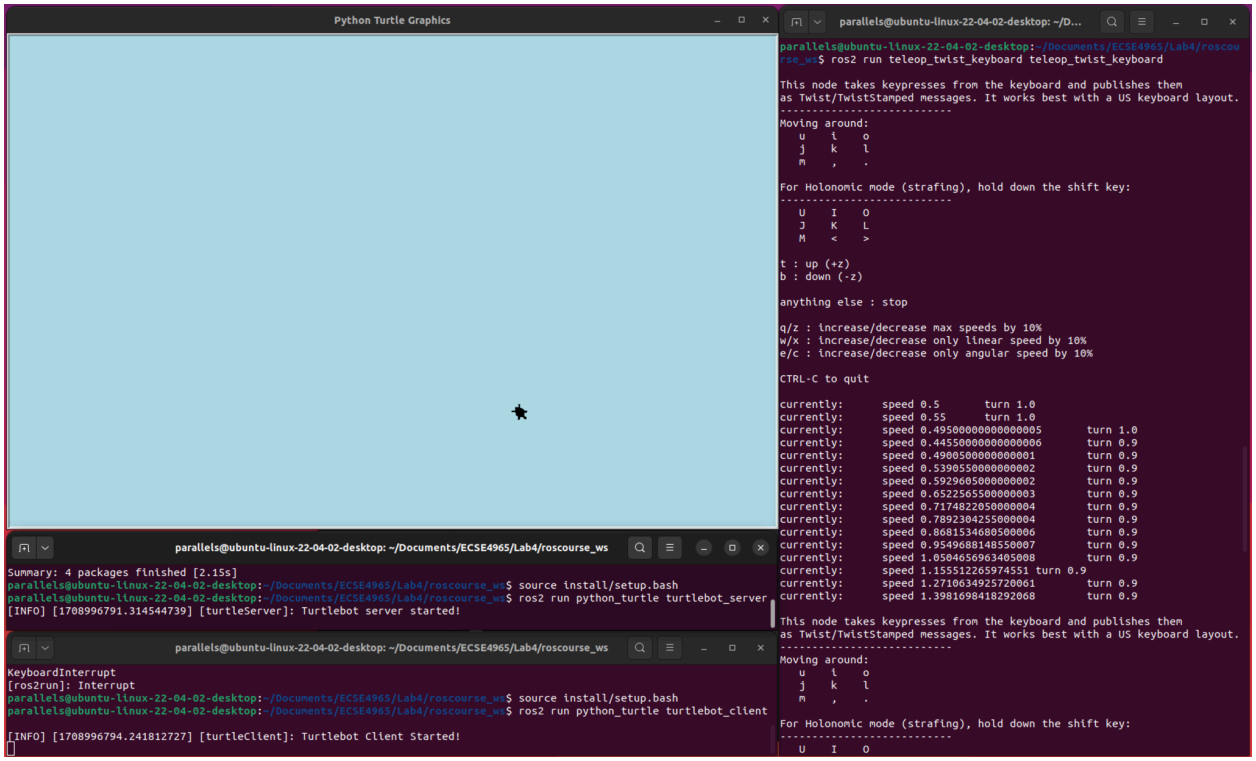
Task 1

• For task 1: Provide the screenshot of the terminals from step 17.

17. Action: In three separate terminals run the turtlebot server, the turtlebot client, and the teleop node. You should now be able to drive your turtlebot using the keyboard.



Task 2

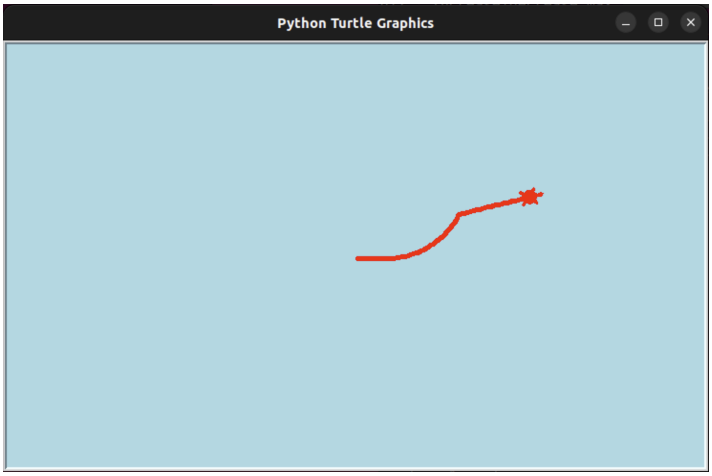
• For task 2: Provide the modified turtlebot_client.py file with appropriate comments and formatting from step 14.

• For task 2: Show the turtle from step 12. It should be a different color than the default parameter value.

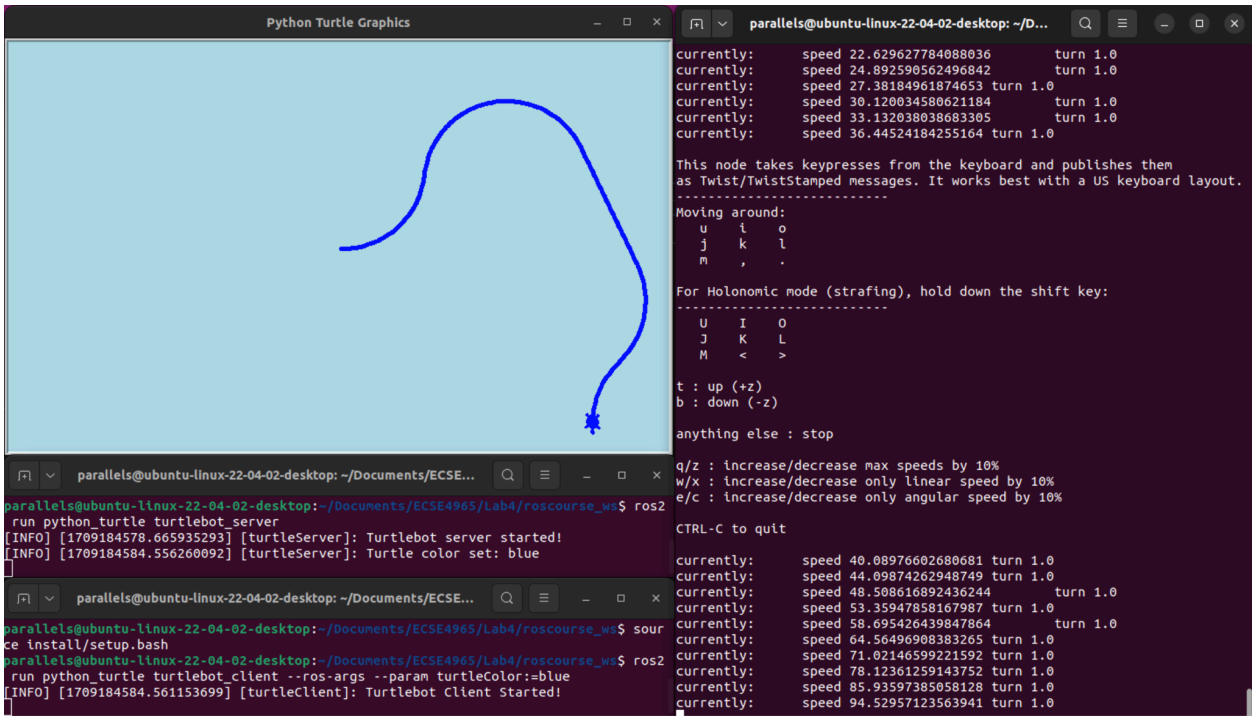
python_turtle > python_turtle > turtlebot_client.py

```python
import rclpy
from rclpy.node import Node
import math
import random

import turtle

from geometry_msgs.msg import Twist, Pose

from rclpy.parameter import Parameter
from rcl_interfaces.msg import ParameterDescriptor

from turtle_interfaces.srv import SetColor
from turtle_interfaces.msg import TurtleMsg

class TurtleClient(Node):
    def __init__(self):
        super().__init__('turtleClient')

        #### Display/Turtle Setup ####
        self.screen = turtle.Screen()
        self.screen.bgcolor('lightblue')
        self.turtle_display = turtle.Turtle()
        self.turtle_display.shape("turtle")
        self.turtle = TurtleMsg()

        #### publisher define ####
        self.twist_pub = self.create_publisher(Twist, 'turtleDrive', 1)
        ##########################

        #### subscribing turtlebot state ####
        self.turtle_sub = self.create_subscription(TurtleMsg, 'turtleState', self.turtle_callback, 1)

        # Pen size
        # Declare a new parameter for pen size with a default value
        self.declare_parameter('pen_size', 5)
        # Retrieve the pen size parameter value
        pen_size = self.get_parameter('pen_size').get_parameter_value().integer_value
        # Set the turtle's pen size
        self.turtle_display.pensize(pen_size)

        # Initial turtle color
        self.declare_parameter('turtleColor', 'red', ParameterDescriptor(description= 'Initial color of the turtle'))
        turtleColor = self.get_parameter('turtleColor').get_parameter_value().string_value
        self.turtle_display.color(turtleColor)

        # Turtle color service
        self.color_cli = self.create_client(SetColor, 'SetColor')
        while not self.color_cli.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('Color service not available, waiting...')
        self.color_req = SetColor.Request()
        self.color_req.color = self.get_parameter('turtleColor').get_parameter_value().string_value
        self.server_call = True
        self.service_future = self.color_cli.call_async(self.color_req)
```

Modified init method of turtlebot_client.py



Default color

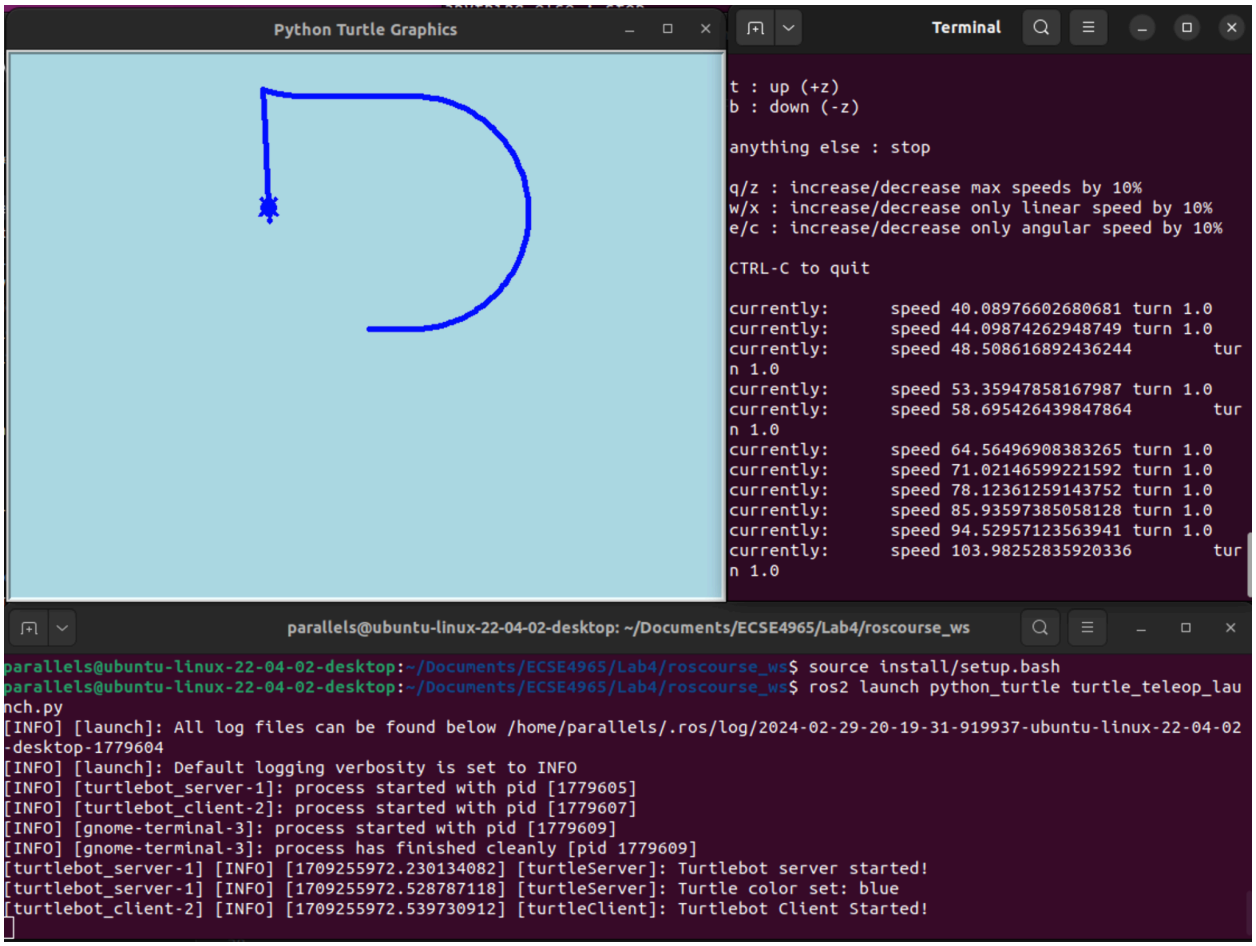


Overrided color

Task 3

• For task 3: Provide a screenshot of the terminals from step 8.

8. Action: Build and source. Now run everything with the command ros2 launch python_turtle turtle_teleop_launch.py



Task 4

• For task 4: Provide a screenshot from the host (the one with all the turtles) and the terminals performing remote control for step 12.