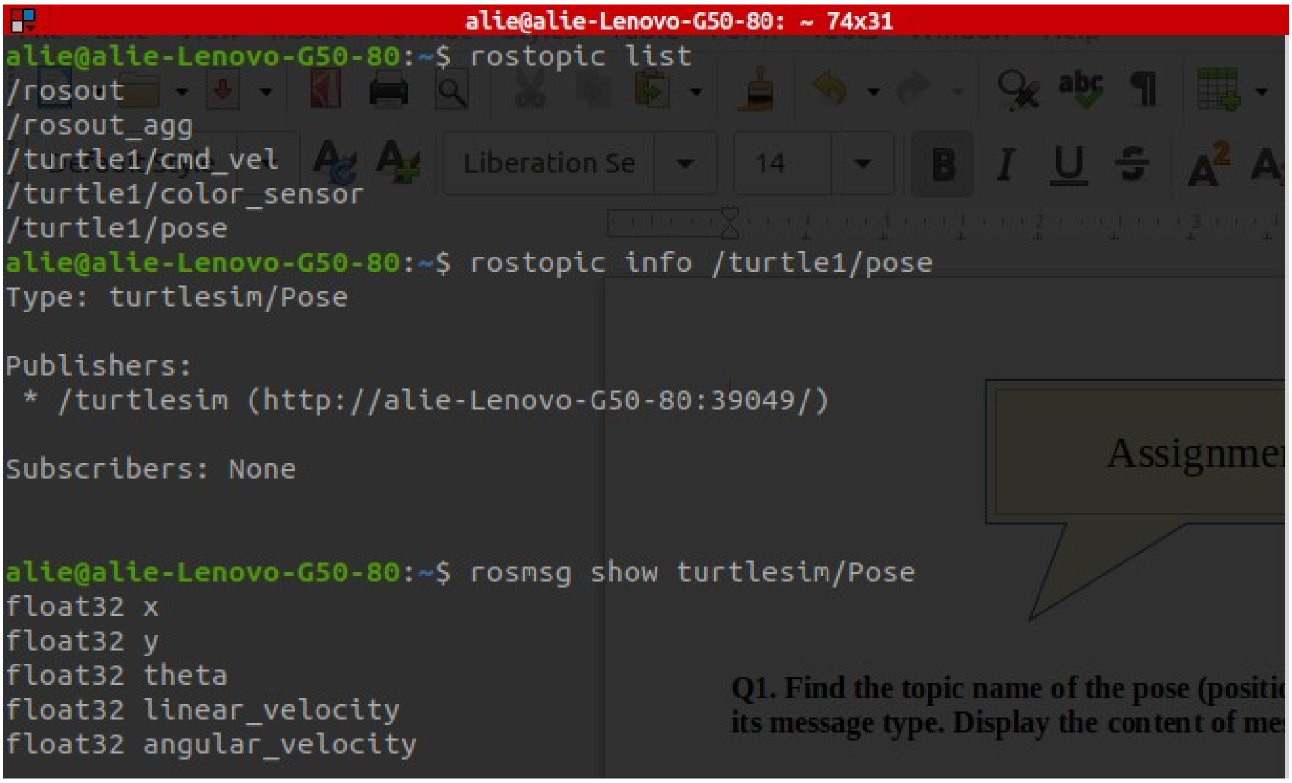
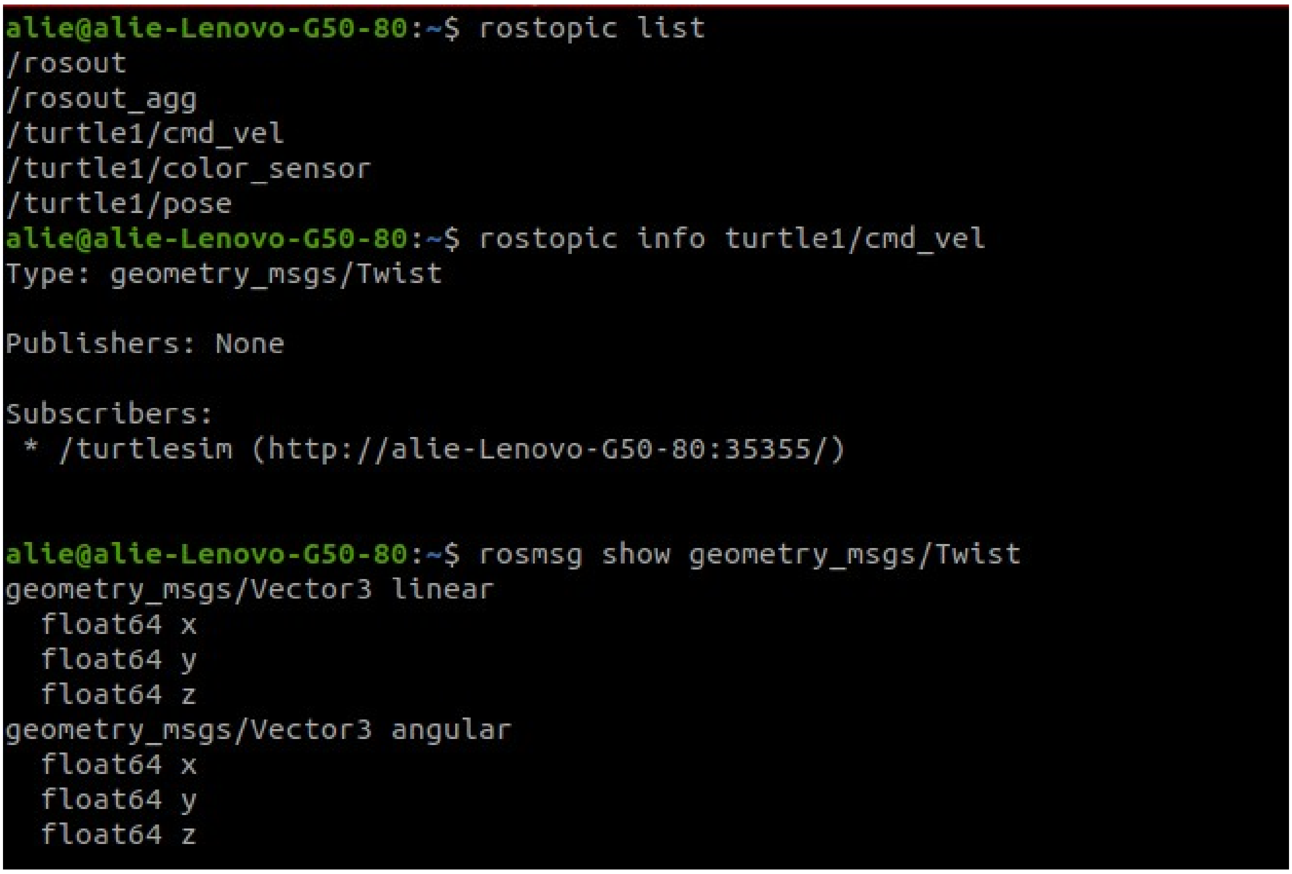
Assignment 2

Q1. Find the topic name of the pose (position and orientation) of turtlesim and its message type. Display the content of message of the pose.



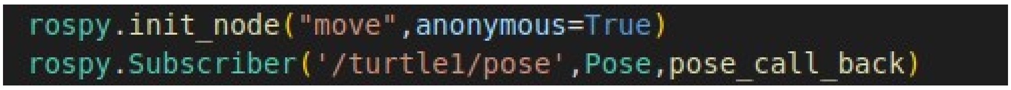
```
alie@alie-Lenovo-G50-80:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
alie@alie-Lenovo-G50-80:~$ rostopic info /turtle1/pose
Type: turtlesim/Pose

Publishers:
 * /turtlesim (http://alie-Lenovo-G50-80:39049/)

Subscribers: None


alie@alie-Lenovo-G50-80:~$ rosmsg show turtlesim/Pose
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

Q2. Find the topic name of the velocity command of turtlesim and its message type. Display the content of message of the velocity command.

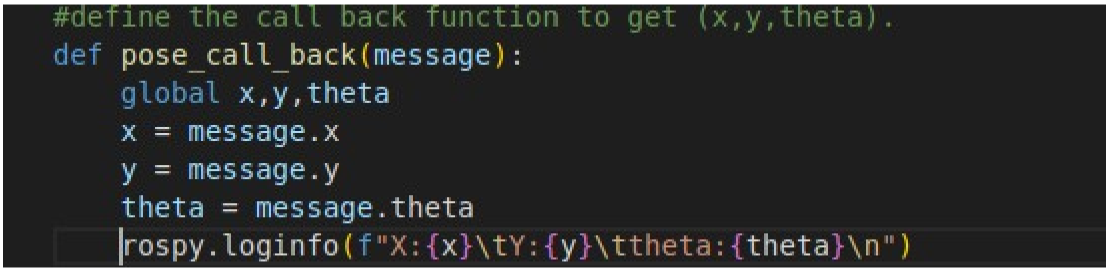


```
alie@alie-Lenovo-G50-80:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
alie@alie-Lenovo-G50-80:~$ rostopic info turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
 * /turtlesim (http://alie-Lenovo-G50-80:35355/)


alie@alie-Lenovo-G50-80:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

Q3. Write a simple ROS node in a script file called turtlesim_pose.py, which subscribes to the topic of the
pose, and then prints the position of the robot in the callback function.

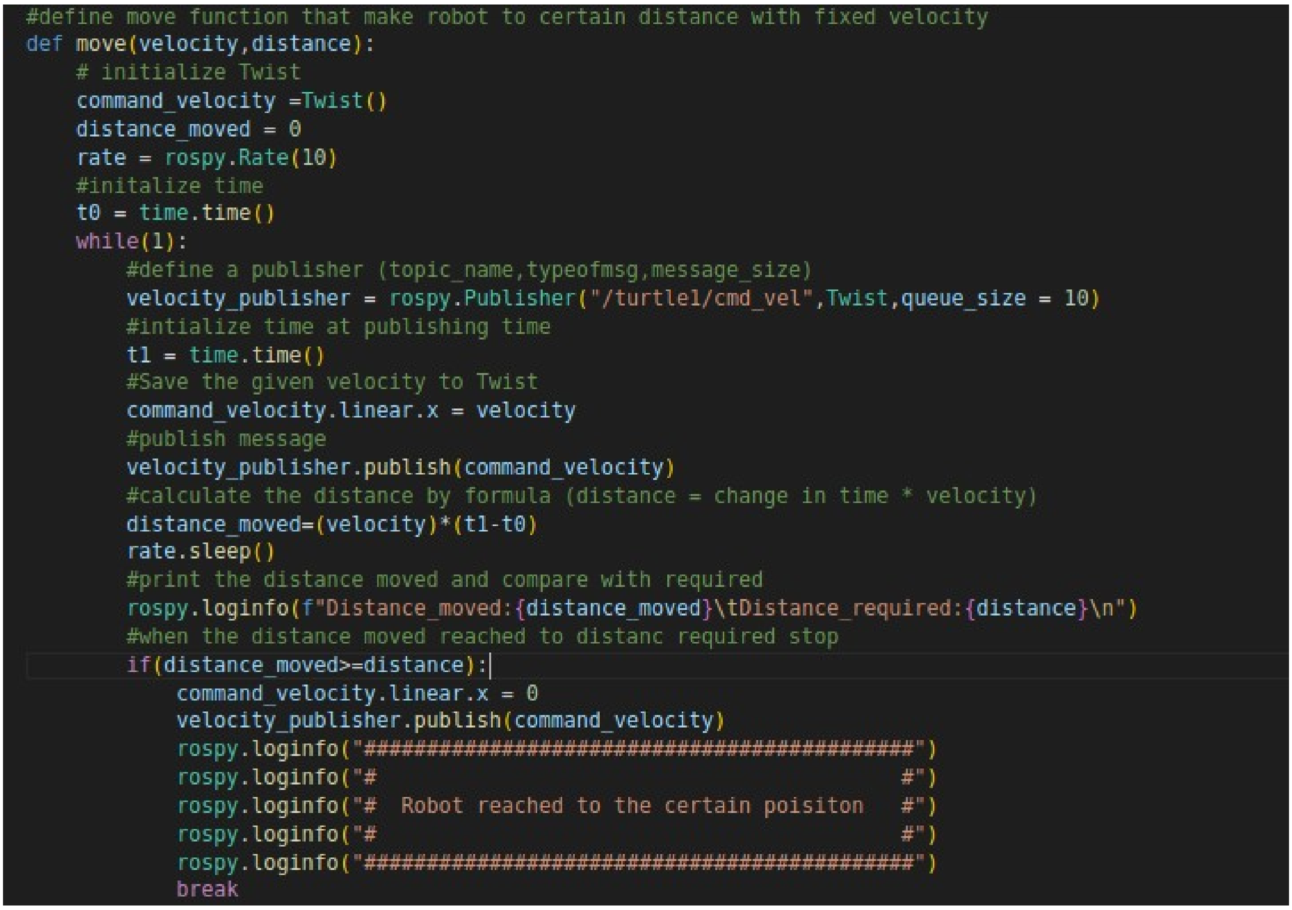


```
rospy.init_node("move",anonymous=True)
rospy.Subscriber('/turtle1/pose',Pose,pose_call_back)
```

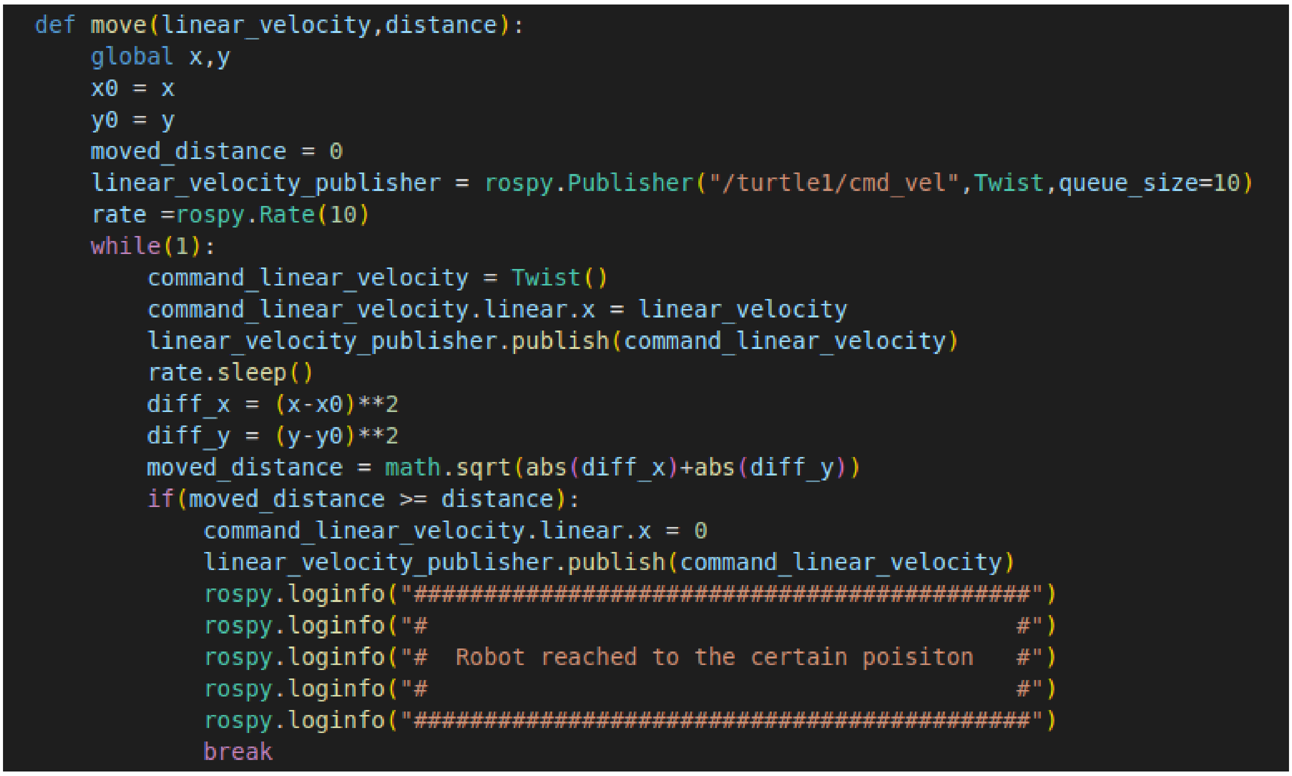


```
#define the call back function to get (x,y,theta).
def pose_call_back(message):
    global x,y,theta
    x = message.x
    y = message.y
    theta = message.theta
    rospy.loginfo(f"X:{x}\tY:{y}\ttheta:{theta}\n")
```

Q4. Complete the previous code in turtlesim_pose.py to add a publisher to the velocity and make the robot
move for a certain distance. Hint: you can use the rule
Distance = linear_speed * time



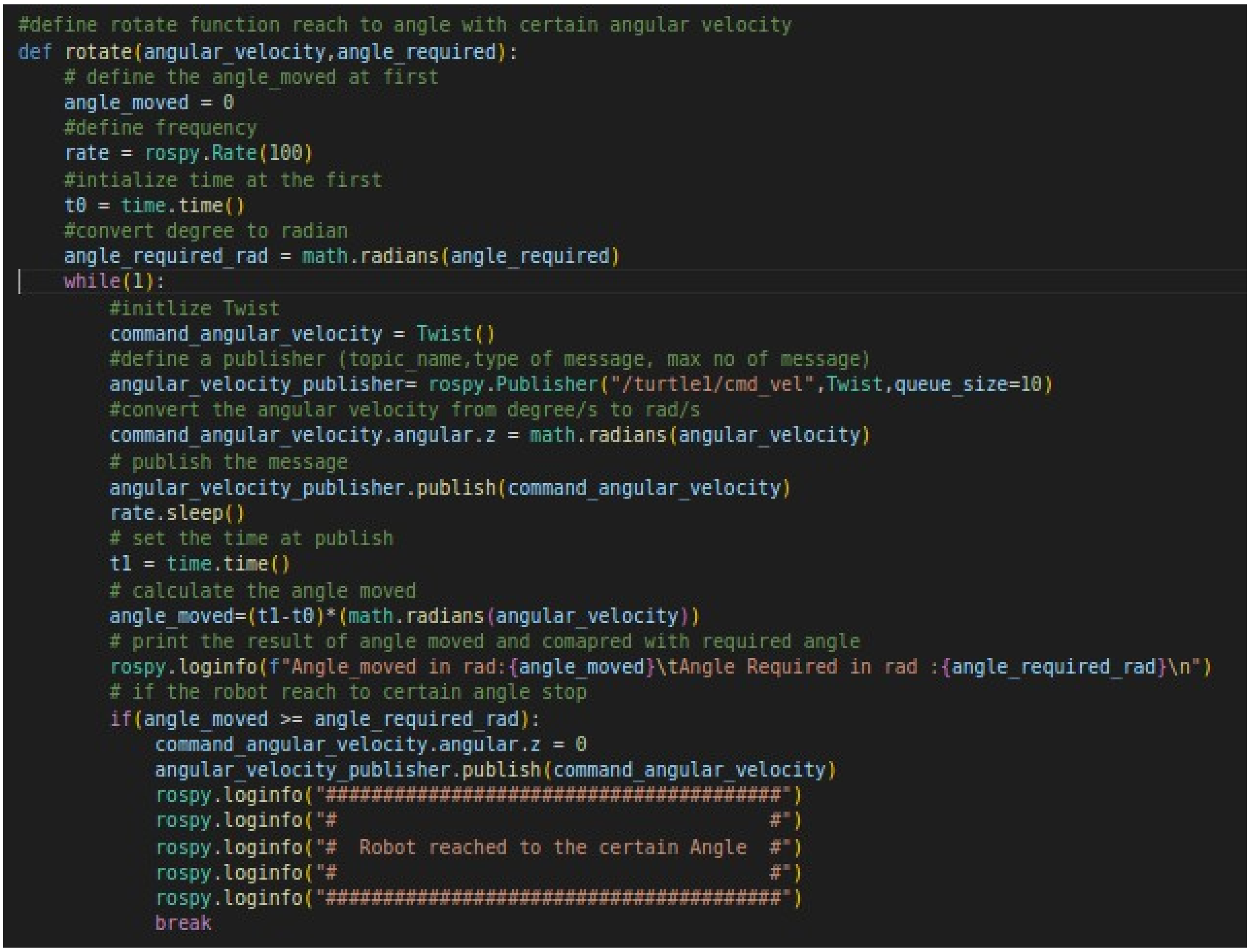
```python
#define move function that make robot to certain distance with fixed velocity
def move(velocity,distance):
    # initialize Twist
    command_velocity =Twist()
    distance_moved = 0
    rate = rospy.Rate(10)
    #initalize time
    t0 = time.time()
    while(1):
        #define a publisher (topic_name,typeofmsg,message_size)
        velocity_publisher = rospy.Publisher("/turtle1/cmd_vel",Twist,queue_size = 10)
        #intialize time at publishing time
        t1 = time.time()
        #Save the given velocity to Twist
        command_velocity.linear.x = velocity
        #publish message
        velocity_publisher.publish(command_velocity)
        #calculate the distance by formula (distance = change in time * velocity)
        distance_moved=(velocity)*(t1-t0)
        rate.sleep()
        #print the distance moved and compare with required
        rospy.loginfo(f"Distance_moved:{distance_moved}\tDistance_required:{distance}\n")
        #when the distance moved reached to distanc required stop
        if(distance_moved>=distance):
            command_velocity.linear.x = 0
            velocity_publisher.publish(command_velocity)
            rospy.loginfo("############################################")
            rospy.loginfo("#                                          #")
            rospy.loginfo("#  Robot reached to the certain poisiton   #")
            rospy.loginfo("#                                          #")
            rospy.loginfo("############################################")
            break
```



```python
def move(linear_velocity,distance):
    global x,y
    x0 = x
    y0 = y
    moved_distance = 0
    linear_velocity_publisher = rospy.Publisher("/turtle1/cmd_vel",Twist,queue_size=10)
    rate =rospy.Rate(10)
    while(1):
        command_linear_velocity = Twist()
        command_linear_velocity.linear.x = linear_velocity
        linear_velocity_publisher.publish(command_linear_velocity)
        rate.sleep()
        diff_x = (x-x0)**2
        diff_y = (y-y0)**2
        moved_distance = math.sqrt(abs(diff_x)+abs(diff_y))
        if(moved_distance >= distance):
            command_linear_velocity.linear.x = 0
            linear_velocity_publisher.publish(command_linear_velocity)
            rospy.loginfo("############################################")
            rospy.loginfo("#                                          #")
            rospy.loginfo("#  Robot reached to the certain poisiton   #")
            rospy.loginfo("#                                          #")
            rospy.loginfo("############################################")
            break
```

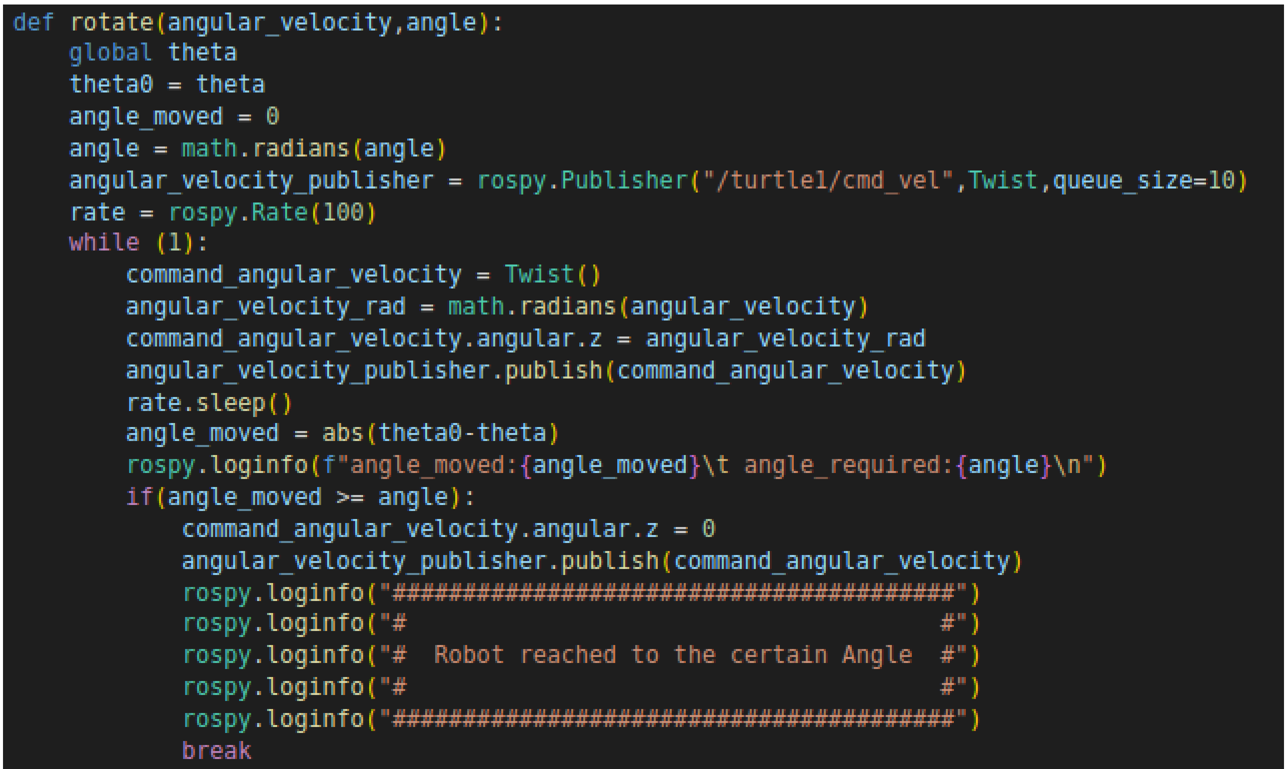
Q5. Complete the previous code in turtlesim_pose.py to add a publisher to the velocity and make the robot
rotate in place for a certain angle. Hint: you can use the rule
Angle = angular_speed * time

Method1



```python
#define rotate function reach to angle with certain angular velocity
def rotate(angular_velocity,angle_required):
    # define the angle_moved at first
    angle_moved = 0
    #define frequency
    rate = rospy.Rate(100)
    #intialize time at the first
    t0 = time.time()
    #convert degree to radian
    angle_required_rad = math.radians(angle_required)
    while(1):
        #initlize Twist
        command_angular_velocity = Twist()
        #define a publisher (topic_name,type of message, max no of message)
        angular_velocity_publisher= rospy.Publisher("/turtle1/cmd_vel",Twist,queue_size=10)
        #convert the angular velocity from degree/s to rad/s
        command_angular_velocity.angular.z = math.radians(angular_velocity)
        # publish the message
        angular_velocity_publisher.publish(command_angular_velocity)
        rate.sleep()
        # set the time at publish
        t1 = time.time()
        # calculate the angle moved
        angle_moved=(t1-t0)*(math.radians(angular_velocity))
        # print the result of angle moved and comapred with required angle
        rospy.loginfo(f"Angle_moved in rad:{angle_moved}\tAngle Required in rad :{angle_required_rad}\n")
        # if the robot reach to certain angle stop
        if(angle_moved >= angle_required_rad):
            command_angular_velocity.angular.z = 0
            angular_velocity_publisher.publish(command_angular_velocity)
            rospy.loginfo("########################################")
            rospy.loginfo("#                                      #")
            rospy.loginfo("#  Robot reached to the certain Angle  #")
            rospy.loginfo("#                                      #")
            rospy.loginfo("########################################")
            break
```

Method2



```python
def rotate(angular_velocity,angle):
    global theta
    theta0 = theta
    angle_moved = 0
    angle = math.radians(angle)
    angular_velocity_publisher = rospy.Publisher("/turtle1/cmd_vel",Twist,queue_size=10)
    rate = rospy.Rate(100)
    while (1):
        command_angular_velocity = Twist()
        angular_velocity_rad = math.radians(angular_velocity)
        command_angular_velocity.angular.z = angular_velocity_rad
        angular_velocity_publisher.publish(command_angular_velocity)
        rate.sleep()
        angle_moved = abs(theta0-theta)
        rospy.loginfo(f"angle_moved:{angle_moved}\t angle_required:{angle}\n")
        if(angle_moved >= angle):
            command_angular_velocity.angular.z = 0
            angular_velocity_publisher.publish(command_angular_velocity)
            rospy.loginfo("########################################")
            rospy.loginfo("#                                      #")
            rospy.loginfo("#  Robot reached to the certain Angle  #")
            rospy.loginfo("#                                      #")
            rospy.loginfo("########################################")
            break
```

Q6. Use your code above to make the robot move 1 meter and rotate 90 degrees.

Attached
- move1.py
- move2.py