3.1 Written Questions
1. (Python & C++) What is a CMakeList?
Is it related to a make file used for compiling C++ objects?
If yes then what is the difference between the two?

It is the main cmake file to build the package, it can call catkin specific functions and macros.
It reads the package.xml file, and find other catkin packages to access libraries, including directories.
It also exports items for other packages depending on what you need.
Essentially builds rules for catkin, and it is the input to the cmake build system.

It is related to the make file in C++, but ROS's catkin_make is essentially:
catkin = cmake + (other nice have features)
CMakeLists can be viewed as a high-level wrapper of Makefile. In Unix systems, a CMakeList will produce a Makefile.

2. (Python & C++) Are you using CMakeList.txt for Python in ROS?
Is there a executable object being created for Python?

If python language is used, yes cmakelist would be used for the configurations in ros.
No, an executable object is not being created for python, you would have to program it yourself and preferably then design and code your launch files in subsequent projects to aid in the flow and ease of use for users.

3. (Python & C++) In which directory would you run catkin make?

The workspace main directory.

4. (Python & C++) The following commands were used in the tutorial
$ source / opt / ros / kinetic (melodic)/ setup . bash
$ source devel / setup . bash
Why do we need to source setup.bash?
What does it do?
Why do we have to different setup.bash files here and what is there difference?

To get access to these scripts and executables, the ROS environment variables need to be added to the bash session, thus we have to source the following bash file. When you "source" something in bash, it will execute each line of the file as if it were typed into the current shell. The setup.bash file is merely adding environment variables to your path to allow ROS to function. You can add the source setup.bash command to your ~/.bashrc file, so that it will be executed every time that you open a new shell.
ROS Melodic requires the following: $ source /opt/ros/melodic/setup.bash

It's convenient if the ROS environment variables are automatically added to the bash session every time a new shell is launched as well.
For example: $ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
If we source our setup.bash files again with source ~/catkin_ws/devel/setup.bash, we will be able to navigate to X package because the ROS_PACKAGE_PATH will be set:
$ source ~/catkin_ws/devel/setup.bash
$ roscd X

4.3 Written Questions
1. (C++)What is a nodehandle object?
Can we have more than one nodehandle objects in a single node?

A NodeHandle is an object which represents your ROS node. The node handle is the access point for communications with the ROS system (topics, services, parameters).
Yes you can, it defines how names are resolved in the ROS system. One typical use case would be to create a regular and a private NodeHandle:
ros::NodeHandle nh();
ros::NodeHandle pnh("~");

2. (Python) Is there a nodehandle object in python?
What is the significance of rospy.init node()?

Rospy doesn't have an object like in C++'s ros::NodeHandle in C++.
One of the first calls you will likely execute in a rospy program is the call to rospy.init_node(), which initializes the ROS node for the process. You can only have one node in a rospy process, so you can only call rospy.init_node() once.
The two most common invocations for init_node() are:
rospy.init_node('my_node_name')
rospy.init_node('my_node_name', anonymous=True)

If you do need to resolve names relative to the node name (or another name) in Python, you can use rospy.resolve_name(name, caller_id=None), but instead of a dedicated object this just returns a string.

3. (C++)What is ros::spinOnce()?
How is it different from ros::Spin()?

Ros::spin() will not return until the node has been shut down. Thus, you have no control over your program while ROS is spinning. The advantage of ros::spinOnce() is that it will check for callbacks/service calls/etc only as often as you call it. Why is this useful? Being able to control the rate at which ROS updates allows you to do certain calculations at a certain frequency. Important for time sensitive operations.

4. (C++)What is ros::rate()?

ROS Rate is a powerful ROS feature you can use for your control loops – be it for reading a sensor, controlling a motor, it ensures that it runs loops at a desired frequency.

5. (Python) How do you control callbacks in python for the subscribers?
Do you need spin() or spinonce() in python?

Rospy.init_node('listener', anonymous=True)

There is this callback function that you can define like this -

```
def callback(data):
        # insert whatever you want it to do
```

Yes, spin is needed to keep python from exiting until this node is done.

5.1 Written Questions

1. (C++)Why did you include the header file of the message file instead of the message file Itself?

According to ROS mechanisms, any customized ROS messages will be generated into a templated message header file to be included in the C++ code. The message file itself is not part of standard C++ and it cannot be parsed by C++ compilers.

2. (Python & C++) In the documentation of the LaserScan message there was also a data type called Header header. What is that?
Can you also include it in your message file?
What information does it provide?
Include Header in your message file too.

Header provides information on the sequential order, the timestamp, and the frame ID of the instance when the message is published.

6.1 Written Questions

1. (Python & C++)Where does the bag file get saved?
How can you change where it is saved?

Current dir, if need be, you can indicate the directory you desire with the -o flag.

2. (Python & C++)Where will the bag file be saved if you were launching the recording of bagfile record through a launch file?
How can you change where it is saved?

Current dir, can append args=../bagfiles/my_rosbag_recordings.bag to the node in the launch file.