LAB2 DELIVERABLES
Jason Zhai
It contains everything for LAB2
The specified deliverables are highlitghted with large red square

Starting Nodes

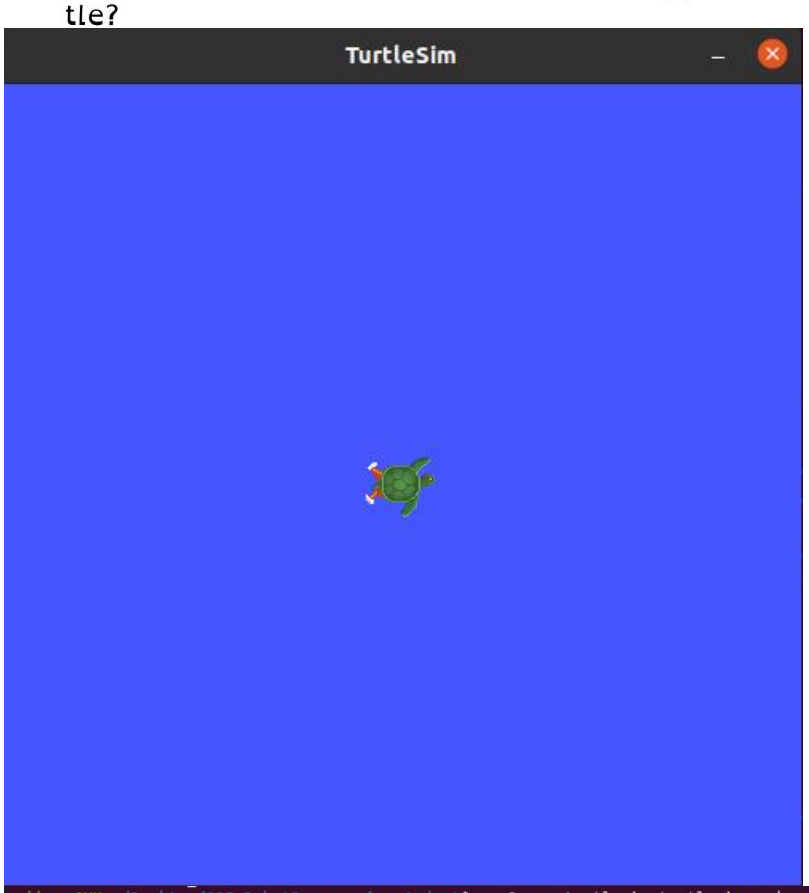
1. **Action:** See which nodes we can execute within this package:
   `ros2 pkg executables turtlesim`

2. **Question:** What are the executable nodes within the turtlesim package?

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

3. **Action:** Start a simulated turtle
   `ros2 run turtlesim turtlesim_node`

4. **Question:** What is the name and starting pose of this simulated turtle?

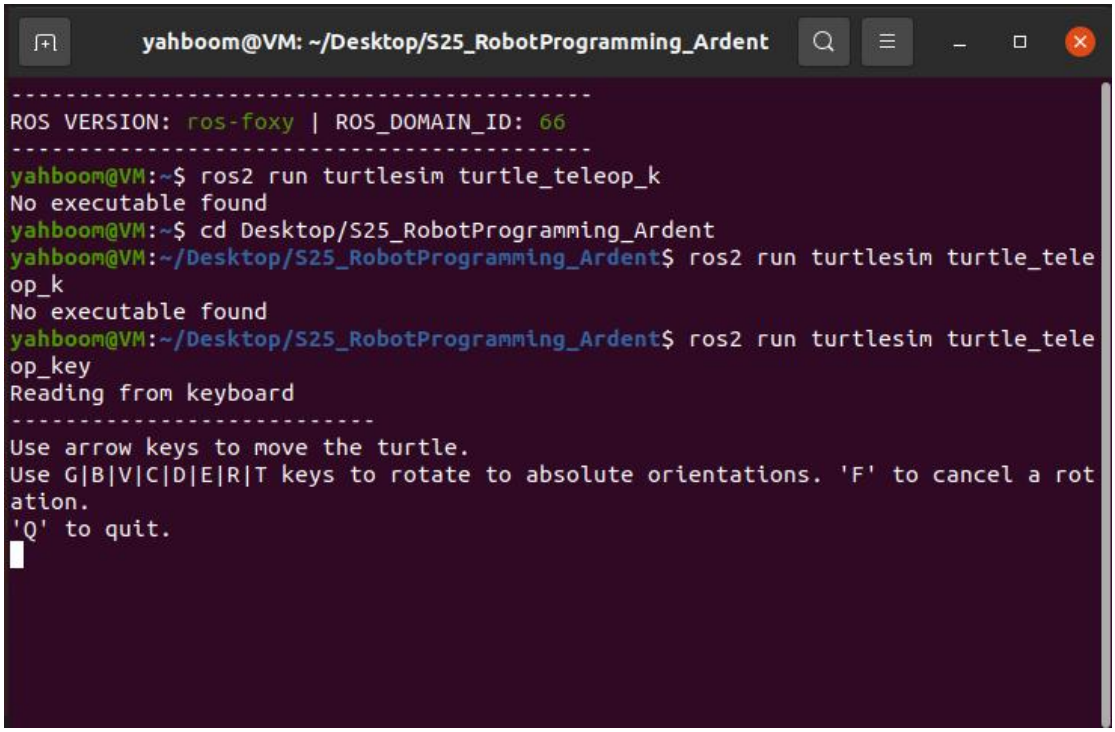


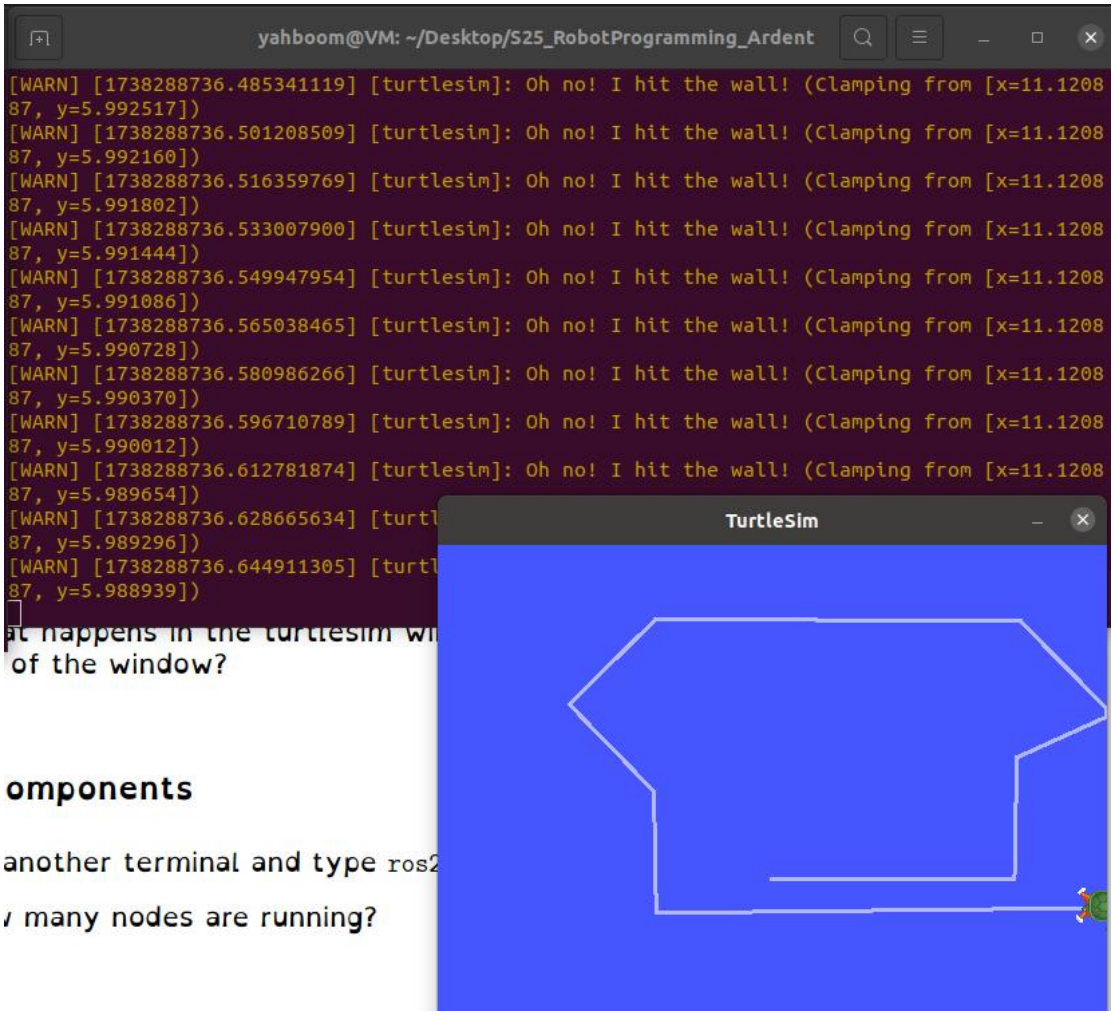

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 run turtlesim turtlesim_node
[INFO] [1738287675.644732765] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1738287675.648759993] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

7. **Action:** Open a new terminal. Position the terminals and simulation window such that you can see all three simultaneously. Start a node to interact with the turtle.
   `ros2 run turtlesim turtle_teleop_key`

8. **Note:** The teleop terminal must be the active terminal to interact with the turtle.

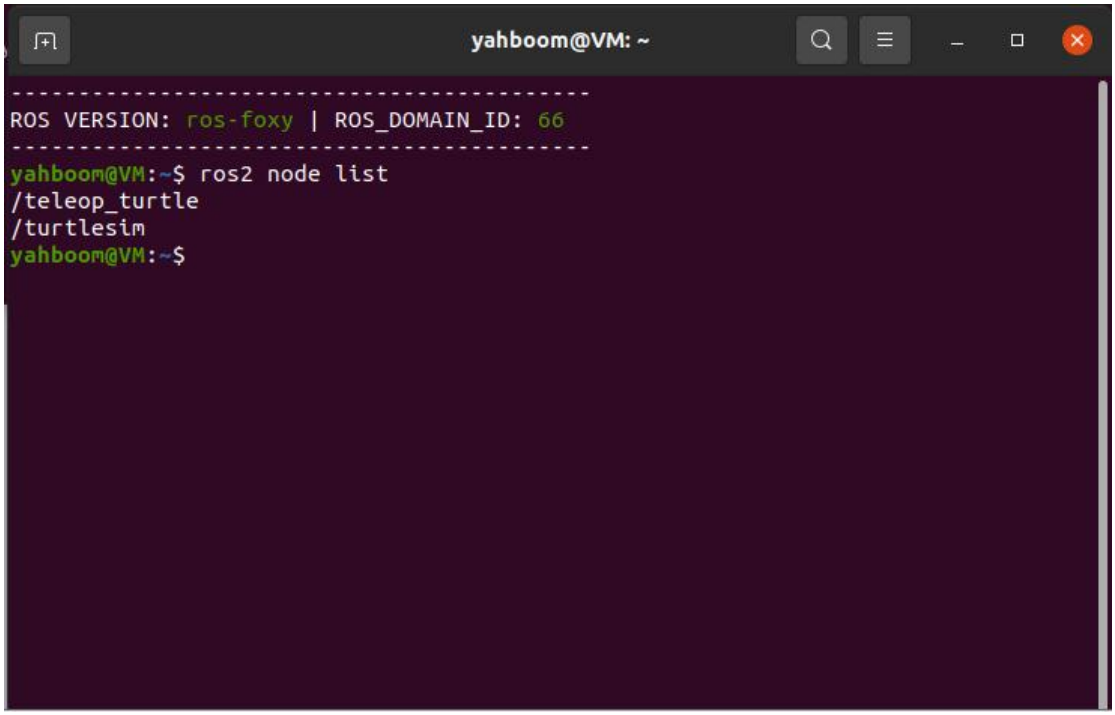
9. **Question:** What happens in the turtlesim window if you try to drive the turtle out of the window?



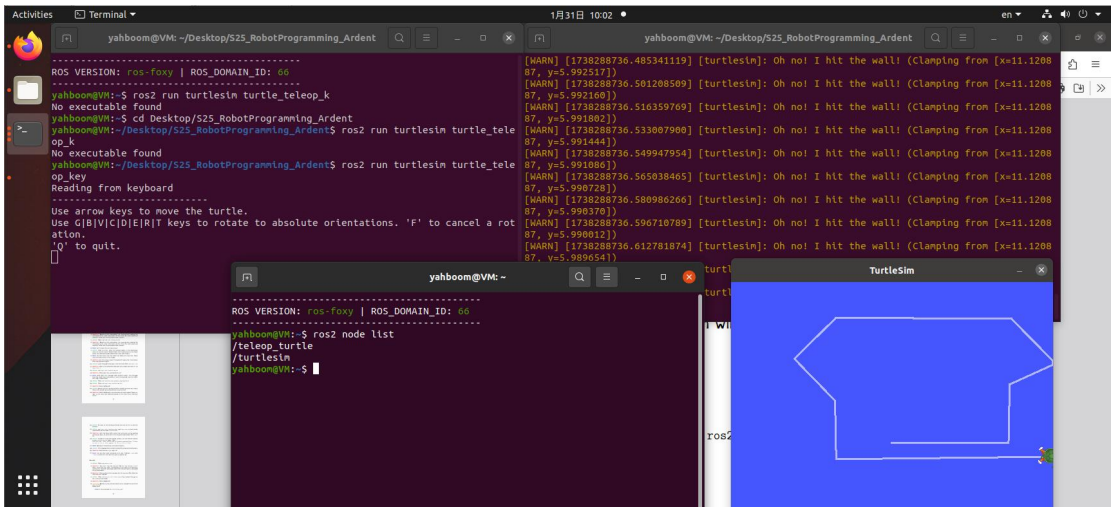


Exploring the Components

1. **Action:** Open another terminal and type `ros2 node list`
2. **Question:** How many nodes are running?



- For starting nodes: Provide a screenshot showing the multiple terminals.



Topics

1. **Action:** Type `ros2 node info /turtlesim`

2. **Note:** Nodes can subscribe or publish to topics. Under the Subscribers/Publishers headings are lists of the form -> topic: message type

3. **Question:** Based on this information, for moving and tracking the simulated turtle, what information do you think this node sends and receives? What are the associated topic names?

```
yahboom@VM:~$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

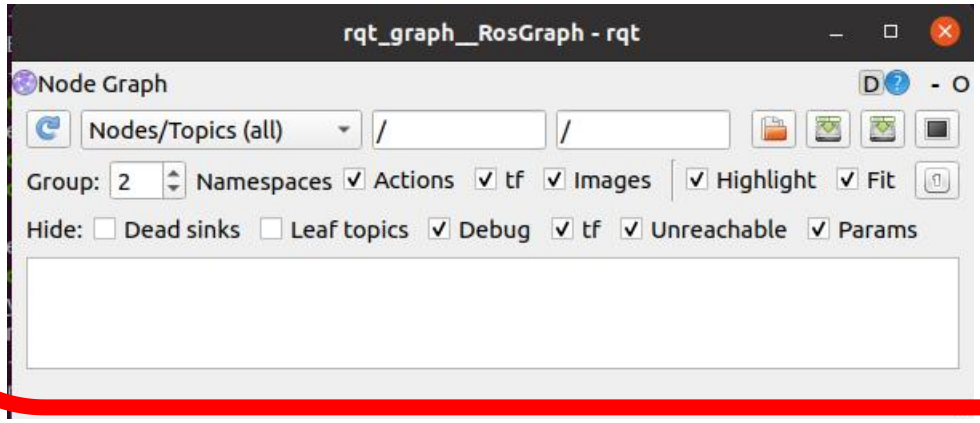
4. **Action:** Type `ros2 node info /teleop_turtle`

5. **Question:** Based on this information, for moving and tracking the simulated turtle, what information do you think this node sends and receives? What are the associated topic names?

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 info /teleop_turtle
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...
ros2: error: argument Call `ros2 <command> -h` for more detailed usage.: invali
d choice: 'info' (choose from 'action', 'bag', 'component', 'daemon', 'doctor',
 'extension_points', 'extensions', 'interface', 'launch', 'lifecycle', 'multica
st', 'node', 'param', 'pkg', 'run', 'security', 'service', 'topic', 'wtf')
```

- For topics: Provide screenshots for steps 7, 21, and 25.

7. **Action:** Type `rqt_graph`. After the window opens, in the drop-down menu on the left select 'Nodes/Topics (all)' and ensure all the check boxes are selected except 'Dead Sinks' and 'Leaf Topics'.
8. **Note:** You can hover over the check box labels for more info. Same with the blocks/lines in the image.
9. **Question:** Do the arrows match the publisher/subscriber information from the previous steps?



10. **Action:** Close the graph and back in the terminal type `ros2 topic list`
11. **Question:** What is the difference between this output and that of `ros2 topic list -t`

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

12. **Action:** `ros2 topic info /turtle1/cmd_vel`
13. **Question:** What does this command tell us?
14. **Note:** What does this message type actually mean? The message tells the nodes how information is sent to/received from the topic. Let's dig in some more.

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 1
```

15. **Action:** Type `ros2 interface show geometry_msgs/msg/Twist`

16. **Action:** Type `ros2 topic echo /turtle1/cmd_vel`

17. **Question:** What happened?

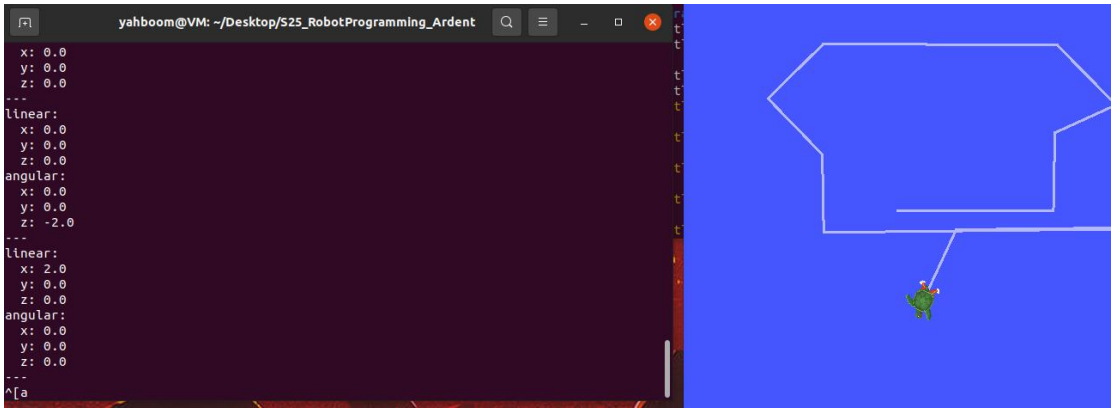
```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 interface show geometry_msgs/m
sg/Twist
# This expresses velocity in free space broken into its linear and angular parts.

Vector3  linear
Vector3  angular
```


```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 interface show geometry_msgs/m
sg/Twist
# This expresses velocity in free space broken into its linear and angular parts.

Vector3  linear
Vector3  angular
```

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic echo /turtle1/cmd_vel
```

18. **Action:** Ensure the this terminal and the teleop terminal are visible. Select the teleop terminal and press an arrow key.

19. **Question:** What happened in the terminal we have recently been using? Is this what was expected based on the result from interface show?



20. **Action:** Go back to the echoing terminal and use Ctrl+C to end the process.

21. **Action:** use `topic info`, `interface show`, and `topic echo` to find similar information for the topic `/turtle1/pose`

22. **Question:** With the pose topic notice that with echo we are getting continuous data. At what rate is the simulator publishing? (Hint: `topic hz`)

```
^[a^Cyahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic info /turtle1 pose
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...
ros2: error: unrecognized arguments: pose
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 interface show /turtle1 pose
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...
ros2: error: unrecognized arguments: pose
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic echo /turtle1 pose
Traceback (most recent call last):
  File "/opt/ros/foxy/bin/ros2", line 11, in <module>
    load_entry_point('ros2cli==0.9.13', 'console_scripts', 'ros2')()
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2cli/cli.py", line 67, in main
    rc = extension.main(parser=parser, args=args)
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2topic/command/topic.py", line 41, in main
    return extension.main(args=args)
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2topic/verb/echo.py", line 71, in main
    return main(args)
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2topic/verb/echo.py", line 88, in main
    message_type = get_message(args.message_type)
  File "/opt/ros/foxy/lib/python3.8/site-packages/rosidl_runtime_py/utilities.py", line 28, in get_message
    interface = import_message_from_namespaced_type(get_message_namespaced_type(identifier))
  File "/opt/ros/foxy/lib/python3.8/site-packages/rosidl_runtime_py/import_message.py", line 30, in import_message_from_namespaced_type
    module = importlib.import_module(
  File "/usr/lib/python3.8/importlib/__init__.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 1011, in _gcd_import
  File "<frozen importlib._bootstrap>", line 950, in _sanity_check
ValueError: Empty module name
```

23. **Action:** Instead of using the teleop window, we can directly publish to the `/turtle1/cmd_vel` topic. Type
    `ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"`

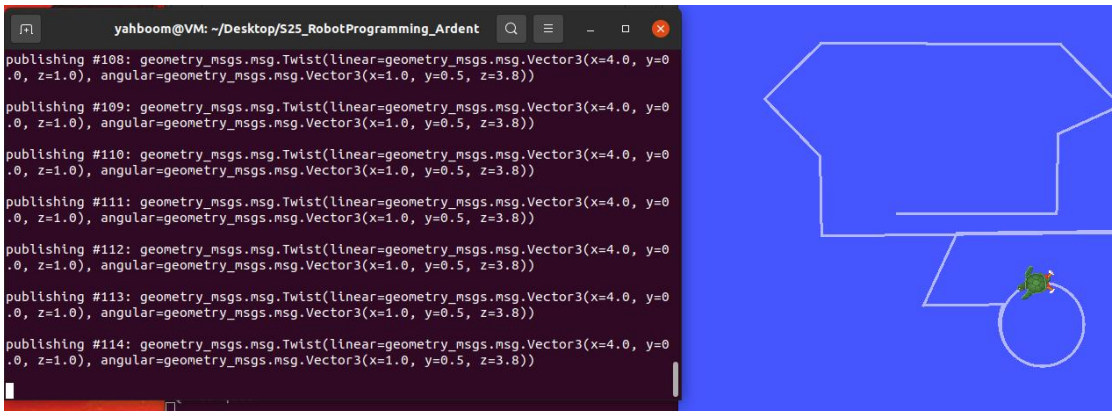
24. **Note:** Spacing in the previous command matters.

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 topic pub --once /turtle1/cmd
_vel geometry_msgs/msg/Twist "{linear:{x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y:
 0.0, z: 1.8}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0
, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.8))
```

25. **Action:** Try changing these numbers and performing some executions.

26. **Question:** What do the x,y,z refer to?

27. **Note:** You can also send commands at a rate. Replace `--once` with `--rate #`, where # is the rate you want to publish at.



Services

1. **Action:** Type `ros2 service list`
2. **Question:** How can I see the service type for the service `/clear`? (Hint: There are two ways. Something we can add to the previous command or another command where the service name is provided to the command.)
3. **Question:** Take a look at the interface for this service type. What do you think this implies?

```
yahboom@VM: ~/Desktop/S25_RobotProgramming_Ardent

^Cyahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service list
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service list -t
/clear [std_srvs/srv/Empty]
/kill [turtlesim/srv/Kill]
/reset [std_srvs/srv/Empty]
/spawn [turtlesim/srv/Spawn]
/teleop_turtle/describe_parameters [rcl_interfaces/srv/DescribeParameters]
/teleop_turtle/get_parameter_types [rcl_interfaces/srv/GetParameterTypes]
/teleop_turtle/get_parameters [rcl_interfaces/srv/GetParameters]
/teleop_turtle/list_parameters [rcl_interfaces/srv/ListParameters]
/teleop_turtle/set_parameters [rcl_interfaces/srv/SetParameters]
/teleop_turtle/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically
]
/turtle1/set_pen [turtlesim/srv/SetPen]
/turtle1/teleport_absolute [turtlesim/srv/TeleportAbsolute]
/turtle1/teleport_relative [turtlesim/srv/TeleportRelative]
/turtlesim/describe_parameters [rcl_interfaces/srv/DescribeParameters]
/turtlesim/get_parameter_types [rcl_interfaces/srv/GetParameterTypes]
/turtlesim/get_parameters [rcl_interfaces/srv/GetParameters]
/turtlesim/list_parameters [rcl_interfaces/srv/ListParameters]
/turtlesim/set_parameters [rcl_interfaces/srv/SetParameters]
/turtlesim/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically]
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service type /clear
std_srvs/srv/Empty
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

4. **Action:** Type `ros2 service call /clear` *`serviceType`* (where you get to fill in the service type)
5. **Question:** What happened?

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /clear std_srvs/
srv/Empty
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()
```

yahboom@VM: ~/Desktop/S25_RobotProgramming_Ardent

```
/turtle1/teleport_relative [turtlesim/srv/TeleportRelative]
/turtlesim/describe_parameters [rcl_interfaces/srv/DescribeParameters]
/turtlesim/get_parameter_types [rcl_interfaces/srv/GetParameterTypes]
/turtlesim/get_parameters [rcl_interfaces/srv/GetParameters]
/turtlesim/list_parameters [rcl_interfaces/srv/ListParameters]
/turtlesim/set_parameters [rcl_interfaces/srv/SetParameters]
/turtlesim/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically]
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service type /clear
std_srvs/srv/Empty
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service type /clear std_srvs/
srv/Empty
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...
ros2: error: unrecognized arguments: std_srvs/srv/Empty
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /clear std_srvs/
srv/Empty
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

3. **Question:** What happened?

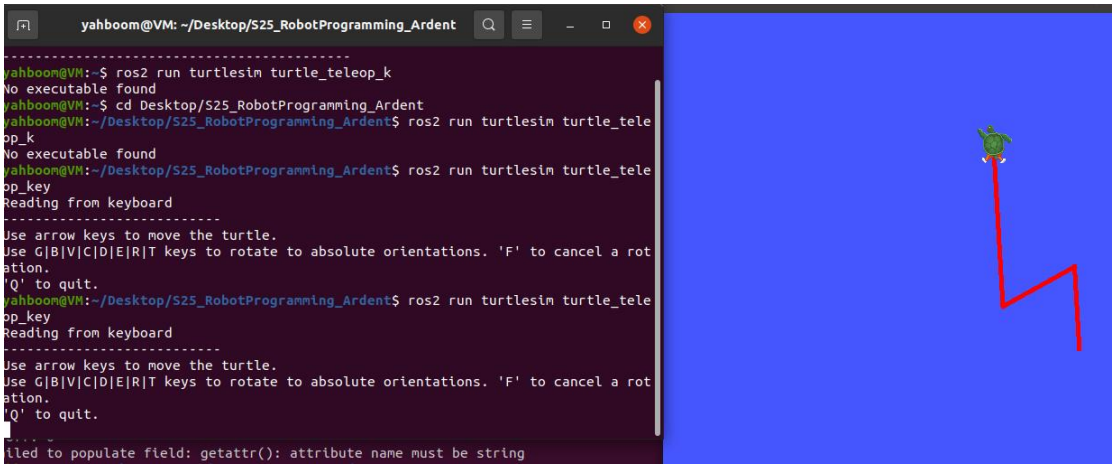
- For services: Provide screenshot for the Challenge.

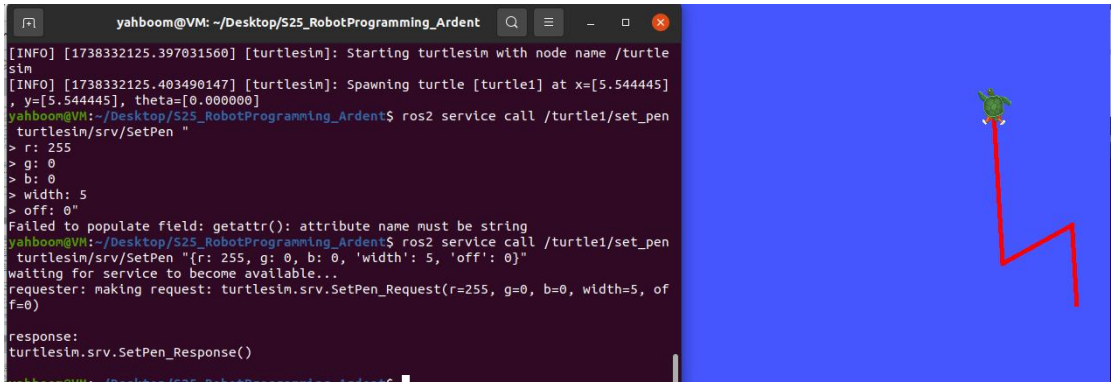
6. **Challenge:** Based on what we have done so far, change the turtle line color to red.
   Some hints:
   - What is the interface for `/turtle1/set_pen`?

- The - - - separates the two data exchanges associated with a service. The first part is the format used to call the service. The second is form of the response from the service.
- When calling a service (similar to publishing to a topic), we have a command of the form
  `ros2 service call` *serviceName serviceType data*
- The data uses YAML format. So we provide the data for this service as "{*attributeName*: *attributeValue*, *nextName*: *nextValue*}" Note that if an attribute name is more than one character, you must surround the name with single quotes so that it is a string data type.

Drive around the turtle using the teleop terminal to see if it worked!

7. **Note:** Note that we called the server part of this service. We can confirm this by looking at the node info for turtlesim again. Note that the set pen service is under the 'Service Servers' list. That means our terminal that called the service was a service client.





1. **Question:** Generate a list of available actions.
2. **Action:** Check the info for each node again.
3. **Question:** Which node is the server and which the client for this action?
4. **Note:** There is another way to get this information.
   Type `ros2 action info` *actionName*
   It lists the clients and servers for you.

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action list
/turtle1/rotate_absolute
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$

yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
```

5. **Action:** Let's look at the interface for this action.
   Type `ros2 interface show` *actionType*
   - **Note:** What is the action type? Let's use the help functions to find a way to get this.
   - **Action:** Type `ros2 action --help`
   - **Note:** We can see there is no 'type' command like we had for topics and services. Maybe there is something in info?
   - **Action:** Type `ros2 action info -h`
   - **Note:** There is an optional argument that we can use with this command to show the type! Give it a try.

- For action: Execute step 8 with a unique goal position.

6. **Note:** The lines starting with # are comments. The - - - separate the sections for an action. Part 1 is the goal format. The middle is the result that is received when the action server terminates. The bottom is the format of the feedback provided by the server while executing the goal.

7. **Action:** Let's send a goal to this action.

```
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
```

8. **Action:** Rerun this command with the addition of `--feedback` at the end to see the feedback.

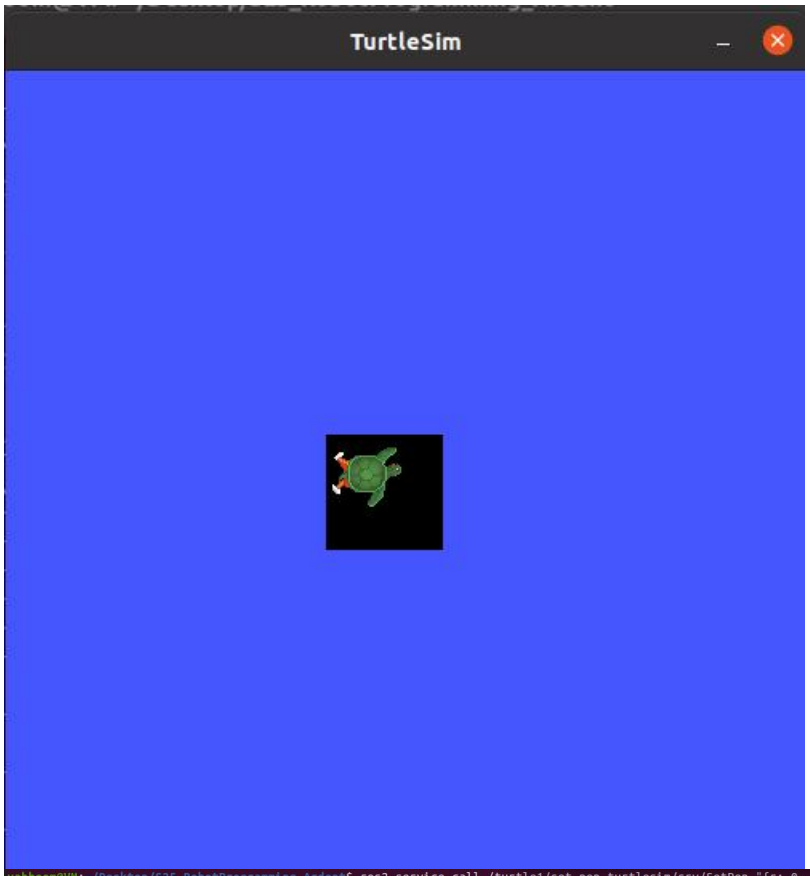
```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action --help
usage: ros2 action [-h]
                   Call `ros2 action <command> -h` for more detailed usage. ...

Various action related sub-commands

optional arguments:
  -h, --help            show this help message and exit

Commands:
  info       Print information about an action
  list       Output a list of action names
  send_goal  Send an action goal

  Call `ros2 action <command> -h` for more detailed usage.
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action info -h
usage: ros2 action info [-h] [-t] [-c] action_name

Print information about an action

positional arguments:
  action_name       Name of the ROS action to get info (e.g. '/fibonacci')

optional arguments:
  -h, --help        show this help message and exit
  -t, --show-types  Additionally show the action type
  -c, --count       Only display the number of action clients and action servers
```

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 interface show turtlesim/action/RotateAbsolute
# The desired heading in radians
float32 theta
---
# The angular displacement in radians to the starting position
float32 delta
---
# The remaining rotation in radians
float32 remaining
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
Waiting for an action server to become available...
Sending goal:
     theta: 1.57

Goal accepted with ID: ebab286fabbf41b28fbeb7e494aa9915

Result:
    delta: 0.04800009727478027

Goal finished with status: SUCCEEDED
```

```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}" --feedback
Waiting for an action server to become available...
Sending goal:
     theta: 1.57

Goal accepted with ID: 309ff3026d1f4757843140f1507245c0

Feedback:
    remaining: -0.01123654842376709

Result:
    delta: 0.0

Goal finished with status: SUCCEEDED
```

Task3

- For task 3: Provide a final image and a text file with a brief explanation.




```
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen "{r: 0, g: 0, b: 0, 'width': 5, 'off': 0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen "{r: 0, g: 0, b: 0, 'width': 50, 'off': 0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 3.0, y: 3.0, theta: 0.0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /clear std_srvs/srv/Empty
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.0, y: 5.0, theta: 0.0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /clear std_srvs/srv/Empty
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.5, y: 5.0, theta: 0.0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.5, y: 5.5, theta: 0.0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.0, y: 5.5, theta: 0.0}"
yahboom@VM:~/Desktop/S25_RobotProgramming_Ardent$
```

To draw a black square in Turtlesim, I first set the pen color to black with the /turtle1/set_pen service. Then I use ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.0, y: 5.0, theta: 0.0}" to reset the position of the turtle. After doing that, I clear the screen using /clear. To form the square, I use the /turtle1/teleport_absolute service, moving the turtle between four precise coordinates, drawing each side. This method ensures accuracy without manual control. I intentionally set the width to 50 to make sure that after the turtle move amaong those four positions, the square can be filled with black color.