

# Create Package

AI ROBOT

Exported on 12/04/2021

## Table of Contents

<b>1 Hello World .....</b>	<b>4</b>
1.1 워크스페이스의 src 폴더로 이동.....	4
1.2 catkin create pkg hello_world --catkin-deps std_msgs rospy.....	4
1.3 현재 폴더 파일 구조 .....	4
1.4 현재 폴더를 포함해서 sublime text 실행 .....	5
1.5 sublime text의 실행화면 .....	5
1.6 package.xml을 보면 방금 지정한 dependancy .....	5
1.7 CMakeLists.txt에는 빌드 정보가 저장 .....	6
1.8 목표 : turtlesim의 정보를 얻어오자 subscriber .....	6
1.9 Python 코드를 보관할 scripts 폴더 생성 .....	6
1.10 빈파일 생성.....	7
1.11 print_turtle_pos.py 생성 .....	7
1.12 현재 파일 구조.....	7
<b>2 Subscriber 생성 .....</b>	<b>8</b>
2.1 전체 코드 .....	8
2.2 첫 시작 .....	8
2.3 turtlesim의 정보를 담고 있는 msg 중 Pose를 사용.....	8
2.4 Pose의 내용 .....	9
2.5 import.....	9
2.6 노드 초기화 및 subscriber 노드 선언 .....	9
2.7 subscriber 후 실행할 함수 .....	9
2.8 현재 print_turtle_pos.py 파일이 scripts 폴더에 있다.....	10
2.9 scripts 폴더로 이동해서 ls를 해보면 그냥 파일이다. 이게 무슨말?.....	10
2.10 chmod +x print_turtle_pos.py : 실행 권한 추가 .....	10
2.11 워크스페이스로 다시 이동하자 .....	10
2.12 패키지 빌드 : catkin build hello_world .....	11
2.13 빌드 완료 .....	11
2.14 패키지 인식 .....	11
2.15 방법 .....	11

2.16 다시 시도 .....	12
2.17 결과 .....	12
2.18 roscore와 turtlesim_node를 실행하고 .....	12
2.19 나의 첫 subscriber 노드 실행 .....	13
2.20 결과 .....	13
2.21 rqt_graph 확인 .....	13
2.22 이번에는 turtlesim에 명령을 줘보자 - Publisher .....	14
<b>3 Publisher 생성 .....</b>	<b>15</b>
3.1 전체 코드 .....	15
3.2 move_turtle.py 생성 .....	15
3.3 근데 turtlesim을 움직이는 topic은 뭐지? .....	16
3.4 cmd_vel은 geometry_msgs/Twist 데이터 타입이었다 .....	16
3.5 import 모듈 .....	16
3.6 move_turtle() .....	17
3.7 전체 코드 .....	17
3.8 try - except를 사용한 이유 .....	17
3.9 실행 권한 부여 .....	18
3.10 아까 빌드는 했고 python 코드만 하나 추가했으므로 그냥 사용해도 됨 .....	18
3.11 move_turtle.py 실행 .....	18
3.12 뭐 지난번과 같지만 .....	18
3.13 현재 상황 .....	19

# 1 Hello World

## 1.1 워크스페이스의 src 폴더로 이동

```
pw@ros:~$ cd catkin_ws/
pw@ros:~/catkin_ws$ ls
build  devel  logs  src
pw@ros:~/catkin_ws$ cd src/
pw@ros:~/catkin_ws/src$ ls
pw@ros:~/catkin_ws/src$
```

## 1.2 catkin create pkg hello\_world --catkin-deps std\_msgs rospy

```
pw@melodic:~/ws/src$ catkin create pkg hello_world --catkin-deps std_msgs rospy
Creating package "hello_world" in "/home/pw/ws/src"...
Created file hello_world/package.xml
Created file hello_world/CMakeLists.txt
Created folder hello_world/src
Successfully created package files in /home/pw/ws/src/hello_world.
pw@melodic:~/ws/src$
```

- 패키지 이름은 hello\_world
- 의존성은 std\_msgs와 rospy

## 1.3 현재 폴더 파일 구조

```
pw@melodic:~/ws/src$
pw@melodic:~/ws/src$ tree
.
└── hello_world
    ├── CMakeLists.txt
    └── package.xml
    └── src

2 directories, 2 files
pw@melodic:~/ws/src$
```

## 1.4 현재 폴더를 포함해서 sublime text 실행

The terminal window shows the following command sequence:

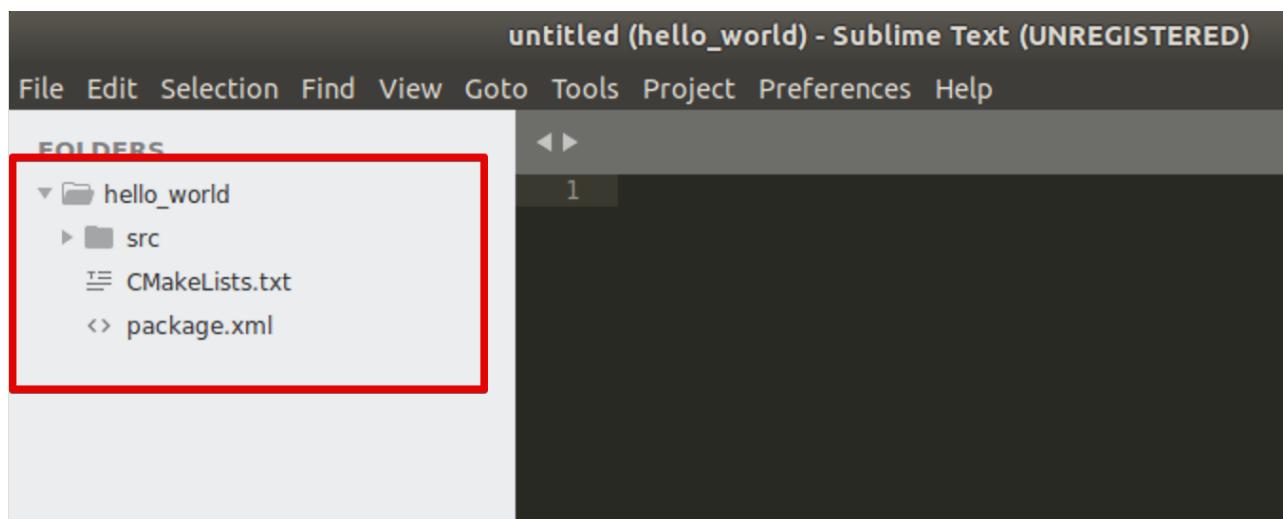
```
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ 
w@melodic:~/ws/src$ cd hello_world/
w@melodic:~/ws/src/hello_world$ subl .
w@melodic:~/ws/src/hello_world$ 
```

The Sublime Text window shows the project structure:

- File menu: File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help
- Folders sidebar: hello\_world (selected), src, CMakeLists.txt, package.xml
- Central view: A large empty area labeled '1'.

- 점(.)을 포함한 것에 주의
- (토막상식) 점 하나(.) 현재 폴더
- (토막상식) 점 두개(..) 상위 폴더

## 1.5 sublime text의 실행화면



## 1.6 package.xml을 보면 방금 지정한 dependancy

The Sublime Text interface shows the package.xml file content:

```
48 <!-- <test_depend>gtest</test_depend> -->
49 <!-- Use doc_depend for packages you need only for building
50 <!-- <doc_depend>doxygen</doc_depend> -->
51 <buildtool_depend>catkin</buildtool_depend>
52 <build_depend>rospy</build_depend>
53 <build_depend>std_msgs</build_depend>
54 <build_export_depend>rospy</build_export_depend>
55 <build_export_depend>std_msgs</build_export_depend>
56 <exec_depend>rospy</exec_depend>
57 <exec_depend>std_msgs</exec_depend>
58
59 
```

The 'package.xml' file in the Folders sidebar is highlighted with a red box. The dependency section from line 51 to 59 is also highlighted with a red box.

- Package manifest : 매니페스트는 패키지에 대한 설명. 패키지, 라이센스, 디펜던시 등의 정보를 제공. package.xml이라는 파일에서 관리
- 특히 여기서 dependency에 대한 정의가 잘 되어 있어야함

## 1.7 CMakeLists.txt에는 빌드 정보가 저장

```

FOLDERS
hello_world
src
CMakeLists.txt
package.xml

cmake_minimum_required(VERSION 2.8.3)
project(hello_world)

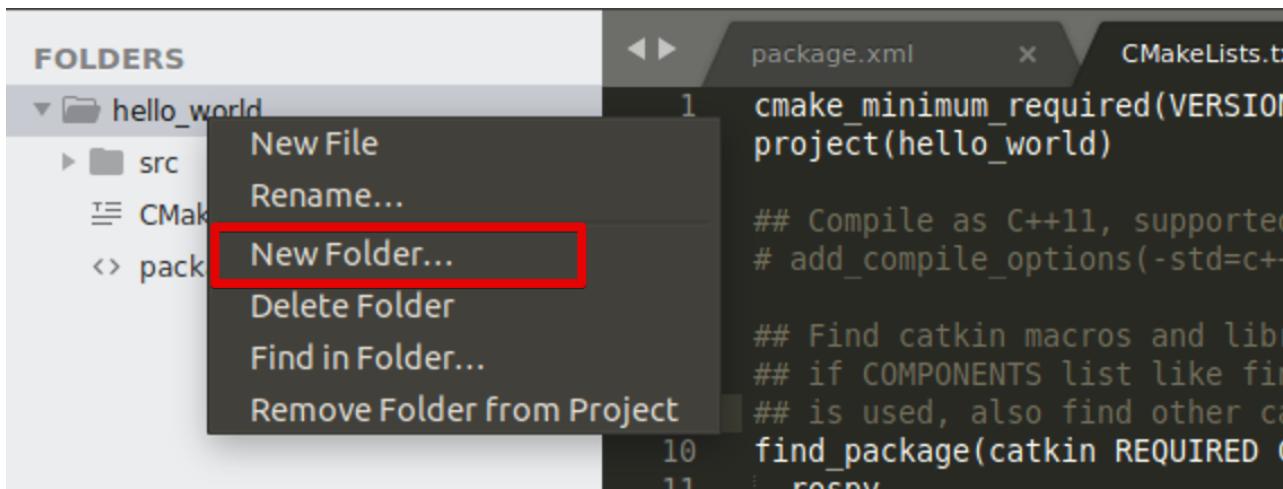
## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS ...
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
rospy
std_msgs
)

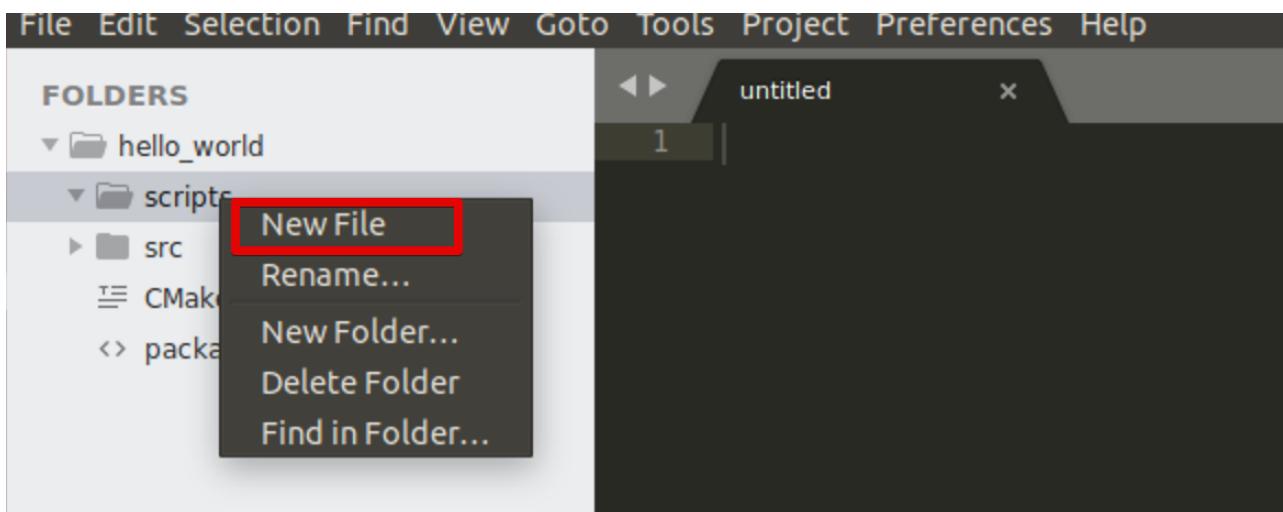
```

## 1.8 목표 : turtlesim의 정보를 얻어오자 subscriber

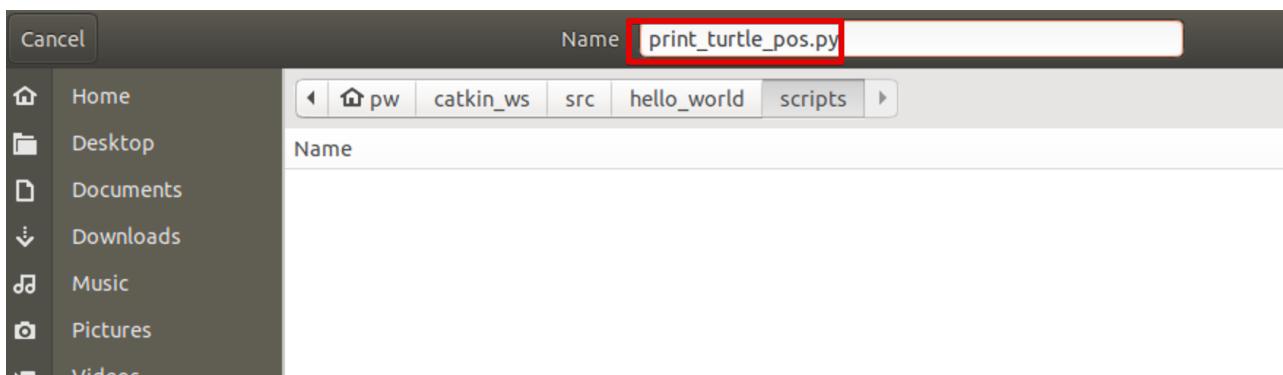
## 1.9 Python 코드를 보관할 scripts 폴더 생성



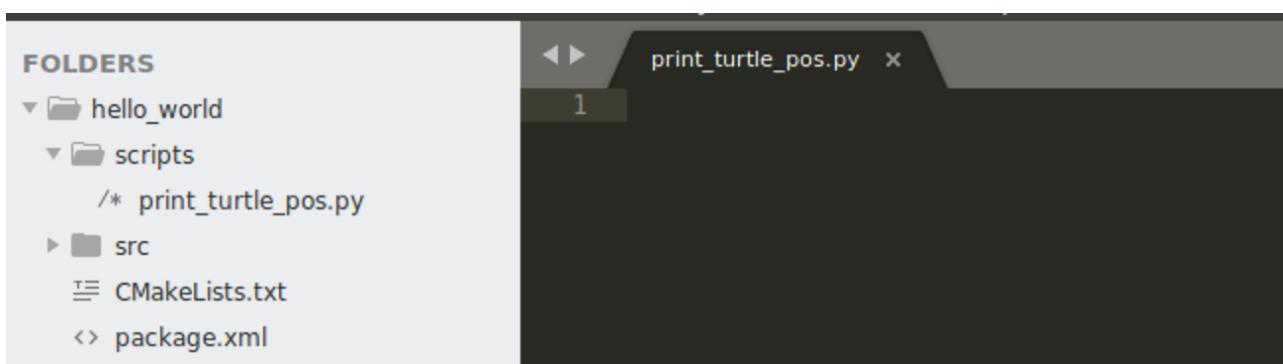
## 1.10 빈파일 생성



## 1.11 print\_turtle\_pos.py 생성



## 1.12 현재 파일 구조



## 2 Subscriber 생성

### 2.1 전체 코드

```

1  #!/usr/bin/env python
2
3  import rospy
4  from turtlesim.msg import Pose
5
6  def print_turtle_pos():
7
8      rospy.init_node('print_turtle_pos', anonymous=False)
9
10     rospy.Subscriber("/turtle1/pose", Pose, callback)
11     rospy.spin()
12
13 def callback(data):
14     rospy.loginfo("Turtle Pose X : %s, Y : %s", data.x, data.y)
15
16 if __name__ == '__main__':
17     print_turtle_pos()

```

### 2.2 첫 시작

```
#!/usr/bin/env python
```

- Python으로 만들어진 ROS내의 모든 코드는 위 줄로 시작해야함
- Python 환경을 지정하는 명령

### 2.3 turtlesim의 정보를 담고 있는 msg 중 Pose를 사용

```

pw@ros:~$ rosmsg package turtlesim
turtlesim/Color
turtlesim/Pose

```

## 2.4 Pose의 내용

```
pw@ros:~$ rosmsg show turtlesim/Pose
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

## 2.5 import

```
import rospy
from turtlesim.msg import Pose
```

- ros 명령을 사용하기 위해 rospy import
- turtlesim의 msg에서 Pose를 import

## 2.6 노드 초기화 및 subscriber 노드 선언

```
def print_turtle_pos():

    rospy.init_node('print_turtle_pos', anonymous=False)

    rospy.Subscriber("/turtle1/pose", Pose, callback)
    rospy.spin()
```

- 노드를 초기화(init\_node)한다
- init\_node에 들어간 이름이 내가 만든 이름이 된다.
- anonymous 플래그는 같은 이름의 노드가 혼선을 부를 수 있어서 True로 해두면 노드이름뒤에 유니크한 ID를 붙여준다
- subscriber는 토픽이 발행될때마다 받아야하니 항상 대기해야한다 → spin
- Subscriber 명령에서 받을 토픽이름을 지정하고, 메세지 형을 지정하고, 토픽을 잘 받았을때 실행할 함수명을 선언한다.

## 2.7 subscriber 후 실행할 함수

```
def callback(data):
```

```
rospy.loginfo("Turtle Pose X : %s, Y : %s", data.x, data.y)
```

- loginfo 명령은
  - 콘솔창에 print 문처럼 메시지를 출력하고
  - 디버그때 사용하는 rosout에 출력을 기록한다
- turtlesim/Pose에서 x, y만 사용

## 2.8 현재 print\_turtle\_pos.py 파일이 scripts 폴더에 있다

```
pw@ros:~/catkin_ws/src/hello_world/
pw@ros:~/catkin_ws/src/hello_world$ tree
.
├── CMakeLists.txt
├── package.xml
└── scripts
    └── print_turtle_pos.py
src

2 directories, 3 files
```

## 2.9 scripts 폴더로 이동해서 ls를 해보면 그냥 파일이다. 이게 무슨말?

```
pw@ros:~/catkin_ws/src/hello_world/scripts$ 
pw@ros:~/catkin_ws/src/hello_world/scripts$ ls
print_turtle_pos.py
pw@ros:~/catkin_ws/src/hello_world/scripts$
```

## 2.10 chmod +x print\_turtle\_pos.py : 실행 권한 추가

```
pw@ros:~/catkin_ws/src/hello_world/scripts$ 
pw@ros:~/catkin_ws/src/hello_world/scripts$ chmod +x print_turtle_pos.py
pw@ros:~/catkin_ws/src/hello_world/scripts$ ls
print_turtle_pos.py
pw@ros:~/catkin_ws/src/hello_world/scripts$
```

## 2.11 워크스페이스로 다시 이동하자

```
pw@ros:~/catkin_ws/src/hello_world/scripts$ 
pw@ros:~/catkin_ws/src/hello_world/scripts$ cd ~/catkin_ws/
pw@ros:~/catkin_ws$
```

## 2.12 패키지 빌드 : catkin build hello\_world

```
pw@ros:~/catkin_ws$  
pw@ros:~/catkin_ws$ catkin build hello_world
```

- 다 타이핑하지 말고 적절히 텁(<TAB>)을 이용하자

## 2.13 빌드 완료

```
[build] Found '1' packages in 0.0 seconds.  
[build] Updating package table.  
Starting >>> hello_world  
Finished <<< hello_world [ 2.0 seconds ]  
[build] Summary: All 1 packages succeeded!  
[build] Ignored: None.  
[build] Warnings: None.  
[build] Abandoned: None.  
[build] Failed: None.  
[build] Runtime: 2.1 seconds total.  
[build] Note: Workspace packages have changed, please re-source setup files to use them.  
pw@ros:~/catkin_ws$
```

## 2.14 패키지 인식

```
[build] Failed: None.  
[build] Runtime: 2.1 seconds total.  
[build] Note: Workspace packages have changed, please re-source setup files to use them.  
pw@ros:~/catkin_ws$  
pw@ros:~/catkin_ws$  
pw@ros:~/catkin_ws$  
pw@ros:~/catkin_ws$ roscl he
```

- roscl he ← 여기까지 입력하고 <TAB>을 눌러보자 자동완성이 되지 않는다면.. 패키지를 인식하지 못했다
- 빌드도 했는데..ㅠㅠ.
- 이유는 ~/catkin\_ws/devel/setup.bash를 읽지 못했기 때문

## 2.15 방법

```
pw@melodic:~/ws$  
pw@melodic:~/ws$ source ~/.bashrc  
ROS1 is activated  
pw@melodic:~/ws$  
pw@melodic:~/ws$
```

- 터미널을 다시 실행하던지 혹은
- source ~/.bashrc를 실행하면 된다

## 2.16 다시 시도

```
pw@ros:~/catkin_ws$  
pw@ros:~/catkin_ws$ roscl hello_world/
```

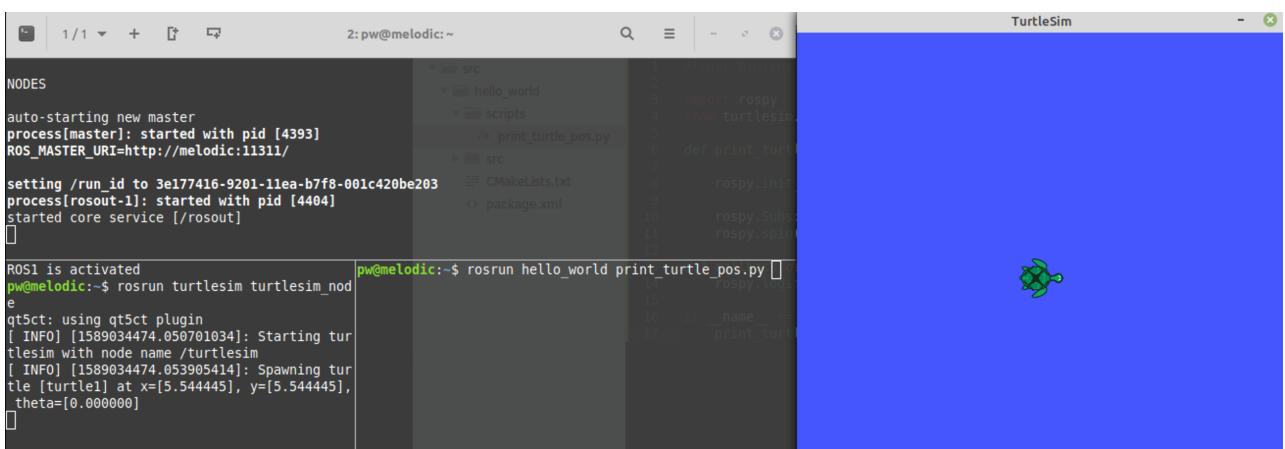
- 이제 roscl he 쯤에서 <TAB>을 입력하면 자동 완성이 될 것이다

## 2.17 결과

```
pw@melodic:~/ws$  
pw@melodic:~/ws$ roscl hello_world/  
pw@melodic:~/ws/src/hello_world$ tree  
. . .  
└── CMakeLists.txt  
└── package.xml  
└── scripts  
    └── print_turtle_pos.py  
└── src  
  
2 directories, 3 files  
pw@melodic:~/ws/src/hello_world$
```

```
10   rospy.Subscribe  
11   rospy.spin  
12  
13   def callback(  
14       rospy.loginfo
```

## 2.18 roscore와 turtlesim\_node를 실행하고



## 2.19 나의 첫 subscriber 노드 실행

```

ROS1 is activated
pw@melodic:~$ rosrun turtlesim turtlesim_node
[ INFO] [1589034474.050701034]: Starting turtlesim with node name /turtlesim
[ INFO] [1589034474.053905414]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[ INFO] [1589034620.128839]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.144227]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.160846]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.177371]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.192264]: Turtle Pose X : 5.544444561, Y : 5.544444561

```

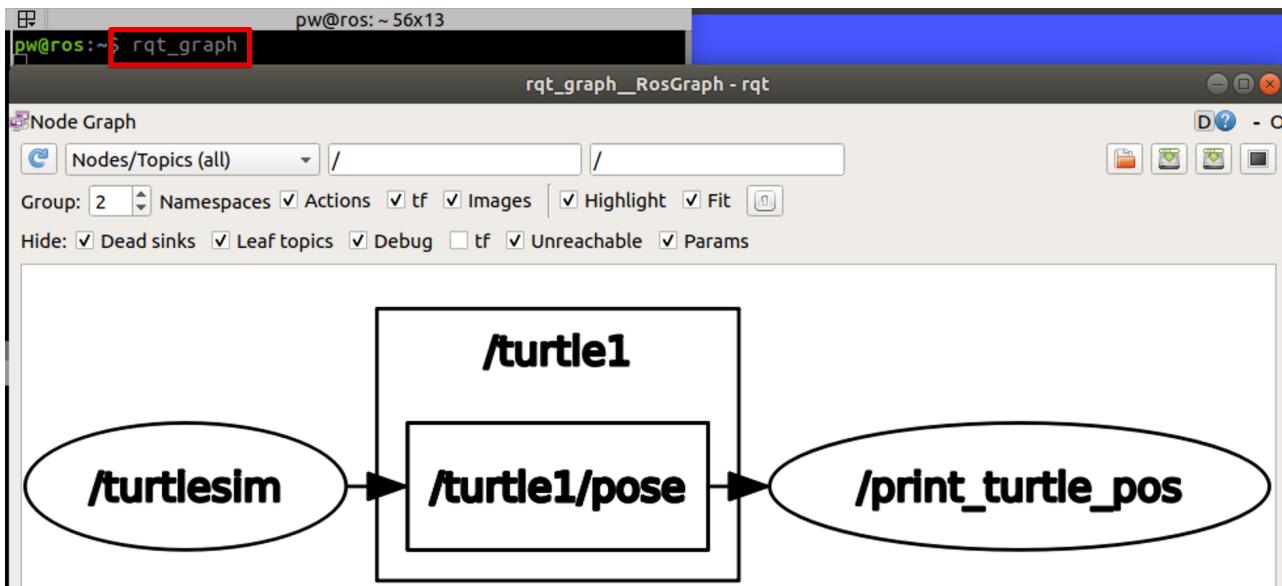
## 2.20 결과

```

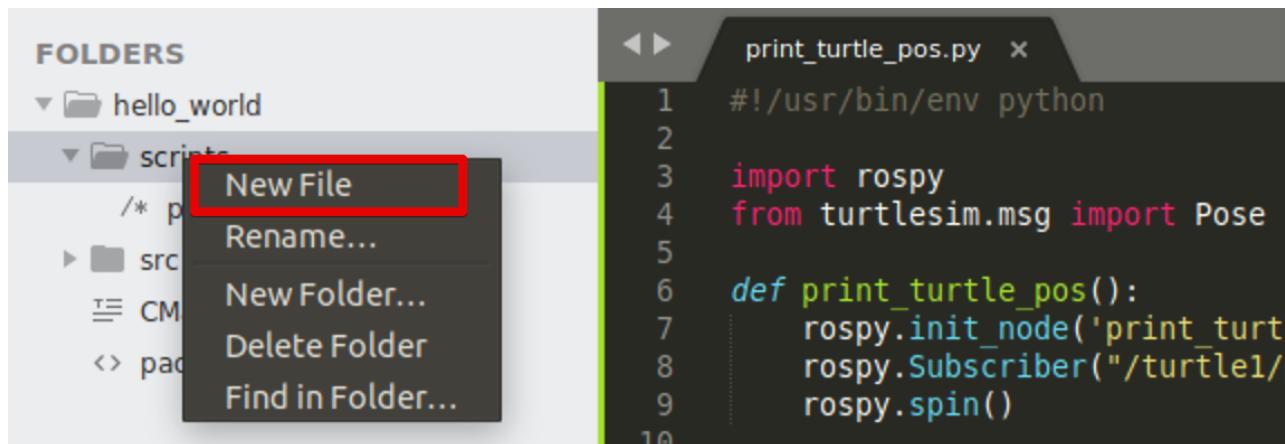
ROS1 is activated
pw@melodic:~$ rosrun turtlesim turtlesim_node
[ INFO] [1589034474.050701034]: Starting turtlesim with node name /turtlesim
[ INFO] [1589034474.053905414]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[ INFO] [1589034620.128839]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.144227]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.160846]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.177371]: Turtle Pose X : 5.544444561, Y : 5.544444561
[ INFO] [1589034620.192264]: Turtle Pose X : 5.544444561, Y : 5.544444561

```

## 2.21 rqt\_graph 확인



## 2.22 이번에는 turtlesim에 명령을 줘보자 - Publisher



## 3 Publisher 생성

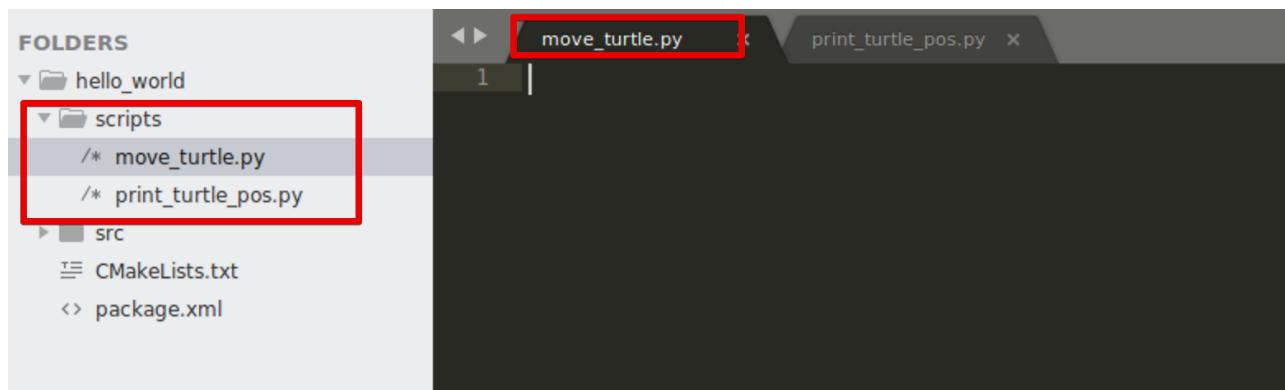
### 3.1 전체 코드

```

1  #!/usr/bin/env python
2
3  import rospy
4  from geometry_msgs.msg import Twist
5
6  def move_turtle():
7      rospy.init_node('move_turtle', anonymous=False)
8      vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
9
10     vel_ref = Twist()
11
12     rate = rospy.Rate(10) # 10hz
13
14     while not rospy.is_shutdown():
15         vel_ref.linear.x = 2
16         vel_ref.angular.z = 2
17         rospy.loginfo(vel_ref)
18         vel_pub.publish(vel_ref)
19         rate.sleep()
20
21 if __name__ == '__main__':
22
23     try:
24         move_turtle()
25     except rospy.ROSInterruptException:
26         pass

```

### 3.2 move\_turtle.py 생성



### 3.3 근데 turtlesim을 움직이는 topic은 뭐지?

```
pw@melodic:~$ rostopic list -v
Published topics:
* /turtle1/color_sensor [turtlesim/Color] 1 publisher
* /rosout [rosgraph_msgs/Log] 1 publisher
* /rosout_agg [rosgraph_msgs/Log] 1 publisher
* /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
* /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
* /rosout [rosgraph_msgs/Log] 1 subscriber
```

### 3.4 cmd\_vel은 geometry\_msgs/Twist 데이터 타입이었다

```
pw@ros:~$ rostopic type /turtle1/cmd_vel | rosmsg show
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

### 3.5 import 모듈



```
print_turtle_pos.py  x  move_turtle.py  x
1 #!/usr/bin/env python
2
3 import rospy
4 from geometry_msgs.msg import Twist
```

- turtle1/cmd\_vel 사용하는 데이터형인 geometry\_msgs/Twist를 import

### 3.6 move\_turtle()

```

6  def move_turtle():
7      rospy.init_node('move_turtle', anonymous=False)
8      vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
9
10     vel_ref = Twist()
11
12     rate = rospy.Rate(10) # 10hz
13
14     while not rospy.is_shutdown():
15         vel_ref.linear.x = 2
16         vel_ref.angular.z = 2
17         rospy.loginfo(vel_ref)
18         vel_pub.publish(vel_ref)
19         rate.sleep()

```

- turtle1/cmd\_ver 토픽에 Twist 데이터형으로 선언함
- queue\_size는 publish한 데이터를 받지 못할때, 계속 데이터를 쌓아둘 수 없어서 그 양을 제한하는 옵션

### 3.7 전체 코드

```

#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

def move_turtle():
    rospy.init_node('move_turtle', anonymous=False)
    vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

    vel_ref = Twist()

    rate = rospy.Rate(10) # 10hz

    while not rospy.is_shutdown():
        vel_ref.linear.x = 2
        vel_ref.angular.z = 2
        rospy.loginfo(vel_ref)
        vel_pub.publish(vel_ref)
        rate.sleep()

if __name__ == '__main__':
    try:
        move_turtle()
    except rospy.ROSInterruptException:
        pass

```

### 3.8 try - except를 사용한 이유

- rospy.ROSInterruptException 예외가 발생하는 이유는 sleep () 후에 실수로 코드를 계속 실행하지 않기 때문

### 3.9 실행 권한 부여

```
pw@melodic:~/ws/src/hello_world/scripts$ chmod +x move_turtle.py
pw@melodic:~/ws/src/hello_world/scripts$ ls
move_turtle.py  print_turtle_pos.py
pw@melodic:~/ws/src/hello_world/scripts$
```

### 3.10 아까 빌드는 했고 python 코드만 하나 추가했으므로 그냥 사용해도 됨

### 3.11 move\_turtle.py 실행

```
ROS1 is activated
pw@melodic:~$ rosrun turtlesim turtlesim_node
[INFO] [1589034474.050701034]: Starting turtlesim with node name /turtlesim
[INFO] [1589034474.053905414]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[INFO] [1589034975.235371]: linear:
  x: 2
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2
[INFO] [1589034975.235371]: linear:
  x: 2
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2
```

### 3.12 뭐 지난번과 같지만



x: 2  
y: 0.0  
z: 0.0  
angular:  
 x: 0.0  
 y: 0.0  
 z: 2  
[INFO] [1574346044.830017]: linear:  
 x: 2  
 y: 0.0  
 z: 0.0  
angular:  
 x: 0.0  
 y: 0.0  
 z: 2

### 3.13 현재 상황

