop, it reads the current frame, converts it to a blob, sets it as input to the network, and forwards it to the output layers. It then processes the outputs to detect objects, returning a list of bounding boxes, confidence scores, and class IDs.

```
6.9.23 14:59
object_detection.py - Visual Studio Code

Run Terminal Help
code browsing. Trust this window to enable all features. Manage Learn More

Follower.cfg follower.launch object_detection.py x velocity_smoother.launch.xml cmd_vel_mux.launch

home > sun > tutorial_ws > src > demo_yolo > script > object_detection.py > ...
1  #!/usr/bin/env python3
2  import rospy
3  import cv2
4  import numpy as np
5  from sensor_msgs.msg import Image
6  from cv_bridge import CvBridge
7  import os
8  import rospkg
9
10 path = rospkg.RosPack().get_path("demo_yolo")
11
12 os.chdir(path)
13
14 class ObjectDetection(object):
15
16     def __init__(self):
17         self.bridge = CvBridge()
18         rospy.init_node("object_detect", anonymous=True)
19         rospy.Subscriber("/usb_cam/image_raw", Image, self.update_frame_callback)
20         rospy.wait_for_message("/usb_cam/image_raw", Image)
21
22     def update_frame_callback(self, data):
23         self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")
24
25     def main(self):
26         net = cv2.dnn.readNet("weight/yolov3-tiny.weights", "cfg/yolov3-tiny.cfg")
27         classes = []
28         with open("cfg/yolov3-tiny.cfg", "r") as f:
29             classes = [line.strip() for line in f.readlines()]
30
31         output_layers = [layer_name for layer_name in net.getUnconnectedOutLayersNames()]
32         colors = np.random.uniform(0, 255, size=(len(classes), 3))
33
34         while not rospy.is_shutdown():
35             frame = self.image
36             height, width, channels = frame.shape
37             blob = cv2.dnn.blobFromImage(frame, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
38             net.setInput(blob)
39             outputs = net.forward(output_layers)
40             boxes = []
41             confs = []
42             class_ids = []
43             for output in outputs:
44                 for detect in output:
45                     scores = detect[5:]
46                     class_id = np.argmax(scores)
47                     conf = scores[class_id]
```

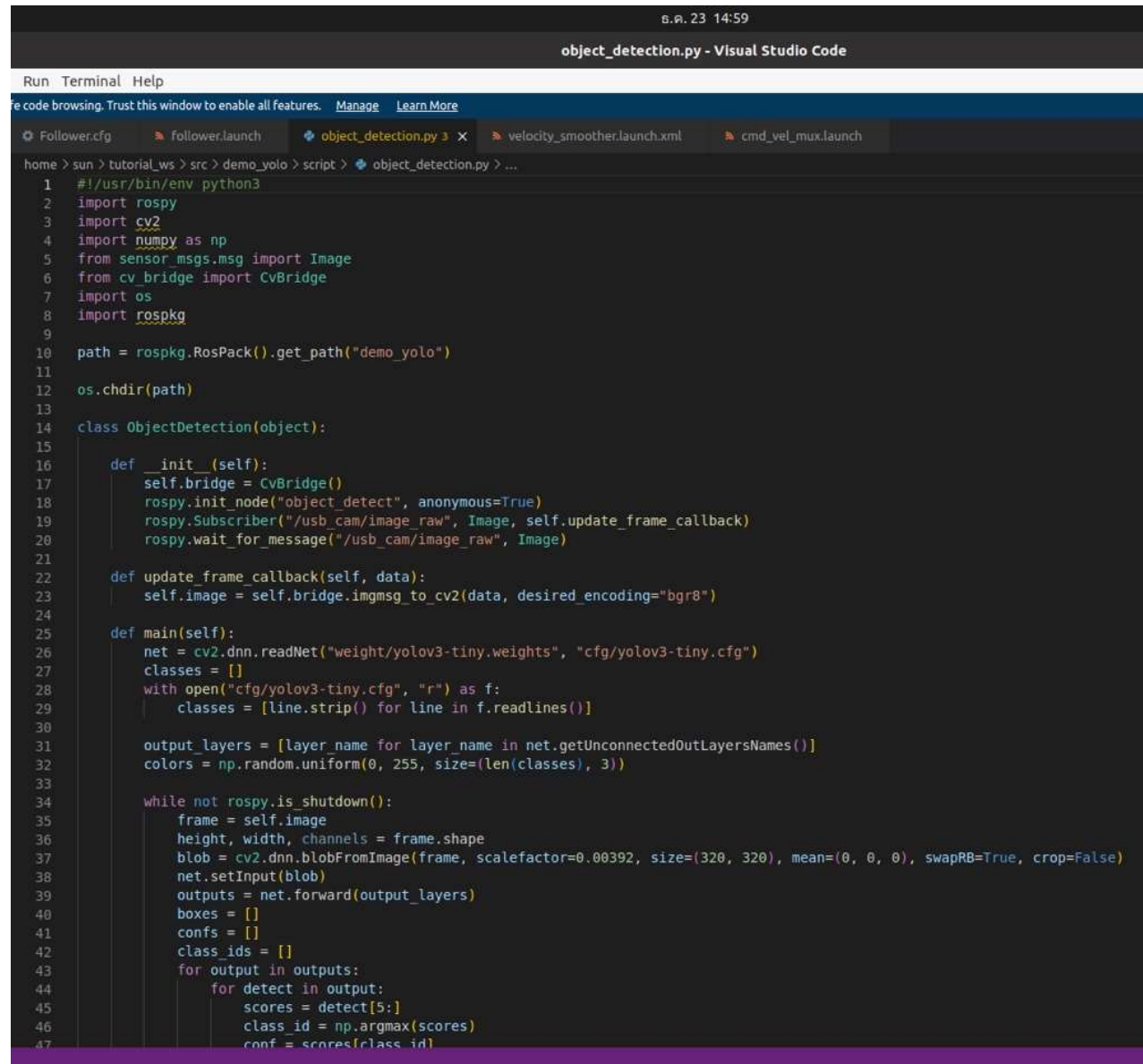
# Task 1.1

for person detection

- declare the path for ros package and change to that directory

In ObjectDetection function:

- Change the input to use the Kinect
- Then importing the weight to detect a person



```
object_detection.py - Visual Studio Code

Run Terminal Help
code browsing. Trust this window to enable all features. Manage Learn More

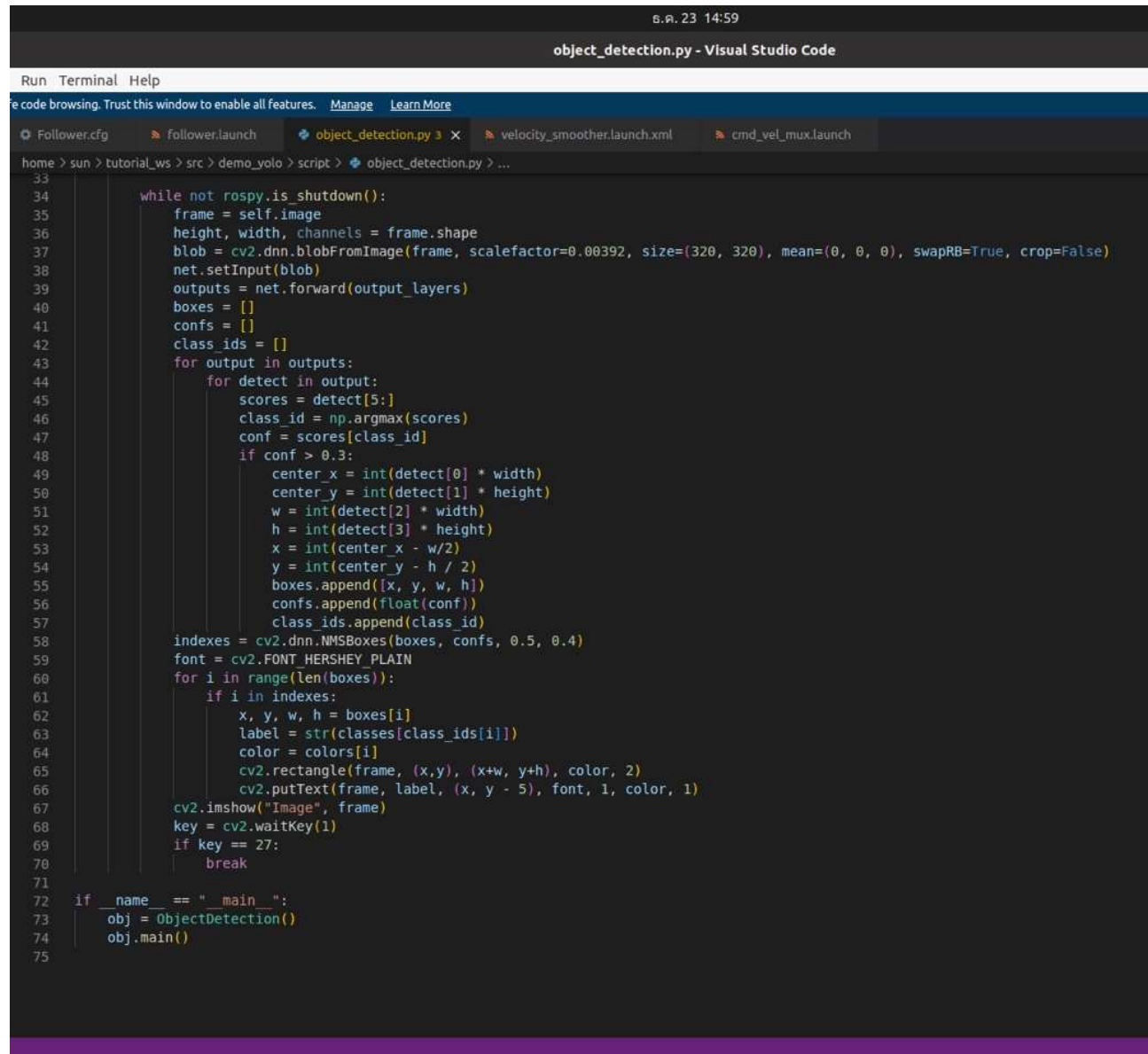
Follower.cfg follower.launch object_detection.py x velocity_smoother.launch.xml cmd_vel_mux.launch

home > sun > tutorial_ws > src > demo_yolo > script > object_detection.py > ...
1  #!/usr/bin/env python3
2  import rospy
3  import cv2
4  import numpy as np
5  from sensor_msgs.msg import Image
6  from cv_bridge import CvBridge
7  import os
8  import rospkg
9
10 path = rospkg.RosPack().get_path("demo_yolo")
11
12 os.chdir(path)
13
14 class ObjectDetection(object):
15
16     def __init__(self):
17         self.bridge = CvBridge()
18         rospy.init_node("object_detect", anonymous=True)
19         rospy.Subscriber("/usb_cam/image_raw", Image, self.update_frame_callback)
20         rospy.wait_for_message("/usb_cam/image_raw", Image)
21
22     def update_frame_callback(self, data):
23         self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")
24
25     def main(self):
26         net = cv2.dnn.readNet("weight/yolov3-tiny.weights", "cfg/yolov3-tiny.cfg")
27         classes = []
28         with open("cfg/yolov3-tiny.cfg", "r") as f:
29             classes = [line.strip() for line in f.readlines()]
30
31         output_layers = [layer_name for layer_name in net.getUnconnectedOutLayersNames()]
32         colors = np.random.uniform(0, 255, size=(len(classes), 3))
33
34         while not rospy.is_shutdown():
35             frame = self.image
36             height, width, channels = frame.shape
37             blob = cv2.dnn.blobFromImage(frame, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
38             net.setInput(blob)
39             outputs = net.forward(output_layers)
40             boxes = []
41             confs = []
42             class_ids = []
43             for output in outputs:
44                 for detect in output:
45                     scores = detect[5:]
46                     class_id = np.argmax(scores)
47                     conf = scores[class_id]
```

# Task 1.1

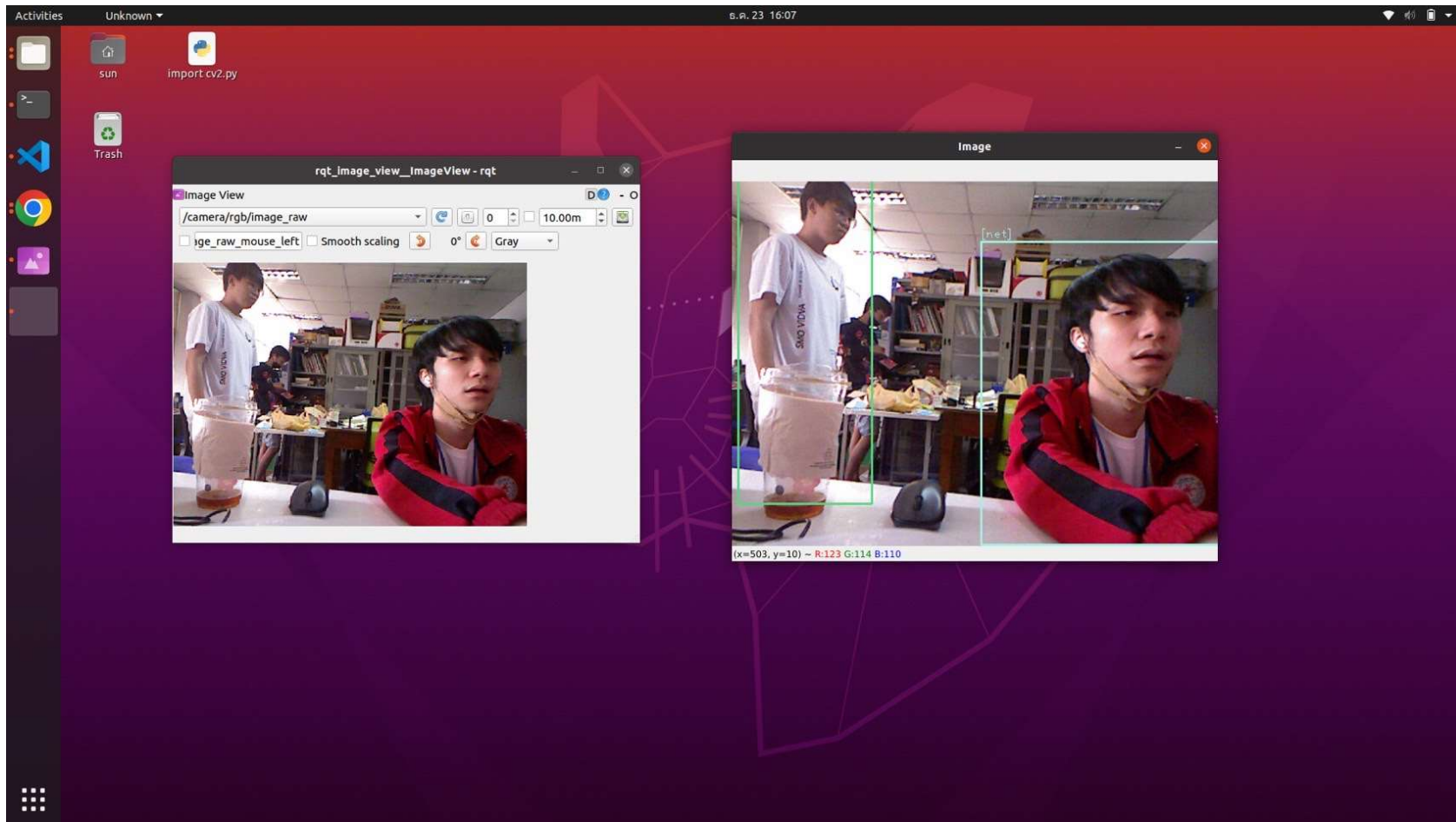
for person detection

- The code below is just for the output to show up as a box with different colour for different persons (the colour is random)
- Then call a command in main



```
object_detection.py - Visual Studio Code
Run Terminal Help
code browsing. Trust this window to enable all features. Manage Learn More
Follower.cfg follower.launch object_detection.py x velocity_smoother.launch.xml cmd_vel_mux.launch
home > sun > tutorial_ws > src > demo_yolo > script > object_detection.py > ...
33
34 while not rospy.is_shutdown():
35     frame = self.image
36     height, width, channels = frame.shape
37     blob = cv2.dnn.blobFromImage(frame, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
38     net.setInput(blob)
39     outputs = net.forward(output_layers)
40     boxes = []
41     confs = []
42     class_ids = []
43     for output in outputs:
44         for detect in output:
45             scores = detect[5:]
46             class_id = np.argmax(scores)
47             conf = scores[class_id]
48             if conf > 0.3:
49                 center_x = int(detect[0] * width)
50                 center_y = int(detect[1] * height)
51                 w = int(detect[2] * width)
52                 h = int(detect[3] * height)
53                 x = int(center_x - w/2)
54                 y = int(center_y - h / 2)
55                 boxes.append([x, y, w, h])
56                 confs.append(float(conf))
57                 class_ids.append(class_id)
58     indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)
59     font = cv2.FONT_HERSHEY_PLAIN
60     for i in range(len(boxes)):
61         if i in indexes:
62             x, y, w, h = boxes[i]
63             label = str(classes[class_ids[i]])
64             color = colors[i]
65             cv2.rectangle(frame, (x,y), (x+w, y+h), color, 2)
66             cv2.putText(frame, label, (x, y - 5), font, 1, color, 1)
67     cv2.imshow("Image", frame)
68     key = cv2.waitKey(1)
69     if key == 27:
70         break
71
72 if __name__ == "__main__":
73     obj = ObjectDetection()
74     obj.main()
75
```

# Person Tracking

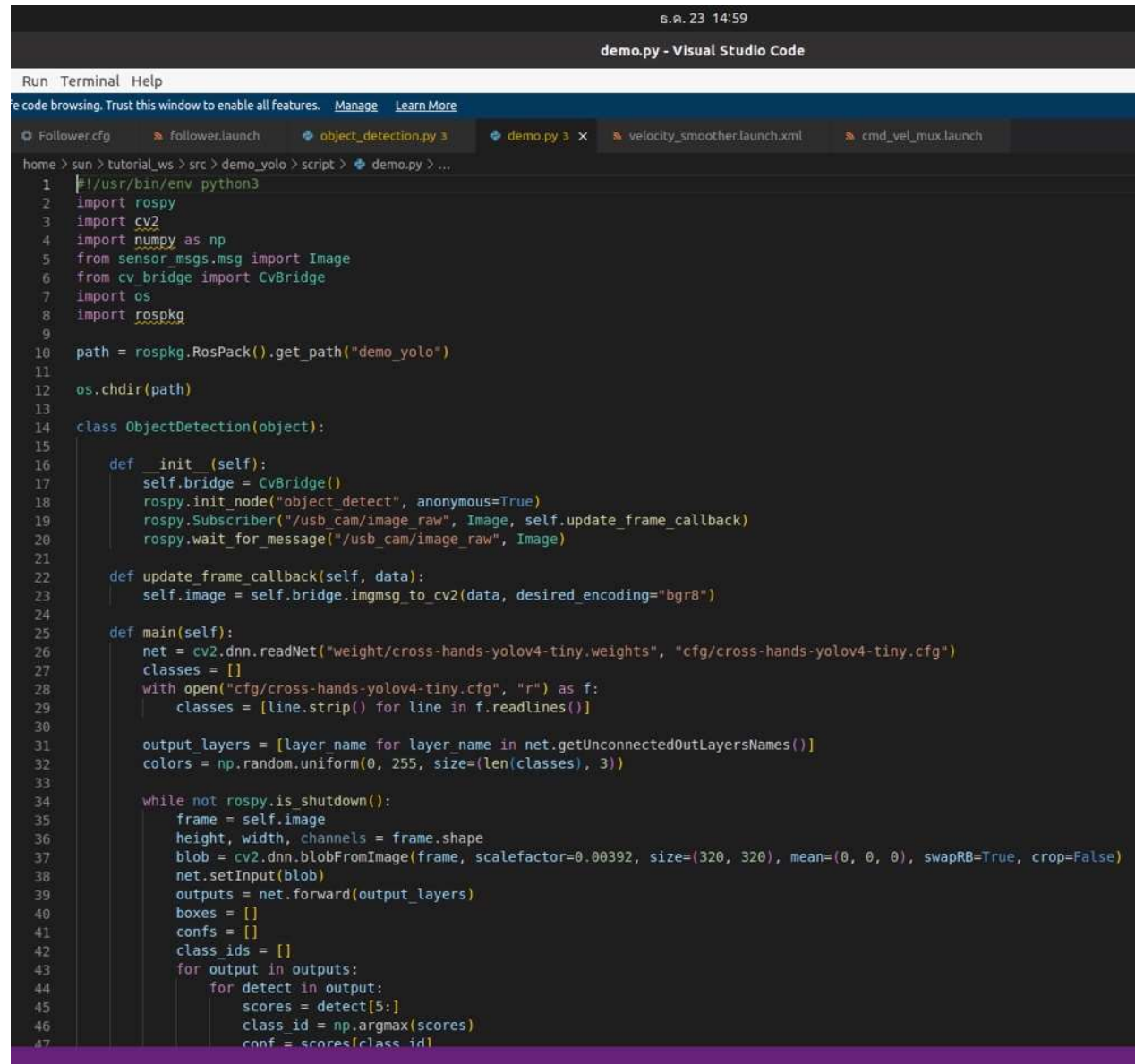




# Task 1.3

for hands detection

- Weight & original code:  
<https://github.com/cansik/yolo-hand-detection>
- Modified from the original code to get hands by changing path and config to hands model

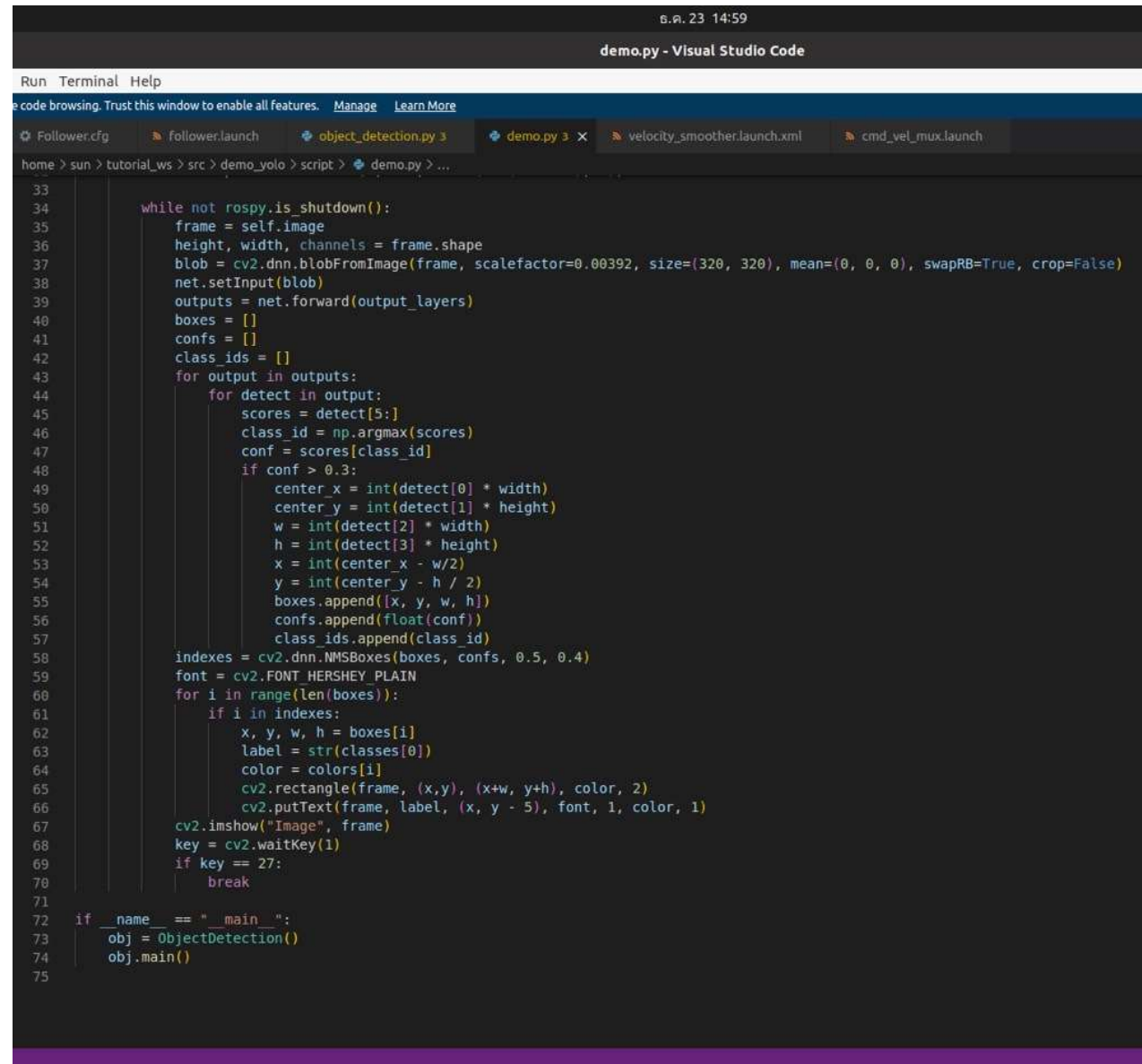


```
6.9. 23 14:59
demo.py - Visual Studio Code
Run Terminal Help
code browsing. Trust this window to enable all features. Manage Learn More
Follower.cfg follower.launch object_detection.py 3 demo.py 3 x velocity_smoother.launch.xml cmd_vel_mux.launch
home > sun > tutorial_ws > src > demo_volo > script > demo.py > ...
1 #!/usr/bin/env python3
2 import rospy
3 import cv2
4 import numpy as np
5 from sensor_msgs.msg import Image
6 from cv_bridge import CvBridge
7 import os
8 import rospkg
9
10 path = rospkg.RosPack().get_path("demo_volo")
11
12 os.chdir(path)
13
14 class ObjectDetection(object):
15
16     def __init__(self):
17         self.bridge = CvBridge()
18         rospy.init_node("object_detect", anonymous=True)
19         rospy.Subscriber("/usb_cam/image_raw", Image, self.update_frame_callback)
20         rospy.wait_for_message("/usb_cam/image_raw", Image)
21
22     def update_frame_callback(self, data):
23         self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")
24
25     def main(self):
26         net = cv2.dnn.readNet("weight/cross-hands-yolov4-tiny.weights", "cfg/cross-hands-yolov4-tiny.cfg")
27         classes = []
28         with open("cfg/cross-hands-yolov4-tiny.cfg", "r") as f:
29             classes = [line.strip() for line in f.readlines()]
30
31         output_layers = [layer_name for layer_name in net.getUnconnectedOutLayersNames()]
32         colors = np.random.uniform(0, 255, size=(len(classes), 3))
33
34         while not rospy.is_shutdown():
35             frame = self.image
36             height, width, channels = frame.shape
37             blob = cv2.dnn.blobFromImage(frame, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
38             net.setInput(blob)
39             outputs = net.forward(output_layers)
40             boxes = []
41             confs = []
42             class_ids = []
43             for output in outputs:
44                 for detect in output:
45                     scores = detect[5:]
46                     class_id = np.argmax(scores)
47                     conf = scores[class_id]
```

# Task 1.3

for hands detection

- The rest is the same as in task 1.1



```
6.6.23 14:59
demo.py - Visual Studio Code

Run Terminal Help
code browsing. Trust this window to enable all features. Manage Learn More
Follower.cfg follower.launch object_detection.py 3 demo.py 3 x velocity_smoother.launch.xml cmd_vel_mux.launch
home > sun > tutorial_ws > src > demo_yolo > script > demo.py > ...

33
34 while not rospy.is_shutdown():
35     frame = self.image
36     height, width, channels = frame.shape
37     blob = cv2.dnn.blobFromImage(frame, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
38     net.setInput(blob)
39     outputs = net.forward(output_layers)
40     boxes = []
41     confs = []
42     class_ids = []
43     for output in outputs:
44         for detect in output:
45             scores = detect[5:]
46             class_id = np.argmax(scores)
47             conf = scores[class_id]
48             if conf > 0.3:
49                 center_x = int(detect[0] * width)
50                 center_y = int(detect[1] * height)
51                 w = int(detect[2] * width)
52                 h = int(detect[3] * height)
53                 x = int(center_x - w/2)
54                 y = int(center_y - h / 2)
55                 boxes.append([x, y, w, h])
56                 confs.append(float(conf))
57                 class_ids.append(class_id)
58     indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)
59     font = cv2.FONT_HERSHEY_PLAIN
60     for i in range(len(boxes)):
61         if i in indexes:
62             x, y, w, h = boxes[i]
63             label = str(classes[class_ids[i]])
64             color = colors[class_ids[i]]
65             cv2.rectangle(frame, (x,y), (x+w, y+h), color, 2)
66             cv2.putText(frame, label, (x, y - 5), font, 1, color, 1)
67     cv2.imshow("Image", frame)
68     key = cv2.waitKey(1)
69     if key == 27:
70         break
71
72 if __name__ == "__main__":
73     obj = ObjectDetection()
74     obj.main()
75
```



# Hand Tracking



# State Machine

(Not working)

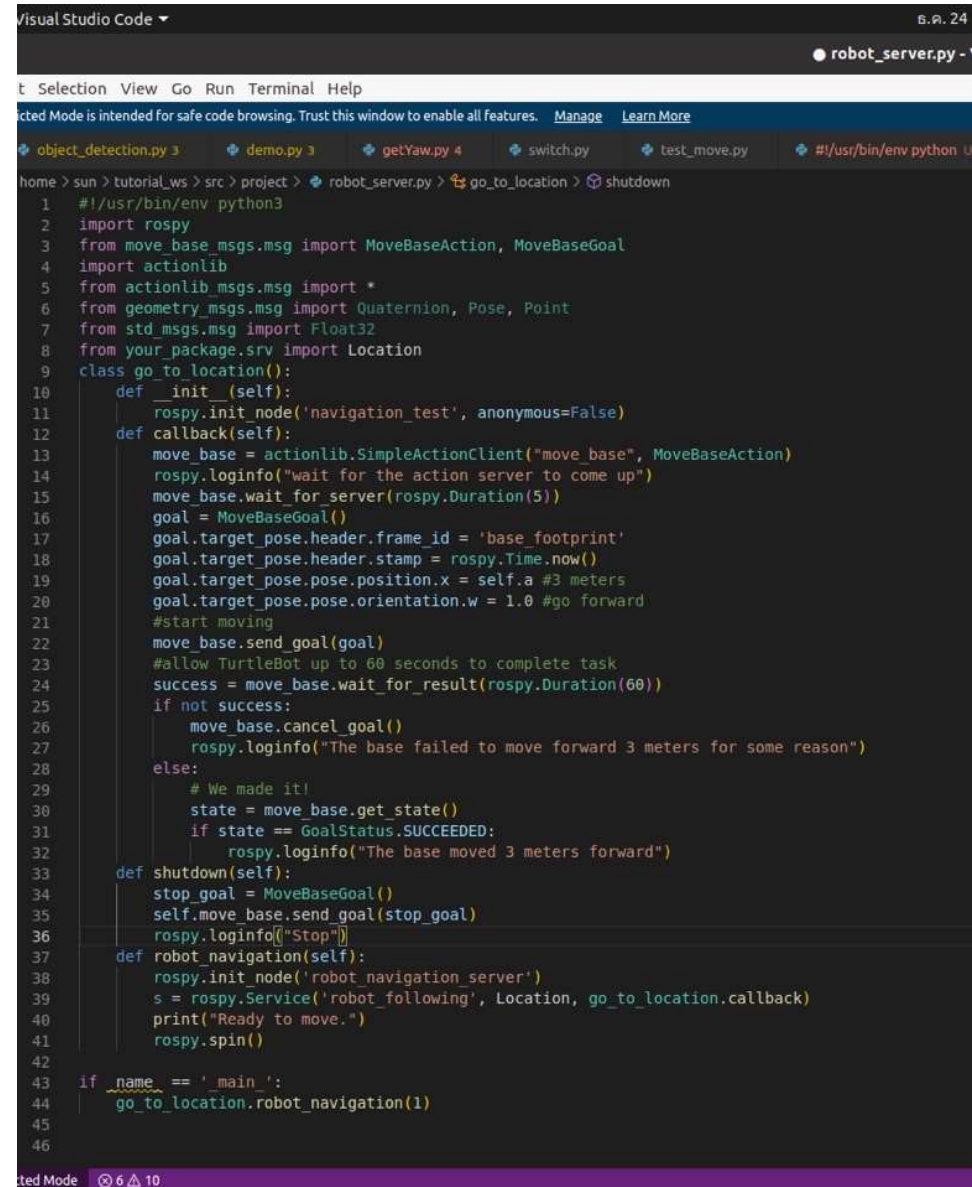
In theory:

- Get string confirming result
- If person is detected (move on to detect hand)
- If hand is detected (nav command run that included with obstacle avoidance)

```
smach.py •
smach.py > ...
1  #!/usr/bin/env python3
2
3  import rospy
4  import smach
5  import smach_ros
6  from std_msgs.msg import String
7
8  def callback(data):
9      rospy.loginfo(rospy.get_caller_id() + "next state", data.data)
10 def detect_p(res1):
11     rospy.init_node('detect_p', anonymous=True)
12     rospy.Subscriber("person", String, callback)
13     rospy.spin()
14     return res1 == True
15 def detect_h(res2):
16     rospy.init_node('detect_h', anonymous=True)
17     rospy.Subscriber("hand", String, callback)
18     rospy.spin()
19     return res2 == True
20
21 # define state Foo
22 class Foo(smach.State):
23     def __init__(self):
24         smach.State.__init__(self, outcomes=['outcome1', 'outcome2'])
25
26     def execute(self, userdata):
27         rospy.loginfo('Executing state Find Peeps')
28         if detect_p()==True:
29             return 'outcome1'
30         else:
31             return 'outcome2'
32
33 # define state Bar
34 class Bar(smach.State):
```

# Skipped Tasks

- GoToLocation code modification
- The obstacle avoidance is in gotolocation



The image shows a Visual Studio Code editor window with a Python script named `robot_server.py`. The script is for a robot navigation server. It includes imports for `rospy`, `move_base_msgs`, `actionlib`, `geometry_msgs`, `std_msgs`, and a custom `Location` service. The main logic is in the `go_to_location` class, which has methods for initialization, goal sending, state checking, and shutdown. The `robot_navigation` method sets up a service to handle navigation requests. The script is run in a terminal window with the command `python3 robot_server.py`.

```
Visual Studio Code
robot_server.py

t Selection View Go Run Terminal Help
cted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

object_detection.py 3 demo.py 3 getYaw.py 4 switch.py test_move.py #!/usr/bin/env python3

home > sun > tutorial_ws > src > project > robot_server.py > go_to_location > shutdown

1 #!/usr/bin/env python3
2 import rospy
3 from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
4 import actionlib
5 from actionlib_msgs.msg import *
6 from geometry_msgs.msg import Quaternion, Pose, Point
7 from std_msgs.msg import Float32
8 from your_package.srv import Location
9 class go_to_location():
10     def __init__(self):
11         rospy.init_node('navigation_test', anonymous=False)
12     def callback(self):
13         move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
14         rospy.loginfo("wait for the action server to come up")
15         move_base.wait_for_server(rospy.Duration(5))
16         goal = MoveBaseGoal()
17         goal.target_pose.header.frame_id = 'base_footprint'
18         goal.target_pose.header.stamp = rospy.Time.now()
19         goal.target_pose.pose.position.x = self.a #3 meters
20         goal.target_pose.pose.orientation.w = 1.0 #go forward
21         #start moving
22         move_base.send_goal(goal)
23         #allow TurtleBot up to 60 seconds to complete task
24         success = move_base.wait_for_result(rospy.Duration(60))
25         if not success:
26             move_base.cancel_goal()
27             rospy.loginfo("The base failed to move forward 3 meters for some reason")
28         else:
29             # We made it!
30             state = move_base.get_state()
31             if state == GoalStatus.SUCCEEDED:
32                 rospy.loginfo("The base moved 3 meters forward")
33     def shutdown(self):
34         stop_goal = MoveBaseGoal()
35         self.move_base.send_goal(stop_goal)
36         rospy.loginfo("Stop")
37     def robot_navigation(self):
38         rospy.init_node('robot_navigation_server')
39         s = rospy.Service('robot_following', Location, go_to_location.callback)
40         print("Ready to move.")
41         rospy.spin()
42
43 if __name__ == '__main__':
44     go_to_location().robot_navigation(1)
45
46
```