

ROS 2

[04 – ROS2 with Python]

한국폴리텍대학교 성남캠퍼스

1 Jupyter 설치 및 기초

❖ ROS2와 Python

- ROS2는 Node, Package의 설계, 메시지 제작이 주요 해야 할 작업
- C++, Python 등으로 해당 기능들을 구현해야 함
- 이번 강의에서는 ROS를 Python으로 다루는 점에 집중
- Python 프로그래머가 자주 활용하는 Jupyter 개발환경을 이용해서 ROS2를 다루는 Python 명령에 대해 알아보기로 함

❖ Python 및 Jupyter 설치

```
$ sudo apt install python3-pip
```

- pip는 Python에서 모듈을 관리하는 관리자
- pip를 이용하면 Python 모듈을 손쉽게 설치, 제거할 수 있음

```
$ pip3 install --upgrade pip
```

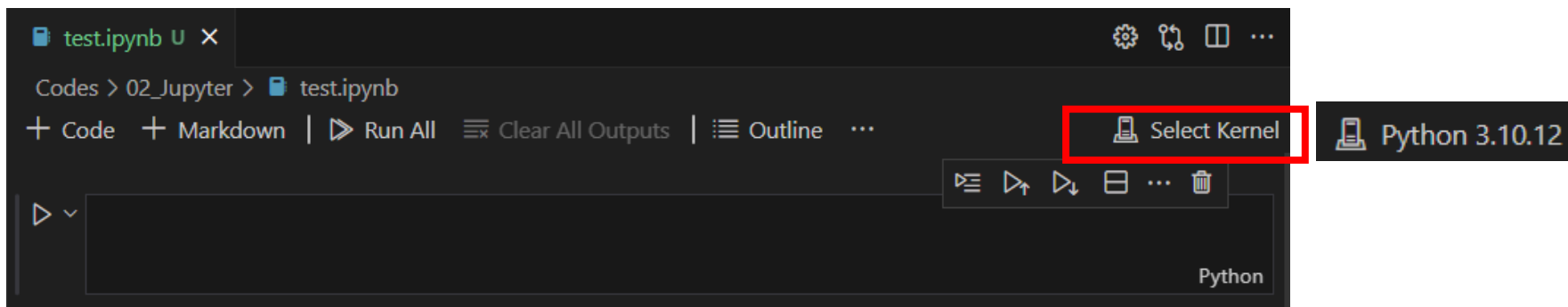
```
$ pip3 install Jupyter ipywidgets pyyaml bqplot
```

❖ Jupyter 기본 사용

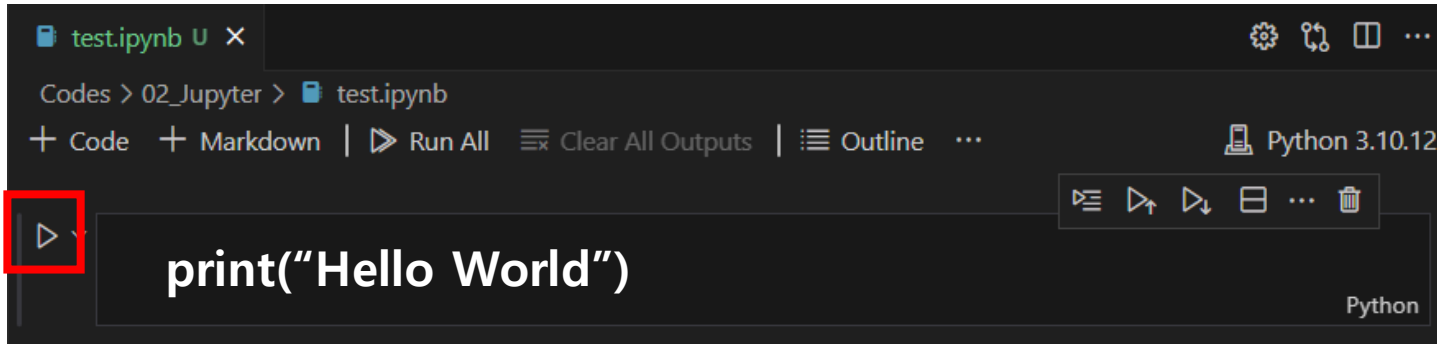
- 원하는 위치에 Jupyter 파일을 저장하기 위한 폴더 만들기
- 대부분 User home에 폴더 생성
- 폴더 생성 후 생성한 폴더로 이동 후 VS-Code 실행

```
$ cd ~  
$ mkdir ( folder_name )  
$ cd ( folder_name )  
$ code
```

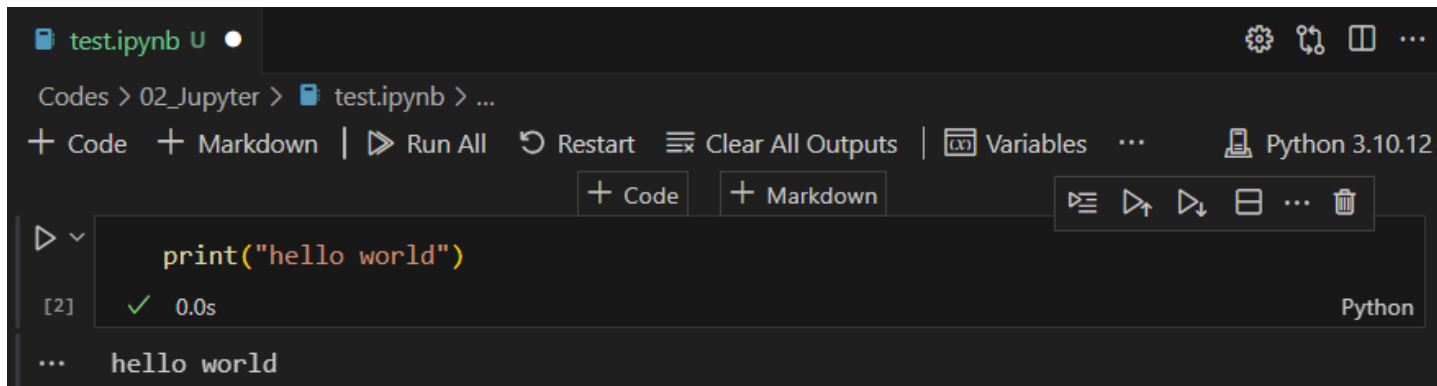
- VS-Code에서 ~~~.ipynb 파일 생성



❖ Jupyter Code 블록

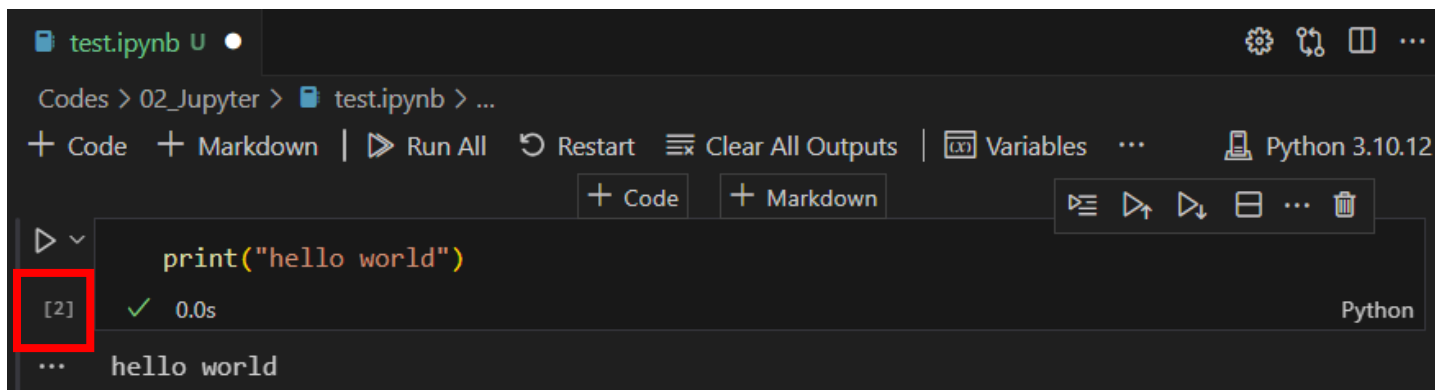


- 좌측에 실행버튼이 있는 공간이 코드를 작성할 수 있는 공간
- 코드를 작성한 후 좌측의 실행 버튼을 누르면 해당 코드 실행



❖ Jupyter Code 블록

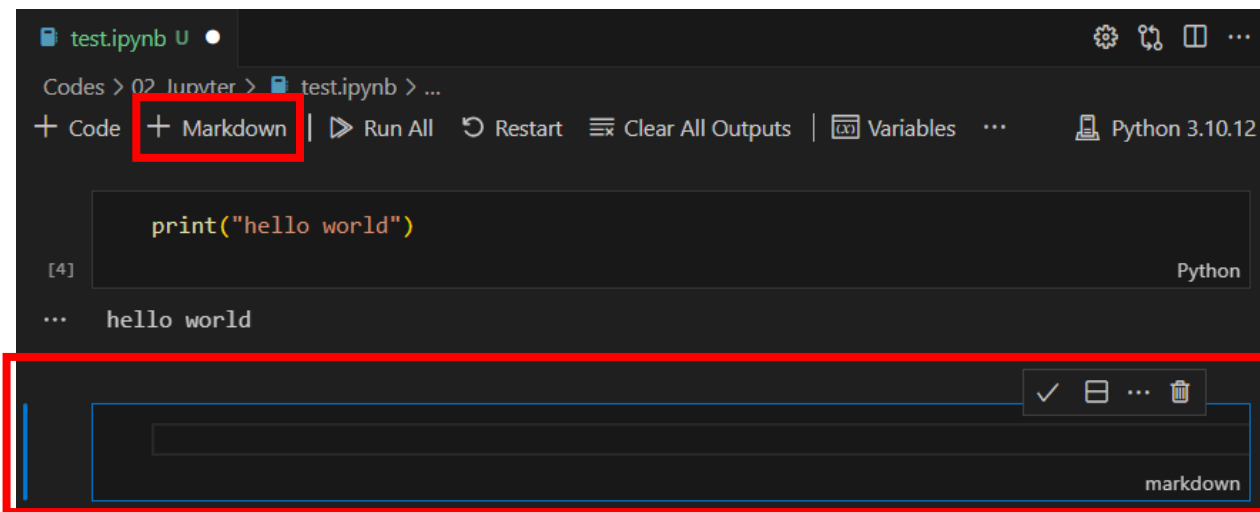
- 코드블록 안의 코드를 실행하는 방법
 - ✓ 좌측의 실행버튼 누르기
 - ✓ Shift + Enter → 현재 코드박스를 실행하고 다음 코드박스 생성
 - ✓ Ctrl + Enter → 현재 코드박스 실행만 처리
- 좌측 실행버튼 아래의 숫자는 코드블록의 실행 순서를 나타내는 번호
 - ✓ Jupyter는 코드가 위에서 부터 순서대로 실행되는 것이 아니라 실행순서를 사용자가 선택하여 실행할 수 있으므로 확인 필요



The screenshot shows a Jupyter Notebook window titled 'test.ipynb U'. The interface includes a toolbar with buttons for '+ Code', '+ Markdown', 'Run All', 'Restart', 'Clear All Outputs', 'Variables', and 'Python 3.10.12'. Below the toolbar, a code cell is displayed with the code `print("hello world")`. To the left of the code, a red box highlights the execution number '[2]'. To the right of the code, a green checkmark and '0.0s' indicate successful execution. Below the code cell, the output 'hello world' is visible.

❖ Jupyter Markdown 블록

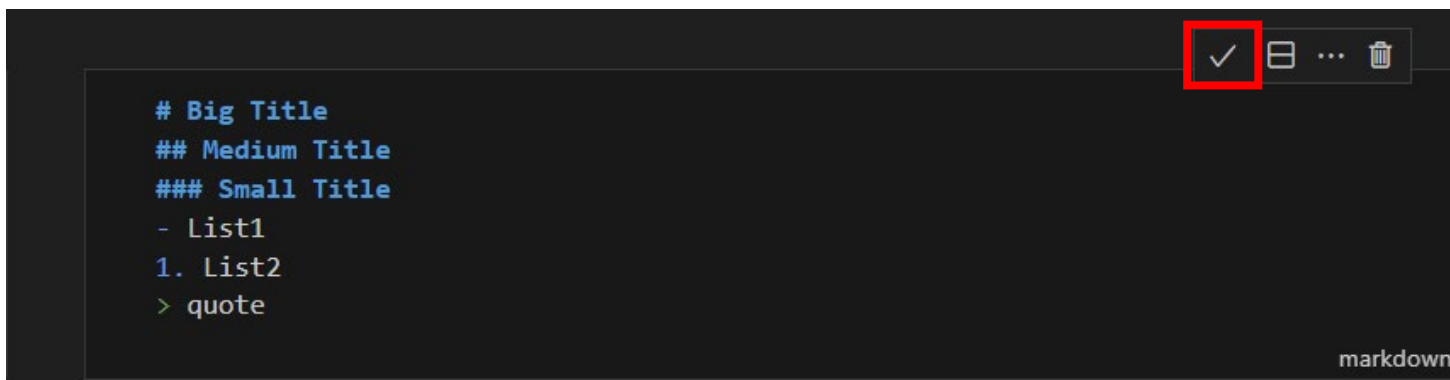
- 마크다운 블록은 Jupyter 파일을 문서로 활용할 수 있도록 지원(LaTeX 문법)
- 주석처럼 사용 가능하며 제목 달기, 수식 작성 등이 가능



- 위의 버튼을 눌러서 마크다운 블록을 생성
- 이미 생성된 코드 블록의 설정을 선택하여 마크다운 블록으로 변경
 - ✓ 마크다운에서 코드블록으로 변경 가능
 - ✓ 마크다운 → 코드 (단축기 y) / 코드 → 마크다운 (단축기 m)

❖ Jupyter Markdown 블록

- 마크다운 블록은 Jupyter 파일을 문서로 활용할 수 있도록 지원(LaTeX 문법)
- 주석처럼 사용 가능하며 제목 달기, 수식 작성 등이 가능



```
# Big Title
## Medium Title
### Small Title
- List1
1. List2
> quote
```

markdown



2 Jupyter로 Topic 접근

❖ Topic 구독 코드 작성

- 터미널에서 turtlesim_node 실행
- ROS2를 실행한 후 VS-Code 실행
- 코드 블록에 다음과 같은 코드 입력

```
import rclpy as rp
from turtlesim.msg import Pose
```

- ✓ 첫째 라인은 rclpy(Ros Client Library for Python) 라는 ROS2를 Python에서 사용할 수 있게 해주는 모듈을 rp라는 이름으로 import
- ✓ 둘째 라인은 Subscribe하고자 하는 Topic인 /turtle1/pose의 type을 참고하여 turtlesim/msg/Pose에서 Pose를 사용할 수 있도록 import

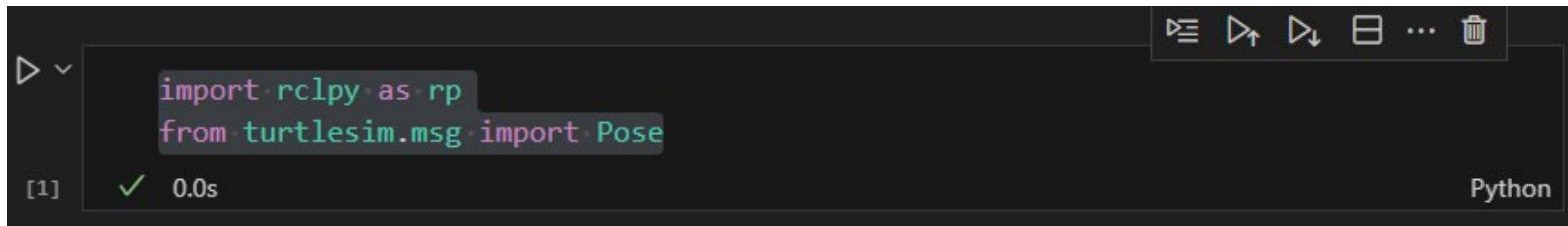
```
daesung@DSThinkPad:~$ ros2 interface show turtlesim/msg/Pose
float32 x
float32 y
float32 theta

float32 linear_velocity
float32 angular_velocity
```

https://docs.ros2.org/crystal/api/rclpy/api/init_shutdown.html

❖ Topic 구독 코드 작성

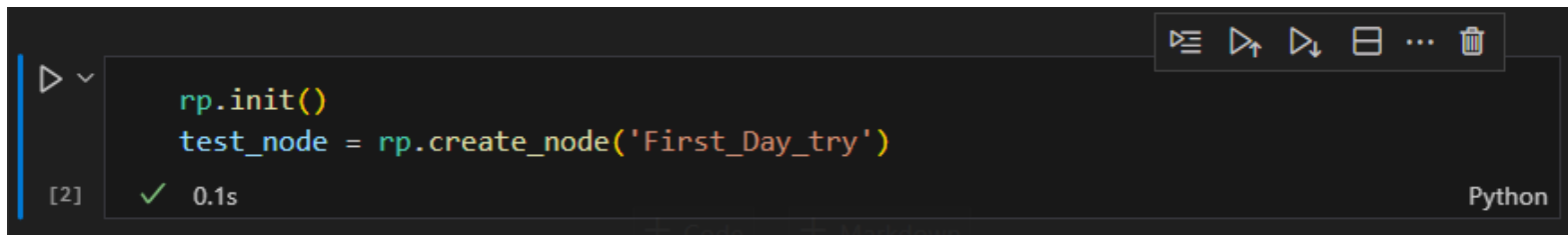
- 실행 시 에러가 없을 경우 정상적으로 환경 설정되었음



```
import rclpy as rp
from turtlesim.msg import Pose
```

[1] ✓ 0.0s Python

- rclpy 초기화 및 해당 라이브러리의 create_node() 메서드를 이용하여 First_Day_Try 라는 이름의 노드를 생성하고 test_node로 객체화



```
rp.init()
test_node = rp.create_node('First_Day_try')
```

[2] ✓ 0.1s Python

❖ Topic 구독 코드 작성

- Topic을 수신할 때 마다 실행하는 메서드는 callback()
- Topic 수신 시 처리할 내용은 callback() 메서드에 작성
- 아래의 코드는 callback함수가 실행할 내용에 대해서만 작성, 함수의 호출은 다른 곳에서 실시
- 해당 코드 블록 실행 시 에러가 없어야 함

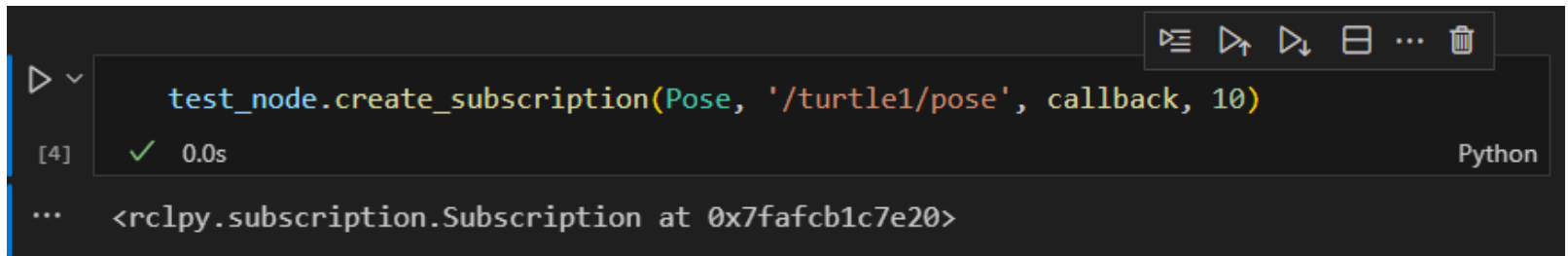
```
def callback(data) :  
    print("---->")  
    print("/turtle1/pose : ", data)  
    print("X : ", data.x)  
    print("y : ", data.y)  
    print("Theta : ", data.theta)
```

[3] ✓ 0.0s

Python

❖ Topic 구독 코드 작성

- Python 코드로 생성한 노드인 test_node에서 Subscription 생성
- 문제가 없을 경우 다음과 같은 실행결과가 나타남
- 마지막 값은 사용자 마다 다르게 나타남
- 해당 코드는 Subscription을 생성하는 코드이며 Subscription을 실행하는 코드는 아님

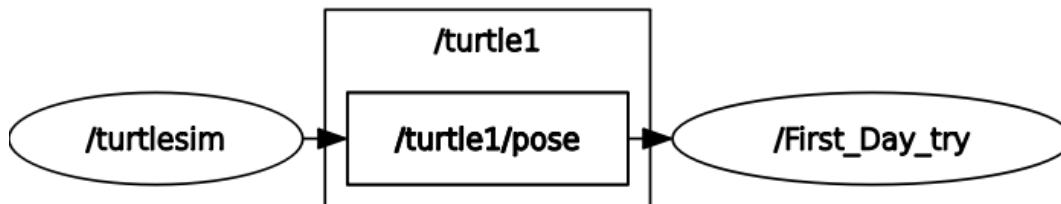


```
test_node.create_subscription(Pose, '/turtle1/pose', callback, 10)
```

[4] ✓ 0.0s Python

... <rcipy.subscription.Subscription at 0x7fafcb1c7e20>

✓ create_subscription([data type] , [topic name] , [callback] , [QoS History])



❖ Topic 구독 코드 작성

- spin_once() 메서드를 이용하여 Topic을 한번만 구독하도록 작성

```
rp.spin_once(test_node)

[5] ✓ 0.0s Python

... --->
/turtle1/pose : turtlesim.msg.Pose(x=5.544444561004639, y=5.544444561004639, theta=0.0
X : 5.544444561004639
y : 5.544444561004639
Theta : 0.0
```

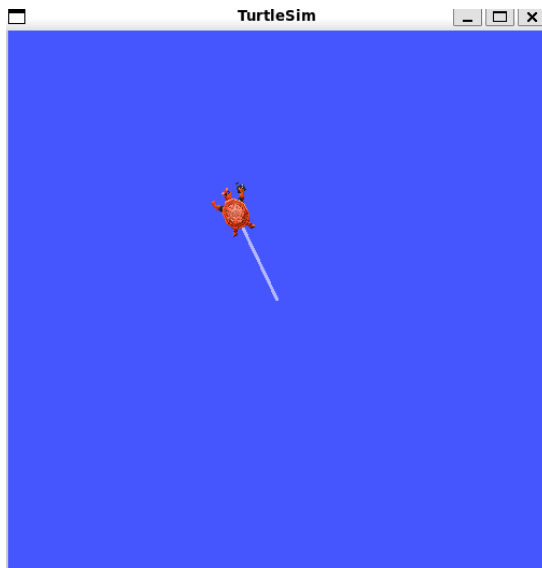
- 거북이의 위치 변경을 위해 turtle_teleop_key 노드를 실행하고 위치 이동 후 Topic 값을 다시 확인하여 제대로 동작하는지 확인

```
daesung@DSThinkPad:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
'Q' to quit.
```

❖ Topic 구독 코드 작성

- 거북이의 위치 변경을 위해 turtle_teleop_key 노드를 실행하고 위치 이동 후 Topic 값을 다시 확인하여 제대로 동작하는지 확인

```
daesung@DSThinkPad:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
'Q' to quit.
```

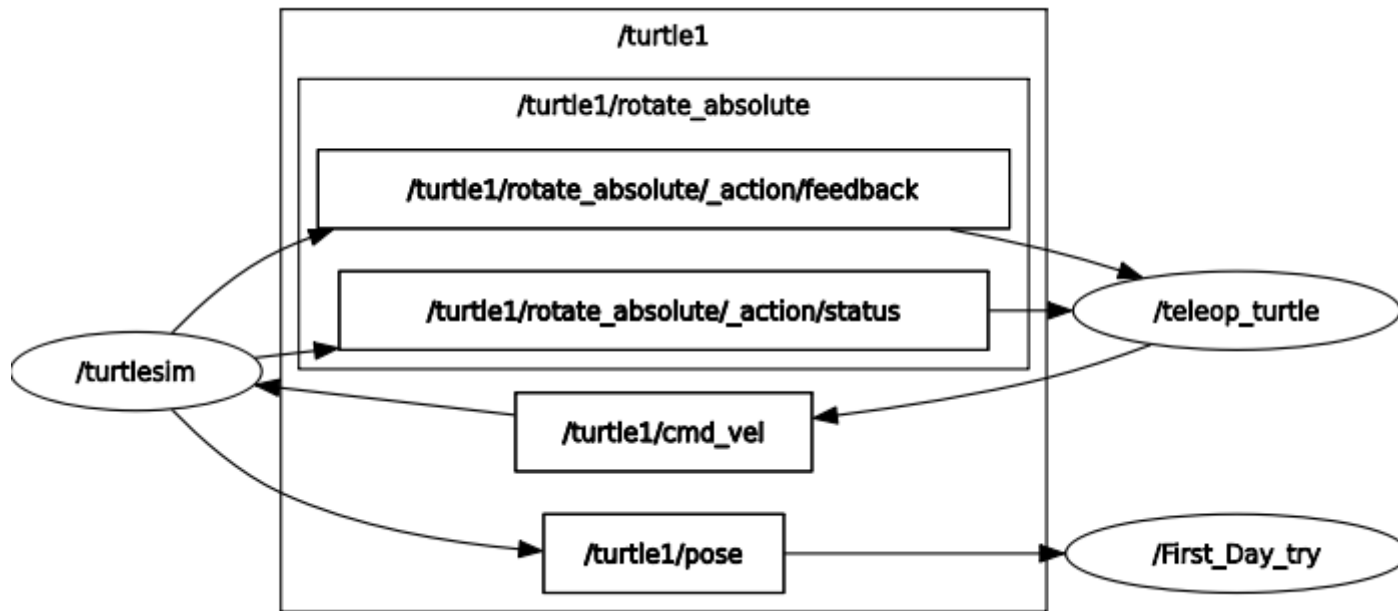


```
rp.spin_once(test_node)
[5] ✓ 0.0s Python
... --->
/turtle1/pose : turtlesim.msg.Pose(x=5.544444561004639, y=5.544444561004639, theta=0.0
X : 5.544444561004639
y : 5.544444561004639
Theta : 0.0
```

```
rp.spin_once(test_node)
[6] ✓ 0.0s Python
... --->
/turtle1/pose : turtlesim.msg.Pose(x=4.676270961761475, y=7.363930702209473, theta=2.0
X : 4.676270961761475
y : 7.363930702209473
Theta : 2.0160000324249268
```

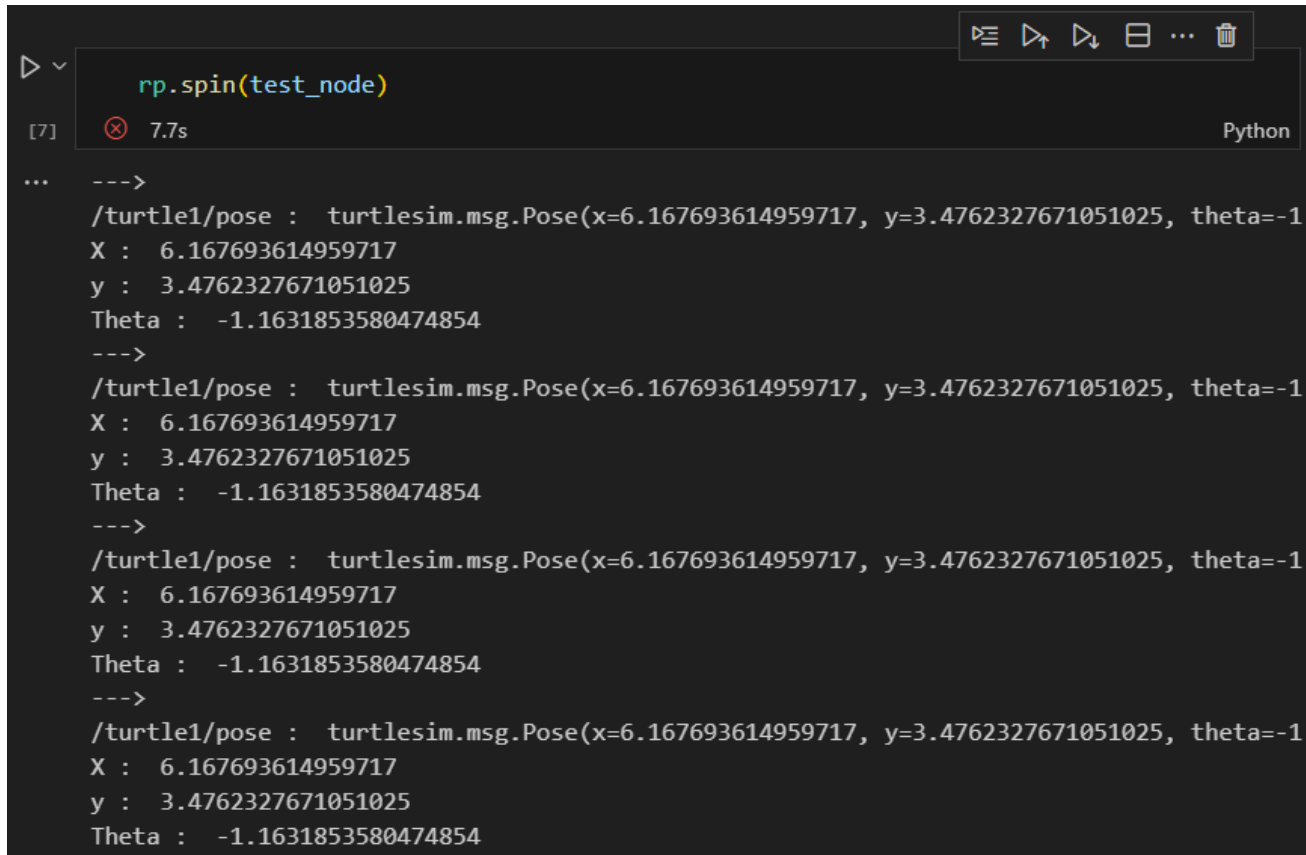

❖ Topic 구독 코드 작성

- rqt_graph를 실행하여 다음과 같이 구성되어 있는가 확인



❖ Topic 구독 횟수 제한하기

- 다음과 같은 spin() 메서드를 사용하면 계속해서 Topic 구독
- 좌측의 실행 멈춤 버튼을 통해 동작 중지 가능



The image shows a Jupyter Notebook interface. At the top, there is a toolbar with icons for running, stepping through, and other actions. Below the toolbar is a code cell with the following Python code:

```
rp.spin(test_node)
```

The cell is labeled [7] and has a red 'x' icon, indicating it has been executed. The output of the cell is displayed below the code, showing a continuous stream of data from the /turtle1/pose topic. The output is as follows:

```
... --->
/turtle1/pose : turtlesim.msg.Pose(x=6.167693614959717, y=3.4762327671051025, theta=-1
X : 6.167693614959717
y : 3.4762327671051025
Theta : -1.1631853580474854
--->
/turtle1/pose : turtlesim.msg.Pose(x=6.167693614959717, y=3.4762327671051025, theta=-1
X : 6.167693614959717
y : 3.4762327671051025
Theta : -1.1631853580474854
--->
/turtle1/pose : turtlesim.msg.Pose(x=6.167693614959717, y=3.4762327671051025, theta=-1
X : 6.167693614959717
y : 3.4762327671051025
Theta : -1.1631853580474854
--->
/turtle1/pose : turtlesim.msg.Pose(x=6.167693614959717, y=3.4762327671051025, theta=-1
X : 6.167693614959717
y : 3.4762327671051025
Theta : -1.1631853580474854
```

❖ Topic 구독 횟수 제한하기

- 구독 횟수를 제한하기 위해 callback() 메서드 수정

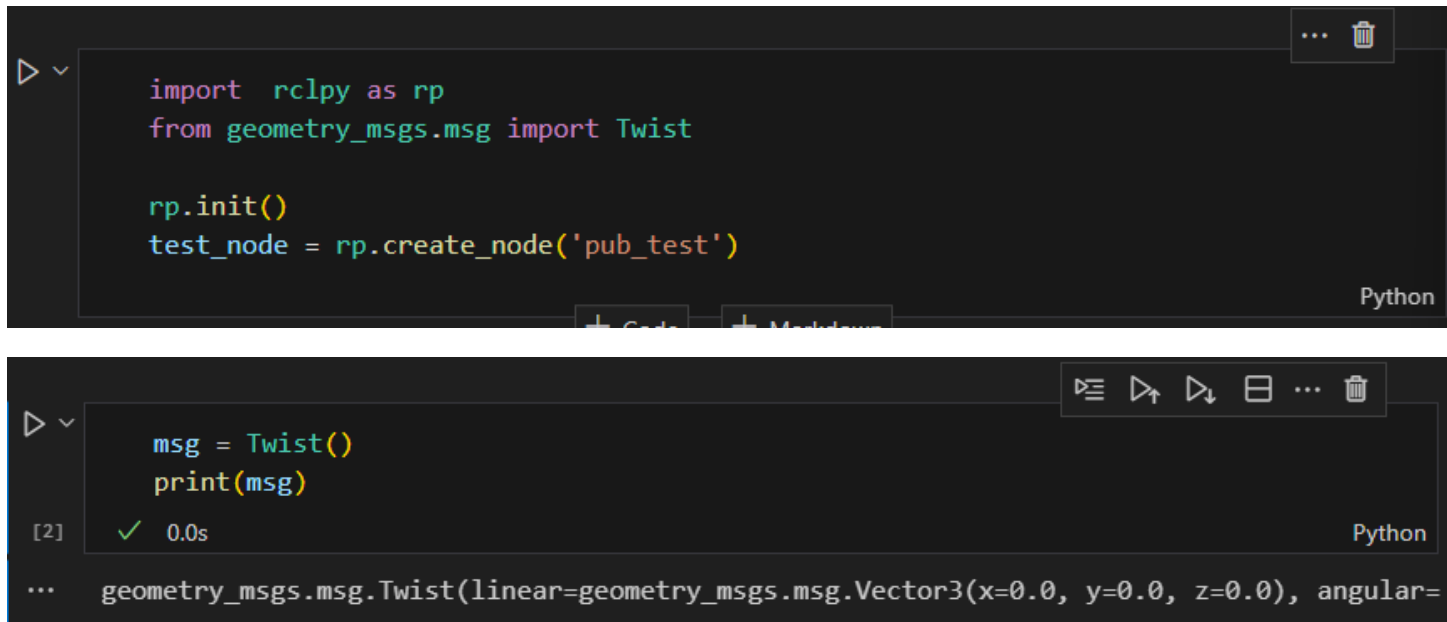
```
cnt = 0
def callback(data) :
    global cnt
    cnt += 1
    print("--->", cnt)
    print("/turtle1/pose : ", data)
    print("X : ", data.x)
    print("y : ", data.y)
    print("Theta : ", data.theta)
    if cnt>3:
        raise Exception("Subscription Stop!")
```

[] Python

- ✓ raise Exception()을 이용하여 예외사항을 발생시키고 메서드 실행 중지
- ✓ 그 외 node의 destroy_subscription() 메서드 사용 해서 중지 가능

❖ Topic 발행 코드 작성

- Topic 발행을 위한 새로운 Jupyter 파일 생성



```
import rclpy as rp
from geometry_msgs.msg import Twist

rp.init()
test_node = rp.create_node('pub_test')
```

```
msg = Twist()
print(msg)
```

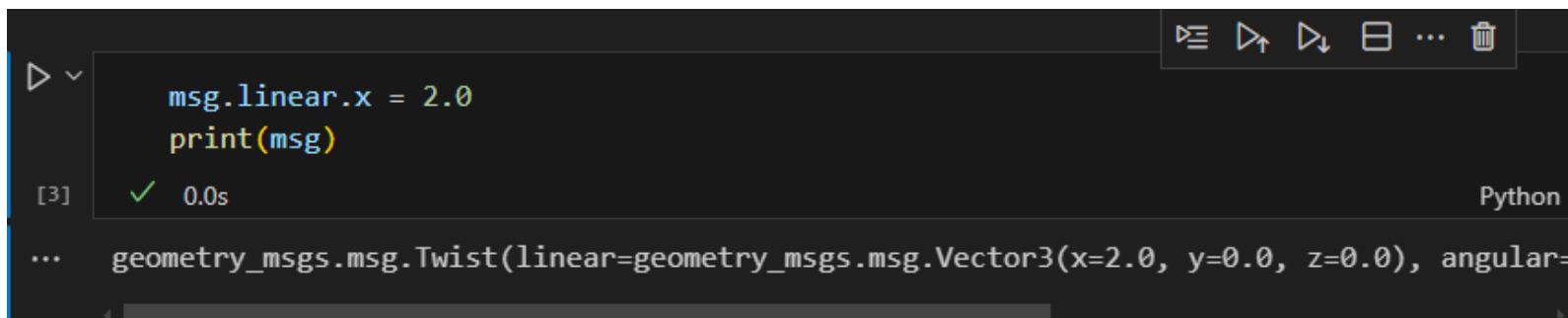
[2] ✓ 0.0s

... geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=

- ✓ Twist의 형태를 가지는 msg 객체 생성 후 내용 출력
- ✓ msg에 linear, angular 데이터가 존재하고 다시 x, y, z값이 존재함을 확인할 수 있음

❖ Topic 발행 코드 작성

- linear의 x값을 변경 후 확인

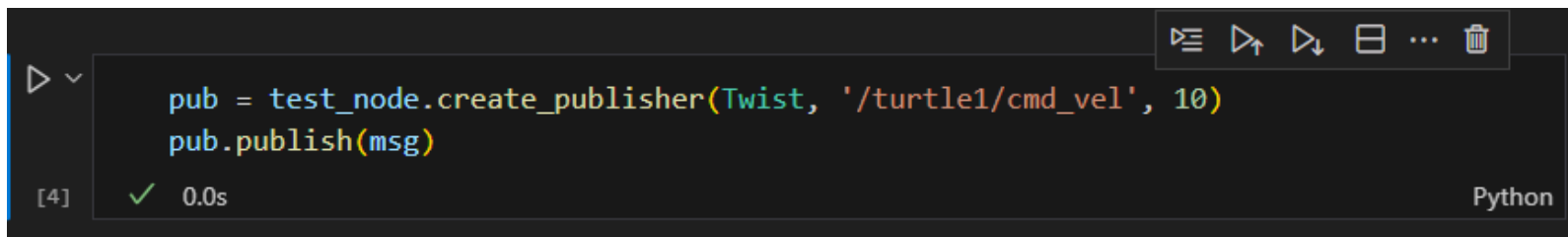


```
msg.linear.x = 2.0
print(msg)
```

[3] ✓ 0.0s Python

... geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=

- Topic 발행코드 작성



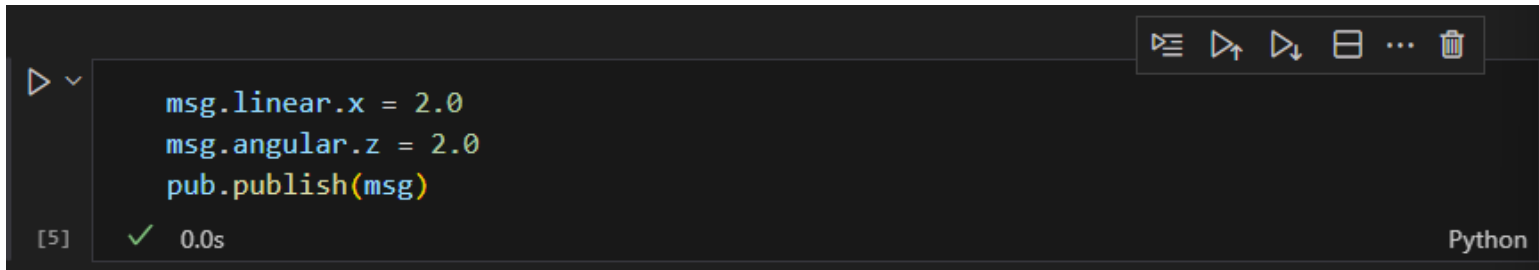
```
pub = test_node.create_publisher(Twist, '/turtle1/cmd_vel', 10)
pub.publish(msg)
```

[4] ✓ 0.0s Python

- ✓ 코드 블록 실행 시 turtlesim의 거북이가 움직이는 것을 확인할 수 있음

❖ Topic 발행 코드 작성

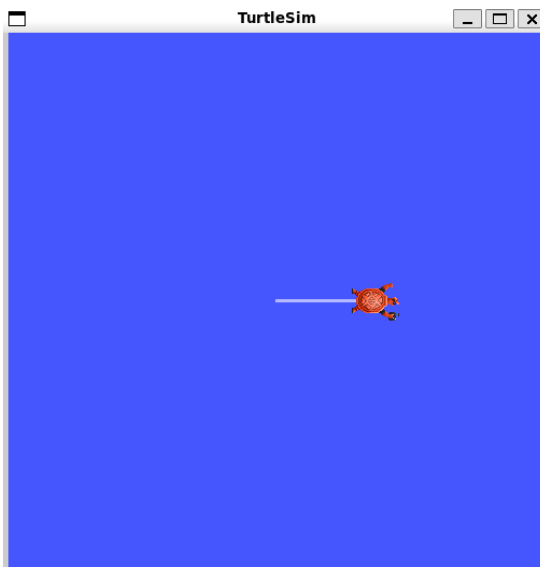
- 값 변경 후 다시 토픽 발행



```
msg.linear.x = 2.0  
msg.angular.z = 2.0  
pub.publish(msg)
```

[5] ✓ 0.0s Python

- ✓ 코드 블록 실행 시 turtlesim의 거북이가 곡선을 그리며 움직이는 것을 확인할 수 있음



❖ Topic 발행 코드 작성

- 타이머를 이용하여 토픽 발행, timer_callback() 메서드 작성

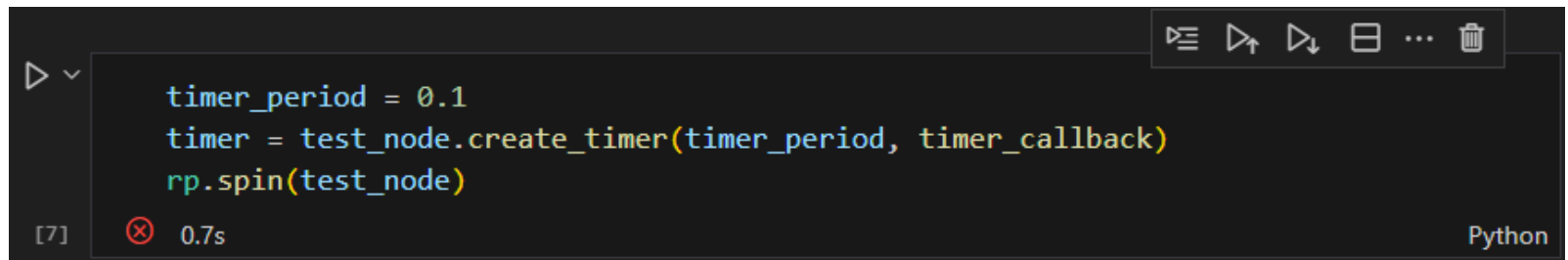


```
cnt=0
def timer_callback():
    global cnt
    cnt += 1
    print(cnt)
    pub.publish(msg)

    if cnt>5 :
        raise Exception("Publish Stop!")
```

[6] ✓ 0.0s Python

The image shows a Jupyter Notebook cell with Python code for a timer callback. The code increments a counter and publishes a message. A TurtleSim window is visible on the right, showing a turtle drawing a circle. The cell status is [6] ✓ 0.0s Python.



```
timer_period = 0.1
timer = test_node.create_timer(timer_period, timer_callback)
rp.spin(test_node)
```

[7] ✗ 0.7s Python

The image shows a Jupyter Notebook cell with Python code for setting up a timer and spinning the node. The code sets a timer period, creates a timer, and calls spin. The cell status is [7] ✗ 0.7s Python.

❖ 노드 종료

- node의 `destroy_node()` 메서드를 이용하여 생성한 노드 종료

A screenshot of a Jupyter Notebook cell. The cell contains the code `test_node.destroy_node()`. Below the code, it shows the execution status: `[8]`, a green checkmark, and `0.0s`. The language is set to `Python`. The cell has a toolbar with icons for running, stepping through, and other Jupyter actions.

```
test_node.destroy_node()
```

[8] ✓ 0.0s Python

❖ Python 으로 Topic 접근 코드 작성하기

- 앞서 작성한 Jupyter Code를 참고하여 Python으로 Turtlesim의 Pose를 구독하고, 동작 시킬 수 있는 응용 프로그램 만들기
 - ✓ 거북이가 계속 원을 그리며 돌 수 있는 코드 작성
 - ✓ 거북이가 사각형을 그리며 주행할 수 있는 코드 작성
 - ✓ 거북이가 일정한 구역 안에서만 랜덤하게 주행하는 코드 작성

3

Jupyter로 Service Client 구현

❖ Service Client 코드 작성

- 터미널에서 turtlesim_node 실행
- ROS2를 실행한 후 VS-Code 실행하고, ~~~.ipynb 파일 생성
- 코드 블록에 다음과 같은 코드 입력

```
import rclpy as rp
from turtlesim.srv import TeleportAbsolute
```

- rclpy 초기화 및 해당 라이브러리의 create_node() 메서드를 이용하여 client_test 라는 이름의 노드 생성

```
rp.init()
test_node = rp.create_node('client_test')
```

❖ Service Client 코드 작성

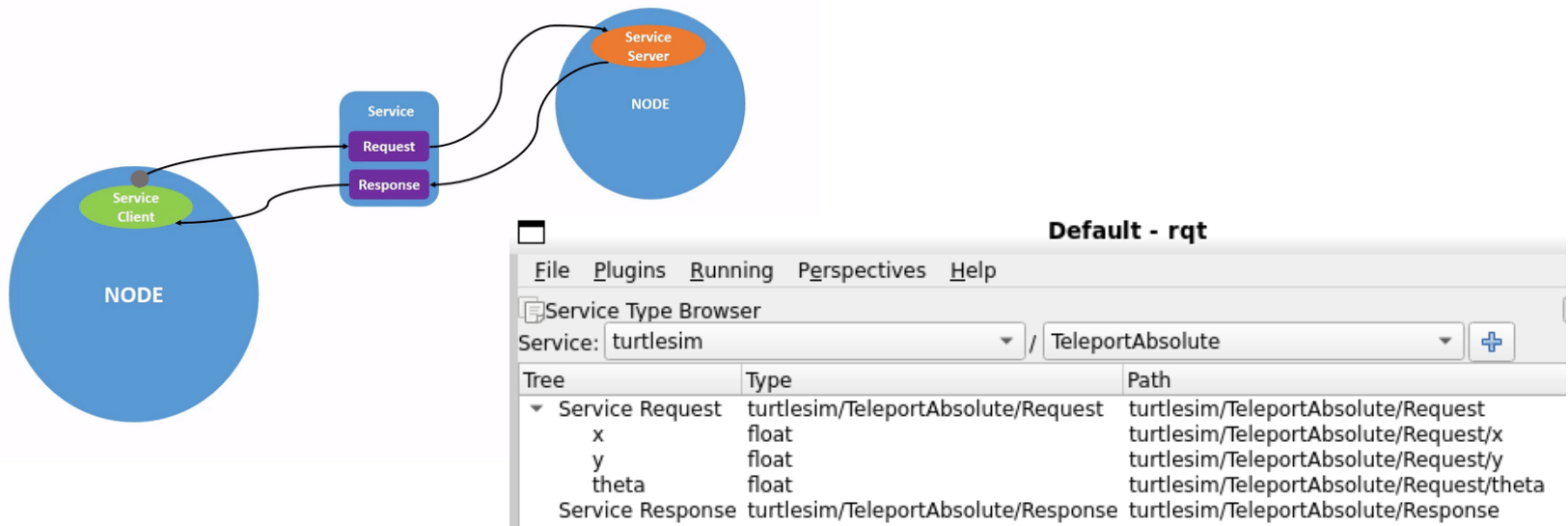
- /turtle1/teleport_absolute Service를 다룰 예정이므로 service_name 변수에 저장
- node의 create_client() 메서드를 이용하여 Client를 생성하고 cli 객체 생성

```
service_name = '/turtle1/teleport_absolute'
cli = test_node.create_client(TeleportAbsolute, service_name)
```

[3]

✓ 0.0s

Python



❖ Service Client 코드 작성

- Service Client가 Request에서 사용할 데이터 객체생성 및 초기값 확인

```
req = TeleportAbsolute.Request()
req
```

[4] ✓ 0.0s Python

... turtlesim.srv.TeleportAbsolute_Request(x=0.0, y=0.0, theta=0.0)

- Request 값 설정 및 확인

```
req.x = 1.
req.y = 1.
req.theta = 1.57

req
```

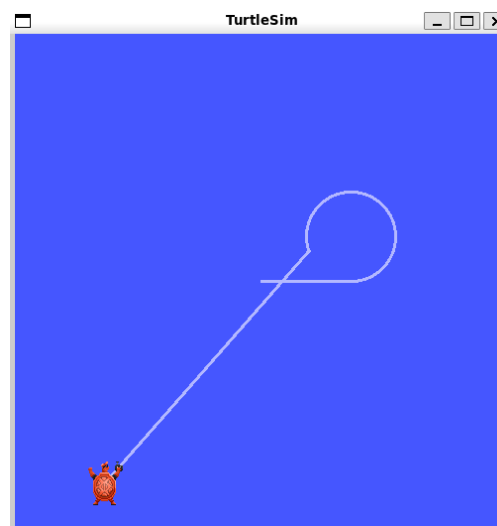
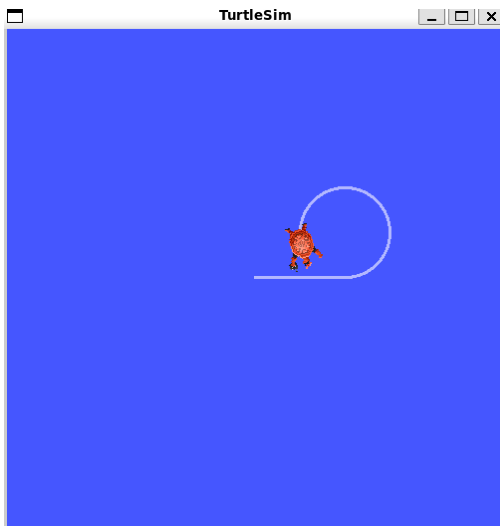
[5] ✓ 0.0s Python

... turtlesim.srv.TeleportAbsolute_Request(x=1.0, y=1.0, theta=1.57)

❖ Service Client 코드 작성

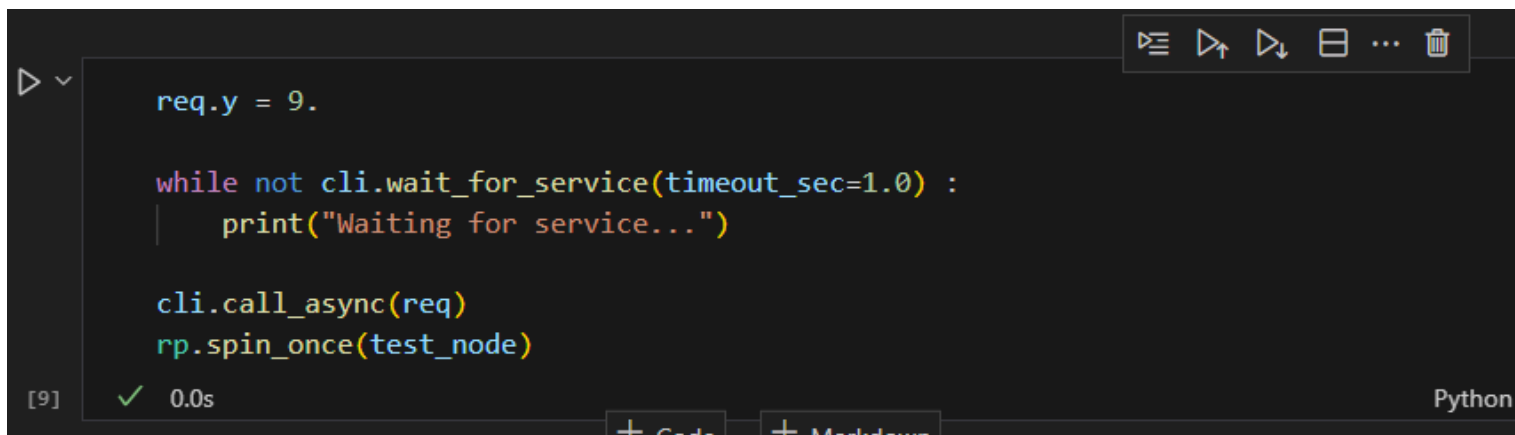
- Service call 한번 실행하기 (값을 바꾸면서 테스트)

```
req.x = 2.  
  
cli.call_async(req)  
rp.spin_once(test_node)  
[6] ✓ 0.0s Python
```



❖ Service Client 코드 작성

- Service Client는 Service Server 노드가 실행되고 있어야 사용할 수 있음
- `wait_for_service()` 메서드는 Service Server보다 Service Client가 먼저 실행되어 있을 경우에 사용하여 Server가 실행되기를 기다릴 수 있음



```
req.y = 9.

while not cli.wait_for_service(timeout_sec=1.0) :
    print("Waiting for service...")

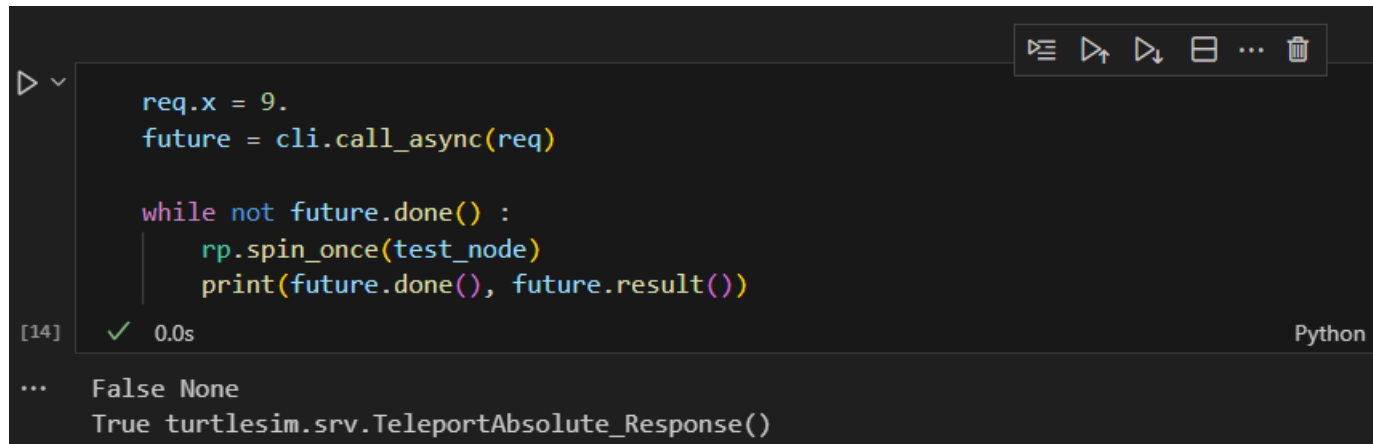
cli.call_async(req)
rp.spin_once(test_node)
```

[9] ✓ 0.0s Python

- ✓ 해당 Service Server가 실행되지 않을 경우 `wait_for_service()` 메서드는 `false` 값 반환
- ✓ 위 코드는 1초에 한 번씩 `print()`코드가 실행되며, 해당 Server가 동작할 경우 다음 라인의 코드를 실행

❖ Service Client 코드 작성

- 앞서 제공한 rclpy 문서 링크에서 확인해 보면 `call_async()` 메서드는 Future라는 값을 반환
- Future 내의 `done()`, `result()` 메서드가 Service Client 호출 후의 상황을 알려줌



```

req.x = 9.
future = cli.call_async(req)

while not future.done() :
    rp.spin_once(test_node)
    print(future.done(), future.result())

```

[14] ✓ 0.0s Python

... False None
True turtlesim.srv.TeleportAbsolute_Response()

- ✓ 요청한 Request가 수행되었다는 Response가 도착할 때 까지 반복문을 계속 실행

❖ Python 으로 Service 접근 코드 작성하기

- 앞서 작성한 Jupyter Code를 참고하여 Python으로 Turtlesim을 Service Server로 하여 요청하는 Service Client 응용 프로그램 만들기
 - ✓ Turtlesim 노드가 처리할 수 있는 Service 목록 확인
 - ✓ Service 목록 중 5가지 Service에 대한 기능을 선택적으로 요청할 수 있는 기능 구현.
 - ❖ 예를 들어 /clear, /spawn, /turtle1/set_pen, turtle1/teleport_relative 등

Q & A
