6.ROS2 node

1. Node introduction

During communication, no matter which method is used, the construction of communication objects depends on nodes. In ROS2, generally each node corresponds to a single functional module.(For example: a radar driver node may be responsible for publishing radar messages, and a camera driver node may be responsible for publishing image messages).A complete robot system may consist of many nodes working together, and a single executable file (C++ program or Python program) in ROS2 can contain one or more nodes.

2. Create node process

1. Programming interface initialization
2. Create nodes and initialize them
3. Implement node functions
4. Destroy nodes and close interfaces

3. Hello World node case

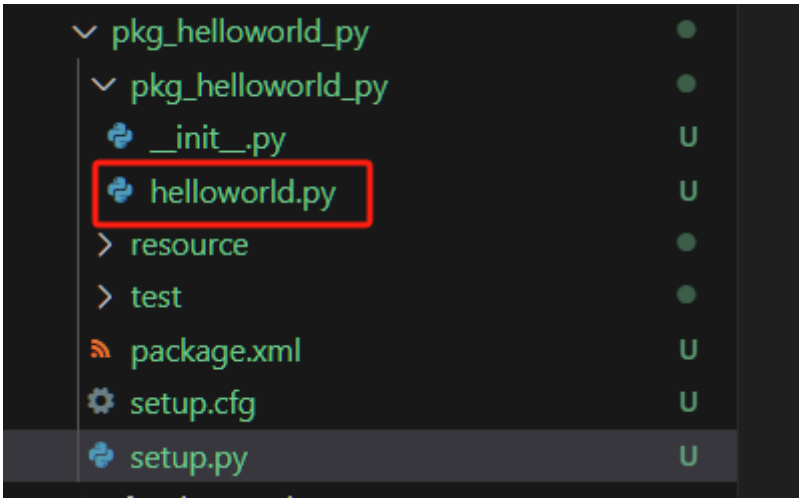
Here we take the python function package as an example to explain

3.1. Create python function package

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws/src
ros2 pkg create pkg_helloworld_py --build-type ament_python --dependencies rclpy
--node-name helloworld
```

3.2. Writing code

After executing the above command, pkg_helloworld_py will be created, and the helloworld.py file will be created to write the node:



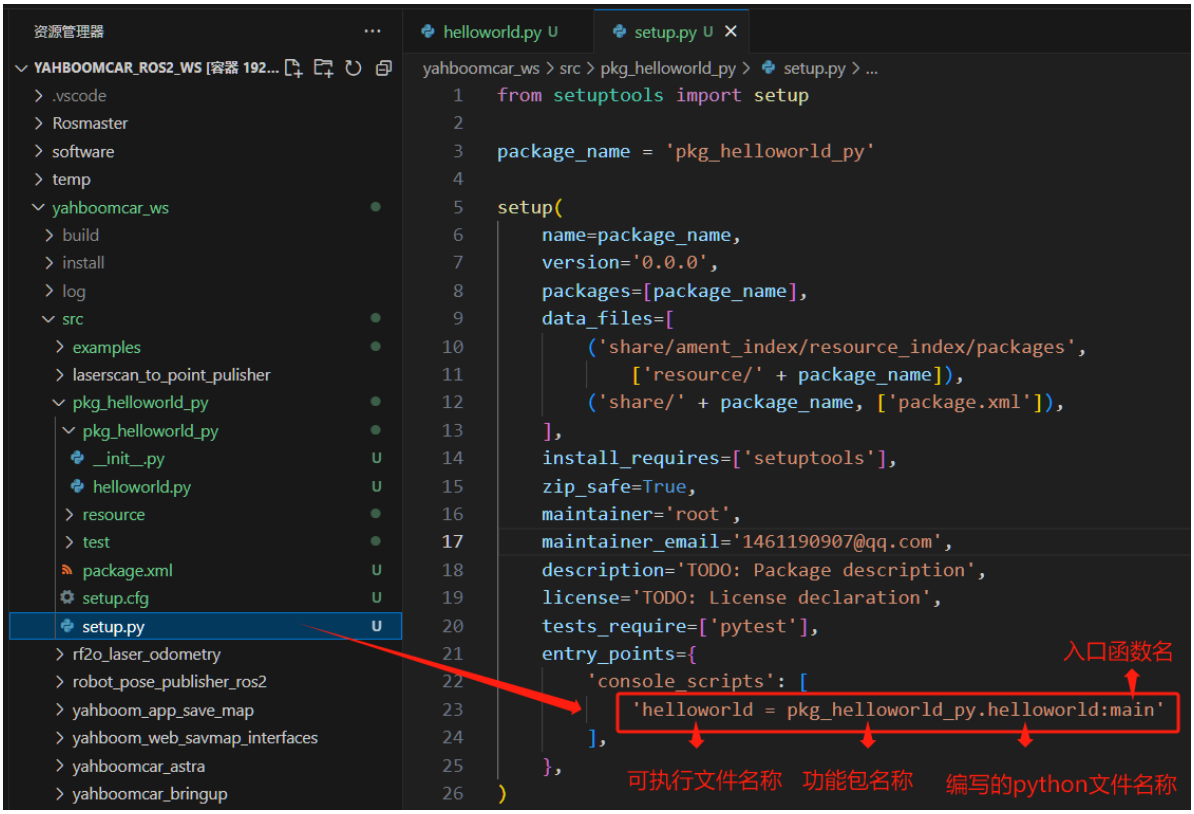
Delete the original helloworld.py code and write the following code:

```
import rclpy                                    # ROS2 Python interface library
from rclpy.node import Node                     # ROS2 Node class
import time

"""
Create a HelloWorld node and output the "hello world" log during initialization
"""
class HelloWorldNode(Node):
    def __init__(self, name):
        super().__init__(name)                  # ROS2 node parent class
initialization
        while rclpy.ok():                       # Whether the ROS2 system is
running normally
            self.get_logger().info("Hello World")  # ROS2 log output
            time.sleep(0.5)                     # sleep control loop time

def main(args=None):                            # ROS2 node main entrance main
function
    rclpy.init(args=args)                       # ROS2 Python interface
initialization
    node = HelloWorldNode("helloworld")         # Create a ROS2 node object
and initialize it
    rclpy.spin(node)                            # Loop waiting for ROS2 to
exit
    node.destroy_node()                         # Destroy node object
    rclpy.shutdown()                            # Close the ROS2 Python
interface
```

After completing the writing of the code, you need to set the compilation options of the function package to let the system know the entrance of the Python program. Open the setup.py file of the function package and add the following entry point configuration:



3.3. Compile function package

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws
colcon build --packages-select pkg_helloworld_py
source install/setup.bash
```

3.4. Run the node

```
ros2 run pkg_helloworld_py helloworld
```

After running successfully, you can see the effect of printing the "Hello World" string in a loop in the terminal:

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_helloworld_py helloworld
[INFO] [1698050077.167925417] [helloworld]: Hello World
[INFO] [1698050077.669687813] [helloworld]: Hello World
[INFO] [1698050078.170830157] [helloworld]: Hello World
[INFO] [1698050078.672503205] [helloworld]: Hello World
[INFO] [1698050079.174207487] [helloworld]: Hello World
[INFO] [1698050079.675991674] [helloworld]: Hello World
[INFO] [1698050080.177530190] [helloworld]: Hello World
[INFO] [1698050080.679924285] [helloworld]: Hello World
[INFO] [1698050081.182654103] [helloworld]: Hello World
[INFO] [1698050081.685459571] [helloworld]: Hello World
[INFO] [1698050082.188045608] [helloworld]: Hello World
```