

ROS 2

[05 – ROS2 Tools & Python Class]

한국폴리텍대학교 성남캠퍼스



Homework Review

❖ Missions

- Turtlesim의 Pose를 구독하고, 동작 시킬 수 있는 응용 프로그램 만들기
 1. 거북이가 계속 원을 그리며 돌 수 있는 코드 작성
 2. 거북이가 일정한 구역 안에서만 랜덤하게 주행하는 코드 작성
- Turtlesim을 Service Server로 하는 Service Client 응용 프로그램 만들기
 - Turtlesim 노드가 처리할 수 있는 Service 목록 확인
 - 3. Service 목록 중 5가지 Service에 대한 기능을 선택적으로 요청할 수 있는 기능 구현.
 - ❖ 예를 들어 /clear, /spawn, /turtle1/set_pen, turtle1/teleport_relative 등

❖ M-1) 거북이가 계속 원을 그리며 돌 수 있는 코드 작성

- Topic을 주기적으로 발행하여 거북이 동작
- /turtle1/cmd_vel Topic을 이용하여 명령어 전달
- m1_node를 만들고 msg 초기화

```
daesung@DSThinkPad:~/test$ 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
```

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

rp.init()
m1_node = rp.create_node('pub_circling')
msg = Twist()
```

❖ M-1) 거북이가 계속 원을 그리며 돌 수 있는 코드 작성

- m1_node에 '/turtle1/cmd_vel' Topic을 발행하는 publisher pub 생성
- msg의 데이터 설정
- 타이머를 이용하여 1초 마다 Topic 발행

```
pub = m1_node.create_publisher(Twist, '/turtle1/cmd_vel', 10)

msg.linear.x = 2.0
msg.angular.z = 3.14

def timer_callback():
    pub.publish(msg)

timer_period = 1
timer = m1_node.create_timer(timer_period, timer_callback)
rp.spin(m1_node)
```

- ❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행
 - [Step-1] 거북이가 랜덤하게 움직일 수 있는 Topic을 주기적으로 발행
 - [Step-2] 거북이가 움직일 때 현재 좌표를 Terminal에 표시
 - [Step-3] 지정한 영역을 벗어난 경우 지정한 영역으로 돌아오는 코드 작성
 - [Step-4] 3가지 코드를 조합하여 하나의 Python 프로그램 완성

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-1] 거북이가 랜덤하게 움직일 수 있는 Topic을 주기적으로 발행

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

rp.init()
m2_node = rp.create_node('AmazingNode')
msg = Twist()
```

- Mission-1과 같이 Topic을 주기적으로 발행하여 거북이 동작
- 동일하게 /turtle1/cmd_vel Topic의 geometry_msgs/msg/Twist을 이용하여 명령어 전달
- m2_node를 만들고 msg 초기화

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-1] 거북이가 랜덤하게 움직일 수 있는 Topic을 주기적으로 발행

```
pub = m2_node.create_publisher(Twist, '/turtle1/cmd_vel', 10)

timer_period = 0.5
timer = m2_node.create_timer(timer_period, timer_callback)
rp.spin(m2_node)
```

- m2_node에 '/turtle1/cmd_vel' Topic을 발행하는 publisher pub 생성
- 타이머를 이용하여 0.5초 마다 Topic 발행
- Topic을 생성하고 발행하는 동작은 timer_callback 메서드에서 작성하여 동작

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-1] 거북이가 랜덤하게 움직일 수 있는 Topic을 주기적으로 발행

```
def timer_callback() :  
    rnd_sign = random.randint(0,1)  
    rnd_linear = float(random.randint(10,30))/10.0  
    rnd_angle = float(random.randint(1,314))/100.0  
  
    msg.linear.x = rnd_linear  
    if(rnd_sign) : msg.angular.z = rnd_angle  
    else : msg.angular.z = -rnd_angle  
  
    pub.publish(msg)
```

- 랜덤주행은 전진만 하는 것으로 가정하며, 이동 값은 rnd_linear의 값을 1.0~3.0 사이에서 랜덤하게 생성
- 좌우 회전을 위해 회전각의 값은 rnd_sign 값을 이용해 True는 좌회전, False는 우회전, rnd_angle 값을 이용해 0~180도의 회전각을 가지도록 설정

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-2] 거북이가 움직일 때 현재 좌표를 Terminal에 표시

```
daesung@DSThinkPad:~/test$ 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
```

```
import random
import rclpy as rp
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
```

- 거북이의 현재 좌표를 구독하기 위해 /turtle1/pose Topic을 구독
- turtlesim/msg/Pose 형태의 값 사용

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-2] 거북이가 움직일 때 현재 좌표를 Terminal에 표시

```
sub = m2_node.create_subscription(Pose, '/turtle1/pose',  
callback, 10)
```

- m2_node에 구독 모듈을 생성하고 /turtle1/pose Topic의 구독과 관련된 사항은 callback 메서드에서 처리

```
def callback(data) :  
    print(f"Pose> X: %.2f y: %.2f"%(data.x, data.y))
```

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-3] 지정한 영역을 벗어난 경우 지정한 영역으로 돌아오는 코드 작성

```
b_linear=0.0  
b_angular=0.0
```

- 지정 영역을 벗어나기 직전에 발행한 토픽 값 저장을 위한 전역 변수 선언
- 랜덤주행 Topic 발행 시 해당 값 저장, flag 변수는 지정영역 안에 있을 경우 True

```
def timer_callback() :  
    global b_linear  
    global b_angular  
    if flag :  
        rnd_sign = random.randint(0,1)  
        rnd_linear = float(random.randint(10,30))/10.0  
        rnd_angle = float(random.randint(1,314))/100.0  
  
        b_linear = rnd_linear  
        b_angular = rnd_angle
```

❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행

- [Step-3] 지정한 영역을 벗어난 경우 지정한 영역으로 돌아오는 코드 작성

```
def callback(data) :  
    global flag  
    print(f"Pose> X: %.2f y: %.2f"%(data.x, data.y))  
    if ((data.x>10.0)or(data.x<1.0)) or  
((data.y>10.0)or(data.y<1.0)):  
        flag = False  
        msg.linear.x = -b_linear  
        msg.angular.z = -b_angular  
        pub.publish(msg)  
        msg.linear.x = 0.0  
        msg.angular.z = 3.14  
        pub.publish(msg)  
    else :  
        flag = True
```

- 지정한 영역을 벗어나면 flag값을 False로 그렇지 않을 경우 True로 설정
- 지정 영역을 벗어난 경우 직전 위치로 이동하는 Topic 발행 후 180도 회전

0

Homework Review

- ❖ M-2) 거북이가 일정한 구역 안에서 랜덤하게 주행
 - [Step-4] 3가지 코드를 조합하여 하나의 Python 프로그램 완성
 - 참고사이트
 - https://github.com/Daesung7723/ROS2/tree/main/Codes/03_Homework

❖ M-3) Turtlesim Service Client 구현

- Turtlesim 노드가 처리할 수 있는 Service 목록 확인
- 5가지 Service에 대한 기능 선택
- /reset, /clear, /turtle1/set_pen, /turtle1/teleport_absolute, /turtle1/teleport_relative

```
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:
```

❖ M-3) Turtlesim Service Client 구현

- Node 및 Service Client 생성

```
import rclpy as rp
from std_srvs.srv import Empty
from turtlesim.srv import SetPen, TeleportAbsolute, TeleportRelative

rp.init()
m3_node = rp.create_node('Service_Command')

client_clr = m3_node.create_client(Empty, '/clear')
client_rst = m3_node.create_client(Empty, '/reset')
client_setpen = m3_node.create_client(SetPen, '/turtle1/set_pen')
client_tabs = m3_node.create_client(TeleportAbsolute,
    '/turtle1/teleport_absolute')
client_trel = m3_node.create_client(TeleportRelative,
    '/turtle1/teleport_relative')
```


❖ M-3) Turtlesim Service Client 구현

- 메뉴 및 Main 코드 생성

```
def menu() -> int:
    print('1. reset')
    print('2. clear')
    print('3. set pen')
    print('4. teleport abs')
    print('5. teleport rel')
    print('6. exit')
    try:
        menu = int(input('Select num = '))
        return menu
    except:
        return 0
```

❖ M-3) Turtlesim Service Client 구현

- 메뉴 및 Main 코드 생성

```
def main():  
    while True:  
        m = menu()  
        if m == 6 : break  
        elif m==1:  
            reset()  
        elif m==2:  
            clear()  
        elif m==3:  
            setpen()  
        elif m==4:  
            teleport_abs()  
        elif m==5:  
            teleport_rel()  
  
if __name__ == '__main__':  
    main()
```

❖ M-3) Turtlesim Service Client 구현

- 각 메뉴 별 실행함수 생성 - 예) /turtle1/setpen

```
def setpen():  
    setpen_req = SetPen.Request()  
  
    setpen_req.r, setpen_req.g, setpen_req.b, setpen_req.width \  
        = map(int, input('R G B W = ').split())  
  
    client_setpen.call_async(setpen_req)
```

- 나머지 함수 및 완성 코드는 다음 링크 참고
- https://github.com/Daesung7723/ROS2/tree/main/Codes/03_Homework

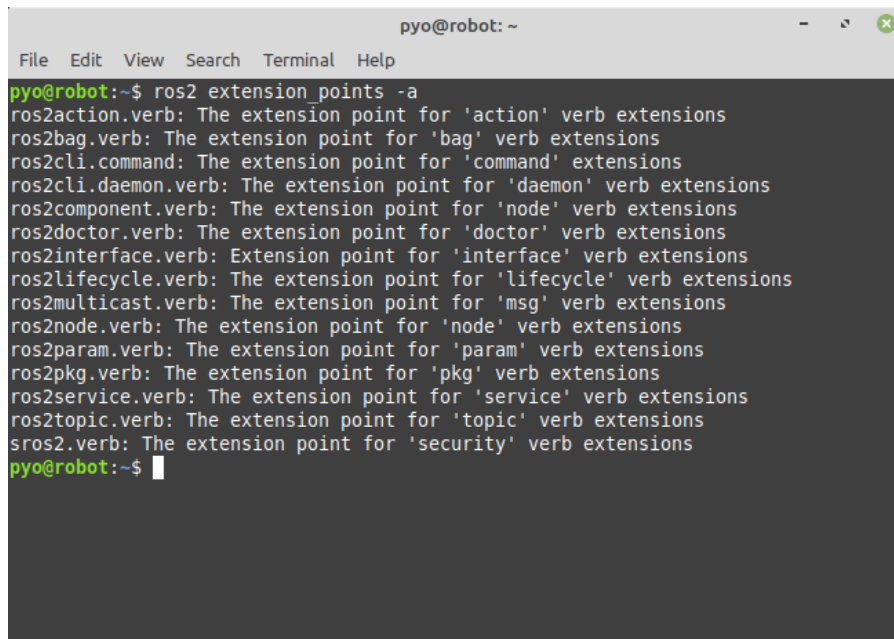
1 ROS2 Tools

❖ ROS2 Tools

- ROS 2의 가장 큰 특징은 로봇 개발에 필요한 다양한 개발 도구가 있는 점
- ROS의 도구들은 ROS 커뮤니티에서 다년간의 의견과 경험이 녹아져 만들어진 필수 도구들이라고 볼 수 있음
- 이를 이용하여 프로젝트의 코드들을 쉽게 빌드하고 패키징하고 패키지와 노드를 관리하고 데이터들을 기록, 재생, 관리
- CLI 형태, GUI 형태의 프로그램을 만들고 3차원 시뮬레이터안에서 자유자재로 디버깅하고 테스트할 수 있음
 - ✓ CLI 형태의 Command-Line Tools, GUI 형태의 RQt, 3차원 시각화 도구인 RViz, 3차원 시뮬레이터 Gazebo

❖ CLI기반 Command-Line Tools

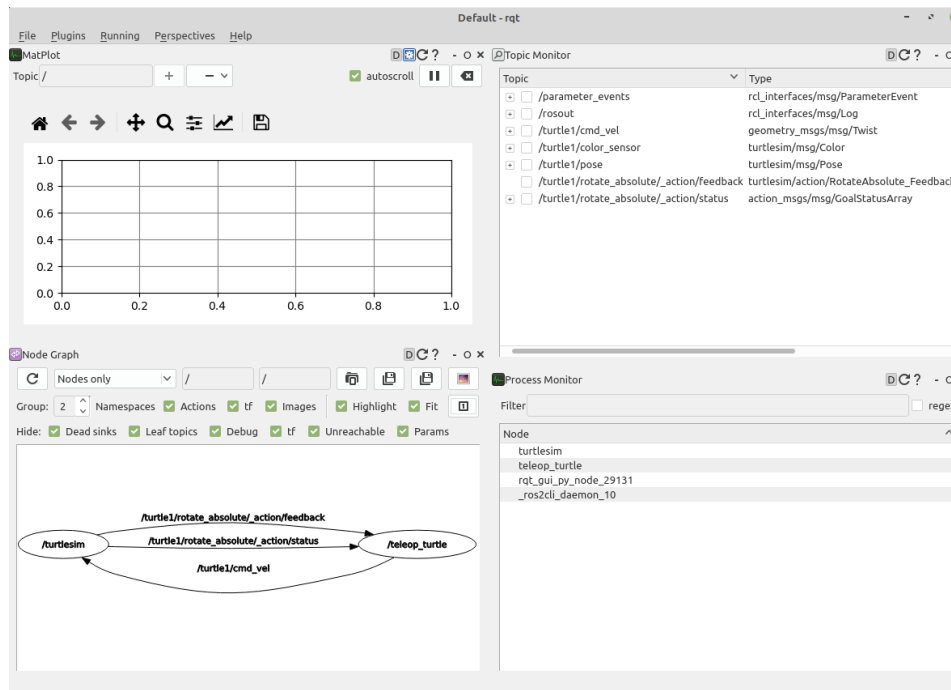
- 명령어 기반의 도구로 로봇 Access 및 거의 모든 ROS 기능을 다룸
 - ✓ 개발환경 및 빌드, 테스트 툴(colcon)
 - ✓ 데이터를 기록, 재생, 관리하는 툴(ros2bag)
 - ✓ colcon, ros2bag 외 20여 가지



```
pyo@robot: ~  
File Edit View Search Terminal Help  
pyo@robot:~$ ros2 extension_points -a  
ros2action.verb: The extension point for 'action' verb extensions  
ros2bag.verb: The extension point for 'bag' verb extensions  
ros2cli.command: The extension point for 'command' extensions  
ros2cli.daemon.verb: The extension point for 'daemon' verb extensions  
ros2component.verb: The extension point for 'node' verb extensions  
ros2doctor.verb: The extension point for 'doctor' verb extensions  
ros2interface.verb: Extension point for 'interface' verb extensions  
ros2lifecycle.verb: The extension point for 'lifecycle' verb extensions  
ros2multicast.verb: The extension point for 'msg' verb extensions  
ros2node.verb: The extension point for 'node' verb extensions  
ros2param.verb: The extension point for 'param' verb extensions  
ros2pkg.verb: The extension point for 'pkg' verb extensions  
ros2service.verb: The extension point for 'service' verb extensions  
ros2topic.verb: The extension point for 'topic' verb extensions  
sros2.verb: The extension point for 'security' verb extensions  
pyo@robot:~$
```

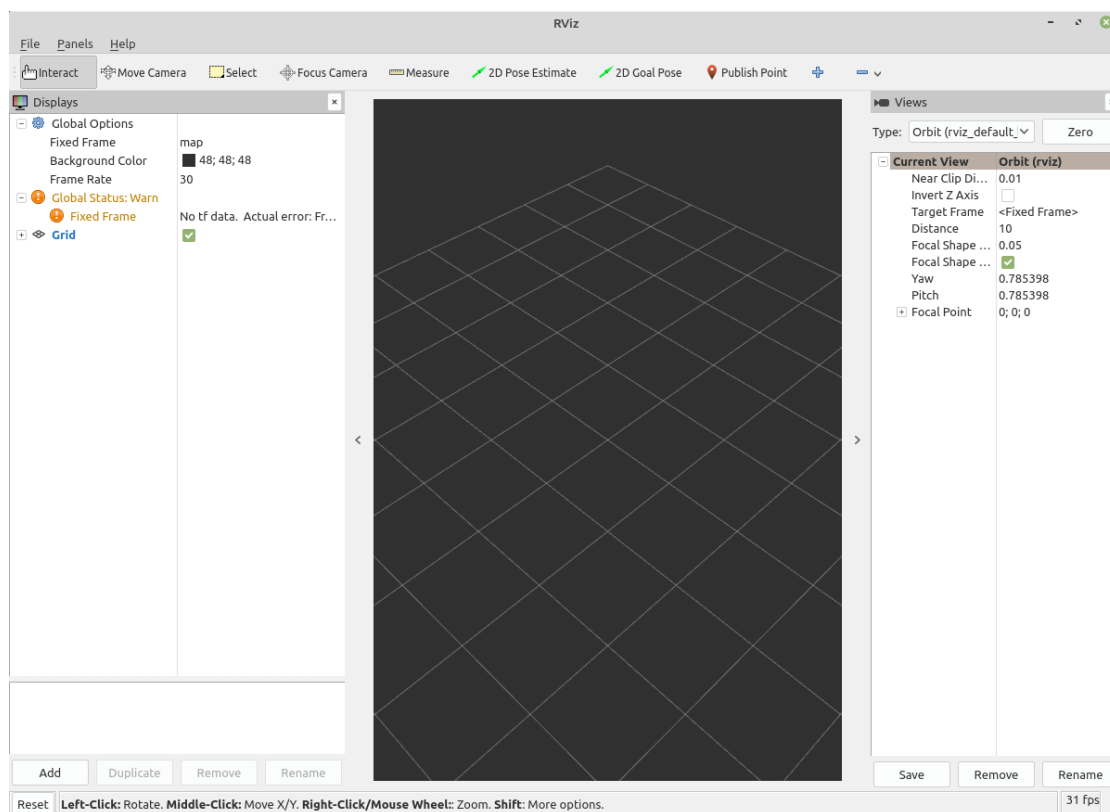
❖ GUI기반 RQt

- 그래픽 인터페이스 개발을 위한 Qt 기반 프레임워크 제공
 - ✓ 노드 간 결 정보 표시(rqt_graph)
 - ✓ 속도, 전압 또는 시간에 따른 데이터 변화 표시(rqt_plot)
 - ✓ rqt_graph, rqt_plot 외 30여 가지



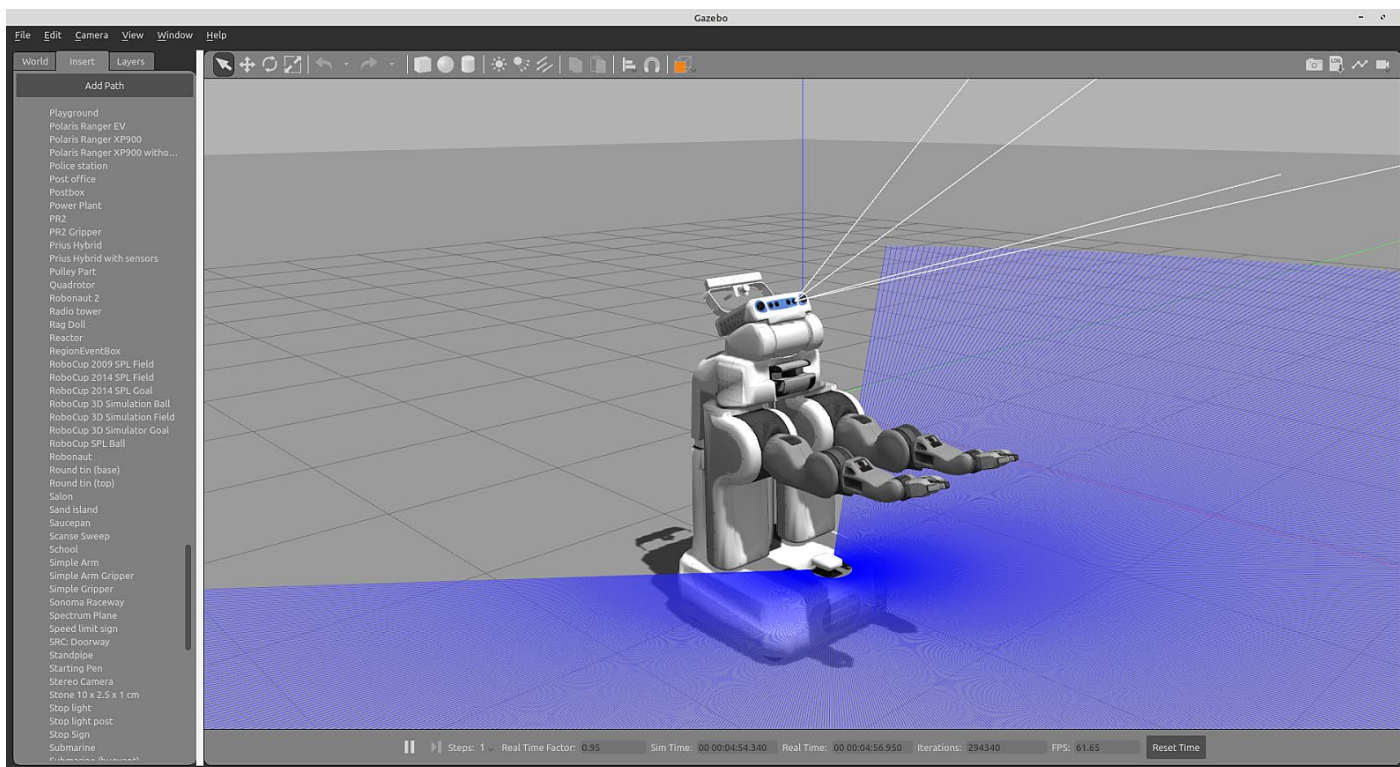
❖ RViz

- 3차원 시각화 도구
 - ✓ 레이저, 카메라 등의 센서 데이터를 시각화
 - ✓ 로봇 외형과 계획된 동작을 표현



❖ Gazebo

- 3차원 시뮬레이터
 - ✓ 물리 엔진을 탑재, 로봇, 센서, 환경 모델 등을 지원
 - ✓ 타 시뮬레이터 대비 ROS와의 높은 호환성



❖ Topic 구독 코드 작성

- ROS2 CLI는 Command Line Interface의 약자로 터미널창에서 지정 텍스트를 입력하여 사용하는 명령어를 칭함
- ros2cli 개발 작업은 ROS 2 TSC의 멤버이며, Ubuntu의 개발사인 Canonical와 TSC 멤버이자 ROS2 Tooling Working Group의 리딩 역할을 맡고 있는 AWS RoboMaker의 개발자들이 참여하고 있음

> action	> bag	> component	> daemon	> doctor
> extension_points	> extensions	> interface	> launch	> lifecycle
> Multicast	> node	> param	> pkg	> run
> Security	> service	> topic	> wtf	

❖ ROS2 CLI 사용법

- 기본적으로는 터미널 창에서 각 명령어가 하나씩 수행되는 방식이며 터미널창에 다음과 같이 `ros2`를 시작으로 verbs와 sub-verbs 그리고 options, arguments 들을 입력하는 방식

```
ros2 [verbs] [sub-verbs] [options] [arguments]
```

- `ros2`는 ROS2 CLI만의 고유한 `entry-point`로 다른 명령어들과 구분
- [verbs] `run`, `launch`, `node` 와 같이 특정 도구를 선택
- [sub-verbs] (예: `node`인 경우 `info`, `list`)를 지정
- 그 뒤에는 선택적으로 옵션 및 매개변수를 입력하여 특정 명령을 수행

❖ ROS2 CLI 종류 및 각 Sub-Verbs 기능

- 모든 명령어를 외울 필요는 없으며 사용법만 기억하면 됨
- 명령어를 확인 하는 방법은 ROS2 CLI를 입력한 후 space를 누르고 tab키를 누르면 사용 가능한 명령어가 나열됨
- Verb, Sub-verb 마찬가지로 tab키를 사용하면 확인할 수 있으며, -h 옵션으로 내용 확인 가능

```
$ ros2 [tab] [tab] [tab] [tab] ...
```

```
$ ros2 -h  
$ ros2 node -h  
$ ros2 node info -h
```

❖ ROS2 CLI 실행 명령어

ros2cli + [verbs]	[arguments]	기능
ros2 run	<package> <executable>	특정 패키지의 특정 노드 실행 (1개의 노드) * executable에 따라 복수 노드도 실행 가능
ros2 launch	<package> <launch-file>	특정 패키지의 특정 런치 파일 실행 (0개~복수개의 노드)

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 launch demo_nodes_cpp talker_listener.launch.py
```

❖ ROS2 정보 조회 명령어

ros2cli + [verbs]	[sub-verbs]	기능
ros2 pkg	create executables list prefix xml	새로운 ROS 2 패키지 생성 지정 패키지의 실행 파일 목록 출력 사용 가능한 패키지 목록 출력 지정 패키지의 저장 위치 출력 지정 패키지의 패키지 정보 파일(xml) 출력
ros2 node	info list	실행 중인 노드 중 지정한 노드의 정보 출력 실행 중인 모든 노드의 목록 출력
ros2 topic	bw delay echo find hz info list pub type	지정 토픽의 대역폭 측정 지정 토픽의 지연시간 측정 지정 토픽의 데이터 출력 지정 타입을 사용하는 토픽 이름 출력 지정 토픽의 주기 측정 지정 토픽의 정보 출력 사용 가능한 토픽 목록 출력 지정 토픽의 토픽 발행 지정 토픽의 토픽 타입 출력

❖ ROS2 정보 조회 명령어

ros2 service	call find list type	지정 서비스의 서비스 요청 전달 지정 서비스 타입의 서비스 출력 사용 가능한 서비스 목록 출력 지정 서비스의 타입 출력
ros2 action	info list send_goal	지정 액션의 정보 출력 사용 가능한 액션 목록 출력 지정 액션의 액션 목표 전송
ros2 interface	list package packages proto show	사용 가능한 모든 인터페이스 목록 출력 특정 패키지에서 사용 가능한 인터페이스 목록 출력 인터페이스 패키지들의 목록 출력 지정 패키지의 프로토타입 출력 지정 인터페이스의 데이터 형태 출력
ros2 param	delete describe dump get list set	지정 파라미터 삭제 지정 파라미터 정보 출력 지정 파라미터 저장 지정 파라미터 읽기 사용 가능한 파라미터 목록 출력 지정 파라미터 쓰기
ros2 bag	info play record	저장된 rosbag 정보 출력 rosbag 기록 rosbag 재생

❖ ROS2 기능 보조 명령어

ros2cli + [verbs]	[sub-verbs] (options)	기능
ros2 extensions	(-a) (-v)	ros2cli의 extension 목록 출력
ros2 extension_points	(-a) (-v)	ros2cli의 extension point 목록 출력
ros2 daemon	start status stop	daemon 시작 daemon 상태 보기 daemon 중지
ros2 multicast	receive send	multicast 수신 multicast 전송
ros2 doctor	hello (-r) (-rf) (-iw)	ROS 설정 및 네트워크, 패키지 버전, rmw 미들웨어 등과 같은 잠재적 문제를 확인하는 도구
ros2 wtf	hello (-r) (-rf) (-iw)	doctor와 동일함 (ros2 doctor의 alias) (WTF: Where's The Fire)

❖ ROS2 기능 보조 명령어

ros2 lifecycle	get list nodes set	라이프사이클 정보 출력 지정 노드의 사용 가능한 상태전이 목록 출력 라이프사이클을 사용하는 노드 목록 출력 라이프사이클 상태 전환 트리거
ros2 component	list load standalone types unload	실행 중인 컨테이너와 컴포넌트 목록 출력 지정 컨테이너 노드의 특정 컴포넌트 실행 표준 컨테이너 노드로 특정 컴포넌트 실행 사용 가능한 컴포넌트들의 목록 출력 지정 컴포넌트의 실행 중지
ros2 security	create_key create_keystore create_permission generate_artifacts generate_policy list_keys	보안키 생성 보안키 저장소 생성 보안 허가 파일 생성 보안 정책 파일을 이용하여 보안키 및 보안 허가 파일 생성 보안 정책 파일(policy.xml) 생성 보안키 목록 출력

❖ rqt, Rviz, Gazebo

- rqt의 경우 앞선 강의에서 실행 및 사용하는 방법에 대해 간단히 알아보았음
- 이 후 진행되는 수업에서 실습과 함께 사용법에 대해 알아볼 예정
- Rviz, Gazebo 역시 ROS의 기본적인 사항에 대해 먼저 공부한 후 알아볼 예정임

2 ROS2 물리량 표현

❖ ROS2 표준단위 필요성

- ROS2 인터페이스에서 설명했던 msg, service, action 인터페이스는 노드 간에 데이터를 교환하며, 어떤 `이름`으로 어떠한 `자료형`을 사용하는 지 기술되어 있음
- geometry_msgs/msg/Twist 메시지 형태는 float64 자료형의 linear.x, linear.y, linear.z, angular.x, angular.y, angular.z 값을 사용
 - ✓ 단위가 다르게 사용될 경우 오동작이나 사고와 같은 문제가 발생
 - ✓ 단위 변환 프로그램을 추가로 넣는 방법도 있지만, 데이터 변환으로 인해 불필요한 계산이 발생
- ROS 개발 초기부터 이러한 단위 불일치로 인한 문제를 줄이기 위해 표준 단위에 관한 규칙을 세움

❖ ROS2 표준단위

- SI 단위는 총 7개, SI 유도 단위는 20개
- 로봇공학에서 주로 사용하는 단위는 다음 표와 같음
- 각 단위의 조합 역시 이러한 규칙을 따름. 예) 병진 속도는 m/s, 회전 속도는 rad/s

물리량	단위 (SI unit)	물리량	단위 (SI derived unit)
length (길이)	meter (m)	angle (평면각)	radian (rad)
mass (질량)	kilogram (kg)	frequency (주파수)	hertz (Hz)
time (시간)	second (s)	force (힘)	newton (N)
current (전류)	ampere (A)	power (일률)	watt (W)
		voltage (전압)	volt (V)
		temperature (온도)	celsius (°C)
		magnetism (자기장)	tesla (T)

❖ ROS2 좌표 표현 통일의 필요성

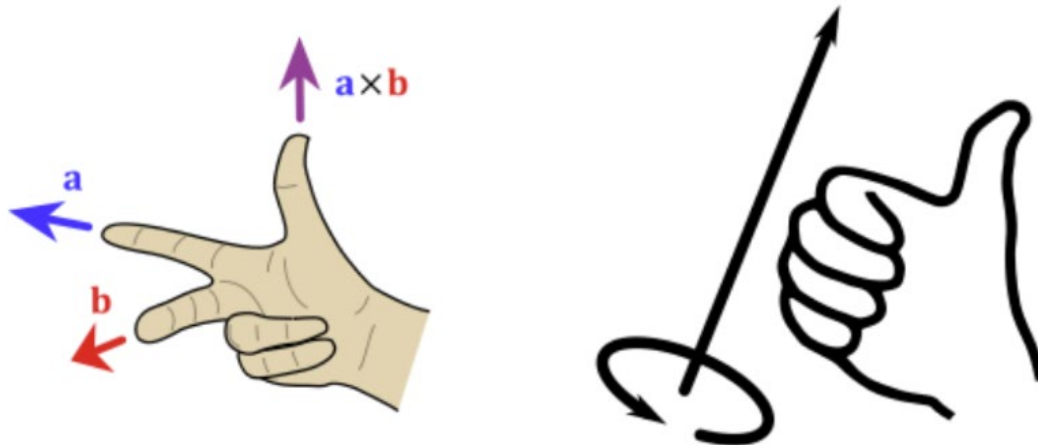
- ROS2 인터페이스에서 설명했던 msg, service, action 인터페이스는 노드 간에 데이터를 교환하며, 어떤 `이름`으로 어떠한 `자료형`을 사용하는 지 기술되어 있음
- geometry_msgs/msg/Twist 메시지 형태는 float64 자료형의 linear.x, linear.y, linear.z, angular.x, angular.y, angular.z 값을 사용
 - ✓ 단위가 다르게 사용될 경우 오동작이나 사고와 같은 문제가 발생
 - ✓ 단위 변환 프로그램을 추가로 넣는 방법도 있지만, 데이터 변환으로 인해 불필요한 계산이 발생
- ROS 개발 초기부터 이러한 단위 불일치로 인한 문제를 줄이기 위해 표준 단위에 관한 규칙을 세움

❖ ROS2 좌표 표현 통일의 필요성

- 카메라의 경우, 컴퓨터 비전 분야에서 널리 사용되는 z forward, x right, y down 기본 좌표계 사용
- 로봇은 기본적으로 x forward, y left, z up를 기본 좌표계로 사용
- 이 외에도 로봇에서 사용하는 LiDAR 및 IMU, Torque 센서들도 제조사별로 서로 다른 좌표계를 사용할 수도 있으며 좌표 변환 API가 있더라도 기본 출력을 어떤 표현을 사용하고 있는지에 대해 확인해야 함

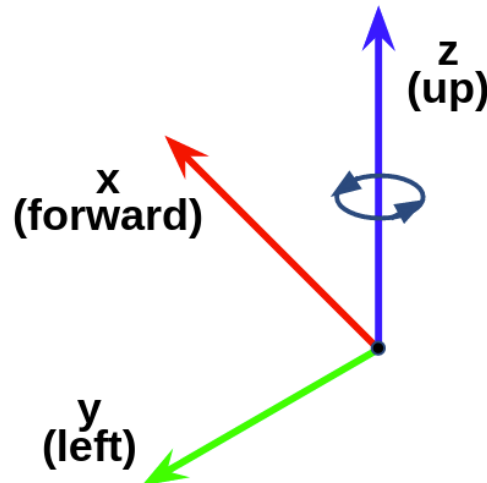
❖ ROS2 좌표 표현의 기본 규칙

- OS 커뮤니티에서는 모든 좌표계를 삼차원 벡터 표기관습을 이해하기 위한 일반적인 기억법인 오른손 법칙에 따라 표현
- 기본적으로 회전 축의 경우에는 검지, 중지, 엄지를 축을 사용하며 오른손의 손가락을 감는 방향이 정회전(+) 방향임
- 예를 들어 로봇이 제자리에서 시계 12시 방향에서 9시 방향으로 회전하였다면 로봇은 $+1.5708$ rad 만큼 회전



❖ ROS2 좌표 표현의 축(Axis Orient)

- x forward, y left, z up 을 기본으로 사용
- 시각화 도구 RViz나 3차원 시뮬레이터 Gazebo에서 이러한 기본 3축의 표현에 있어서 혼란이 발생하지 않도록 RGB의 원색으로 표현
 - ✓ Red는 x 축, Green은 y축, Blue는 z축
- ✓ 나머지 상세 좌표는 사용 시 자세히 설명하기로 함



❖ ROS2 시계와 시간

- 다수의 센서를 사용하는 로봇은 시간에 따른 각 센서 값의 변화량과 그 센서들 간의 시간 동기화가 매우 중요
- ROS2는 여러 노드들이 서로 통신하며 다양한 정보(센서 값, 알고리즘을 수행한 결괏값 등)들을 주고 받기 때문에 해당 정보들이 발간된 정확한 시간이 필수
- ROS2에서는 발간되는 토픽에 주요 데이터 뿐만 아니라 해당 토픽이 발간되는 시간을 함께 포함
- stamp, frame id 를 포함하고 있는 std_msgs/msg/header 데이터 타입은 ROS2에서 제공하는 표준 메시지 타입 중의 하나로서 sensor_msgs, geometry_msgs, nav_msgs 등 ROS2의 기본 메시지 타입 대부분에 포함되어 있음
- ROS2에서 사용하는 기본 시계는 System Clock이며 rclpy 는 time모듈을 캡슐화하여 사용. System Clock 은 국제 표준시인 UTC로 표시

❖ ROS2 시간의 추상화

- ROS2에서는 기본 시계 이외에도 프로그램 실행과 관련된 시간을 제어하며 동작하는 추상적인 개념의 시계로도 사용 가능
 - 특정 시점으로의 시간 이동, 시간의 흐름 속도 제어, 시간 정지 등의 기능을 가지고 있음
 - 이 기능은 시계는 과거에 기록한 데이터를 다룰 때(ros2bag)나 로봇 시뮬레이션(gazebo)에서 사용할 수 있으며, 이를 통해 보다 효율적으로 디버깅할 수 있음

rcl/time.h

- 이를 위해 ROS2는 시간을 추상화 하였고, 총 세가지의 시간을 제공

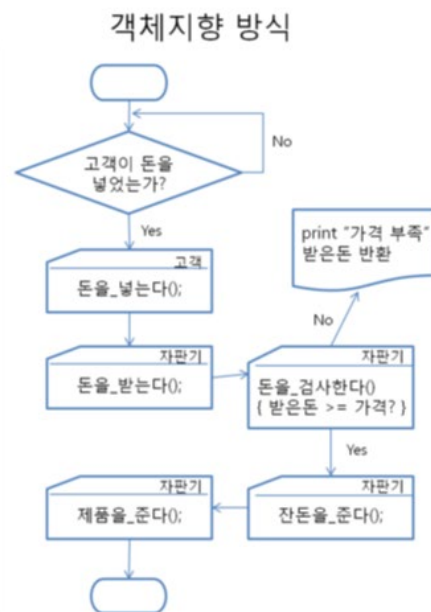
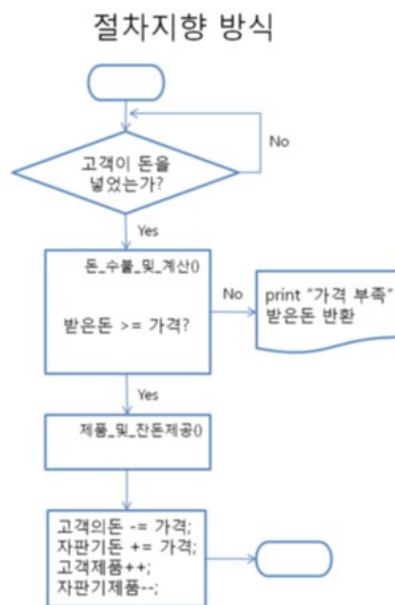
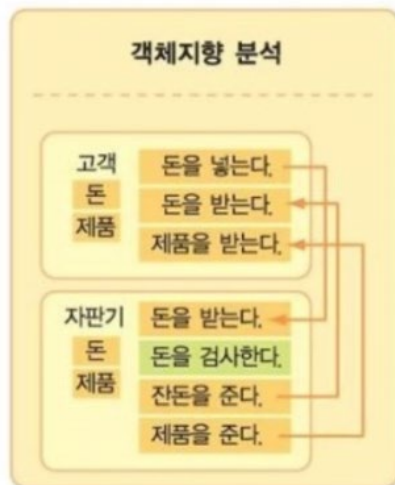
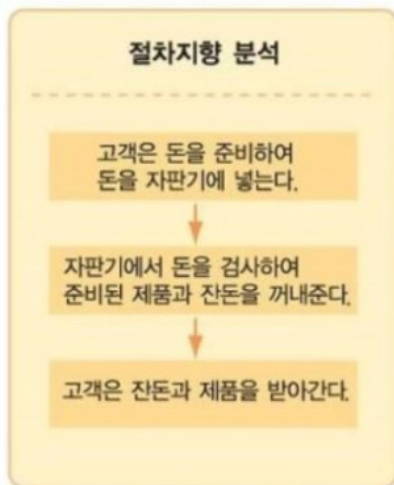
```
enum rcl_time_source_type_t
{
    RCL_TIME_SOURCE_UNINITIALIZED = 0,
    RCL_ROS_TIME,
    RCL_SYSTEM_TIME,
    RCL_STEADY_TIME
};
```

3

ROS2 이해를 위한 Class

❖ Python과 Class

- ROS2와 마찬가지로 Python 코드는 대부분 클래스로 구현되어 있음
- 지금까지는 객체지향 프로그래밍이 아닌 절차지향적 프로그래밍 기법으로 코드를 작성하였음



❖ Jupyter notebook으로 따라하기

```
pip3 install matplotlib
```

- Jupyter notebook 파일 생성 후 다음과 같은 코드 추가

```
import matplotlib.pyplot as plt
import numpy as np
```

[1] ✓ 0.5s Python

- 실행 후 이상이 없는가 확인

❖ Jupyter notebook으로 따라하기

- Numpy를 이용하여 0에서 6까지 0.01간격으로 배열 생성

```

t = np.arange(0, 6, 0.01)
t

```

[2] ✓ 0.0s Python

```

... array([0. , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
         0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
         0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
         0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
         0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
         0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,

```

- Sin 함수를 구한 후 shape을 이용하여 배열의 크기 확인

• sin() function

+ Code + Markdown

```

y = np.sin(np.pi * t)
y.shape

```

[4] ✓ 0.0s Python

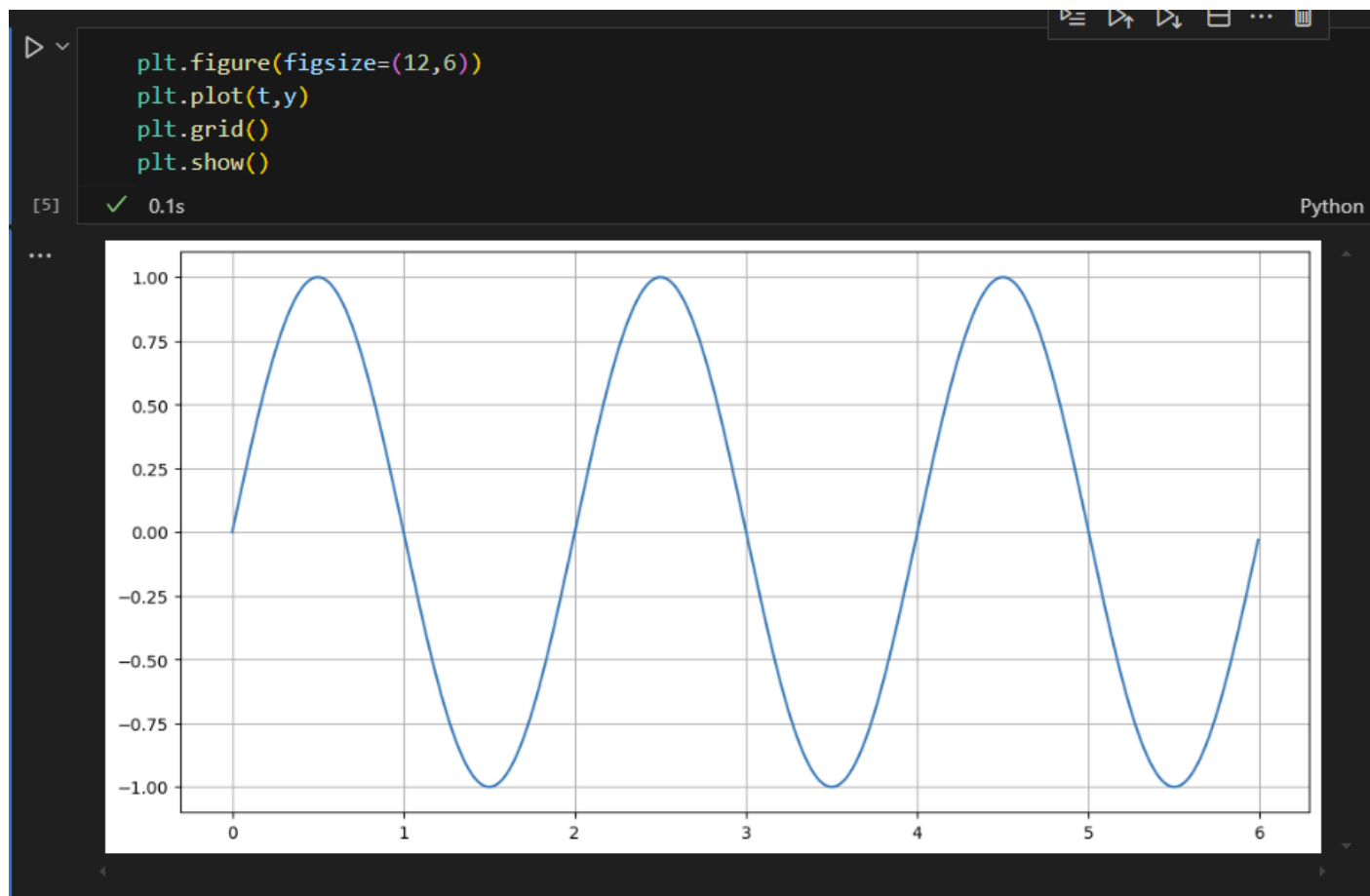
```

... (600,)

```

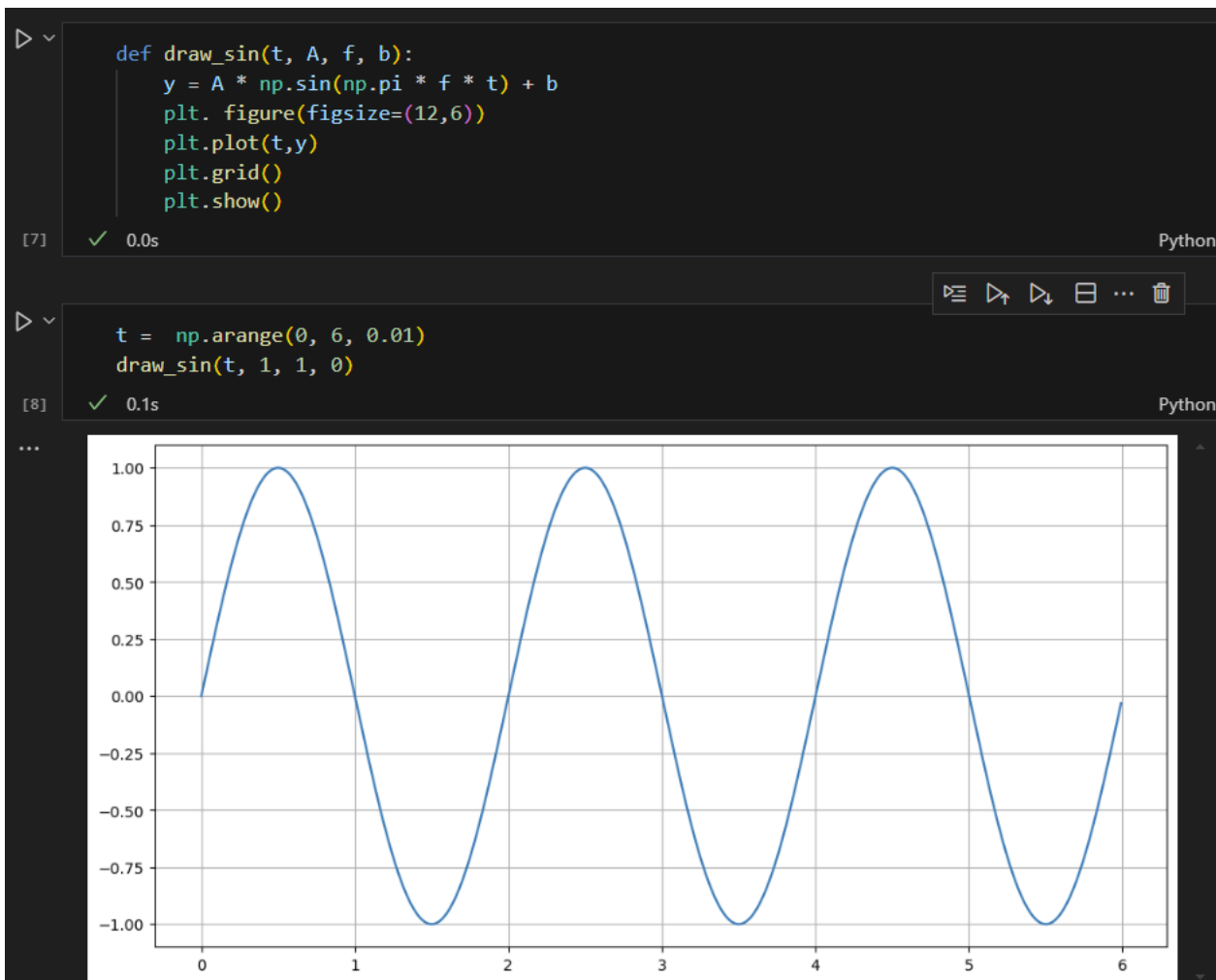
❖ Jupyter notebook으로 따라하기

- 그래프 그리기



❖ Jupyter notebook으로 따라하기

- 함수로 구현하기



❖ Class로 sin함수 구현

- DrawSin 클래스 선언

```
class DrawSin :  
    def __init__(self, amp, freq, bias, end_time) :  
        self.amp = amp  
        self.freq = freq  
        self.bias = bias  
        self.end_time = end_time
```

[9] ✓ 0.0s Python

- 클래스의 생성자를 이용한 초기값 할당 및 값 확인

```
tmp = DrawSin(1,1,0,3)  
tmp.__dict__
```

[10] ✓ 0.0s Python

```
... {'amp': 1, 'freq': 1, 'bias': 0, 'end_time': 3}
```

❖ Class로 sin함수 구현

- 삼각함수를 그리는 클래스 완성하기

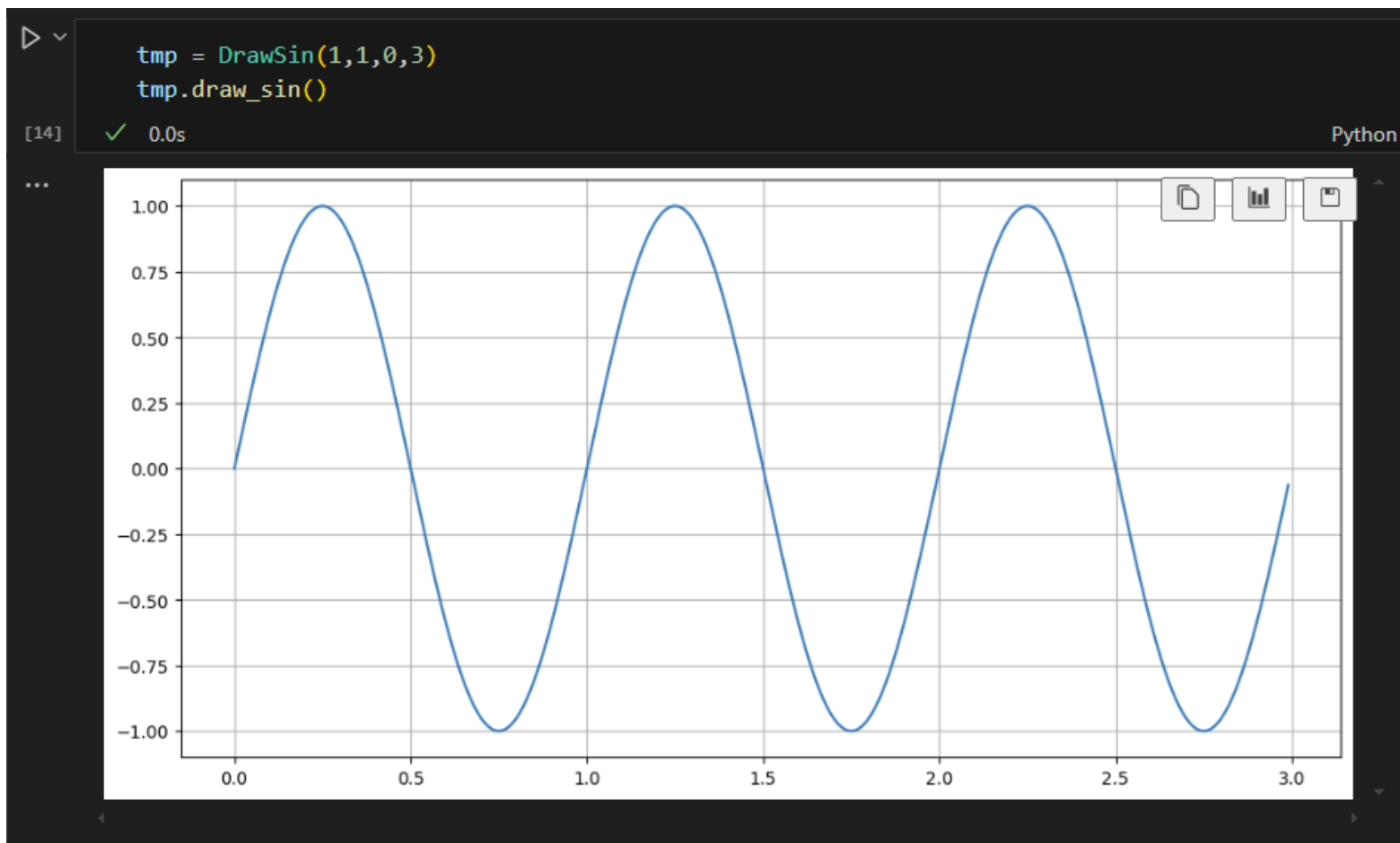
```
class DrawSin :  
    def __init__(self, amp, freq, bias, end_time) :  
        self.amp = amp  
        self.freq = freq  
        self.bias = bias  
        self.end_time = end_time  
  
    def calc_sin(self):  
        self.t = np.arange(0, self.end_time, 0.01)  
        return self.amp * np.sin(2*np.pi*self.freq*self.t) + self.bias  
  
    def draw_sin(self):  
        y = self.calc_sin()  
        plt.figure(figsize=(12,6))  
        plt.plot(self.t, y)  
        plt.grid()  
        plt.show()
```

✓ 0.0s

Python

❖ Class로 sin함수 구현

- 클래스로 sin함수 그리기



❖ Class로 sin함수 구현

- 클래스 내부 데이터 변경 후 다시 그래프 그려보기 실습

```
tmp.amp = 0.5  
tmp.draw_sin()
```

```
tmp.freq = 0.5  
tmp.draw_sin()
```

```
tmp.end_time = 10  
tmp.draw_sin()
```

❖ Class로 상속을 이용한 Cos함수 그리기

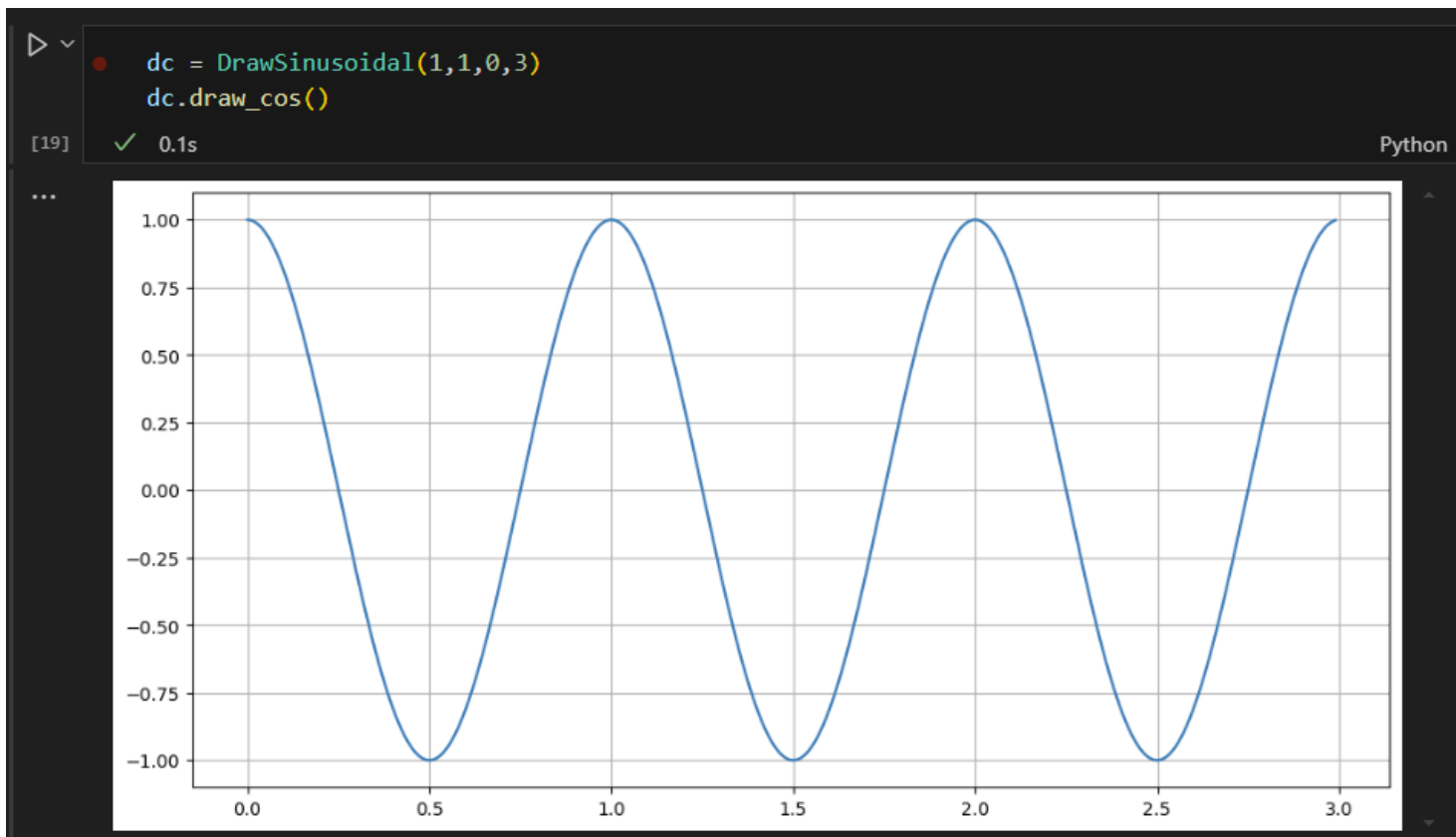
- DrawSin 클래스를 상속 받아서 만들기

```
class DrawSinusoidal(DrawSin):  
    def calc_cos(self):  
        self.t = np.arange(0, self.end_time, 0.01)  
        return self.amp * np.cos(2 * np.pi * self.freq * self.t) + self.bias  
  
    def draw_cos(self):  
        y = self.calc_cos()  
        plt.figure(figsize=(12,6))  
        plt.plot(self.t, y)  
        plt.grid()  
        plt.show()
```

[18] ✓ 0.0s Python

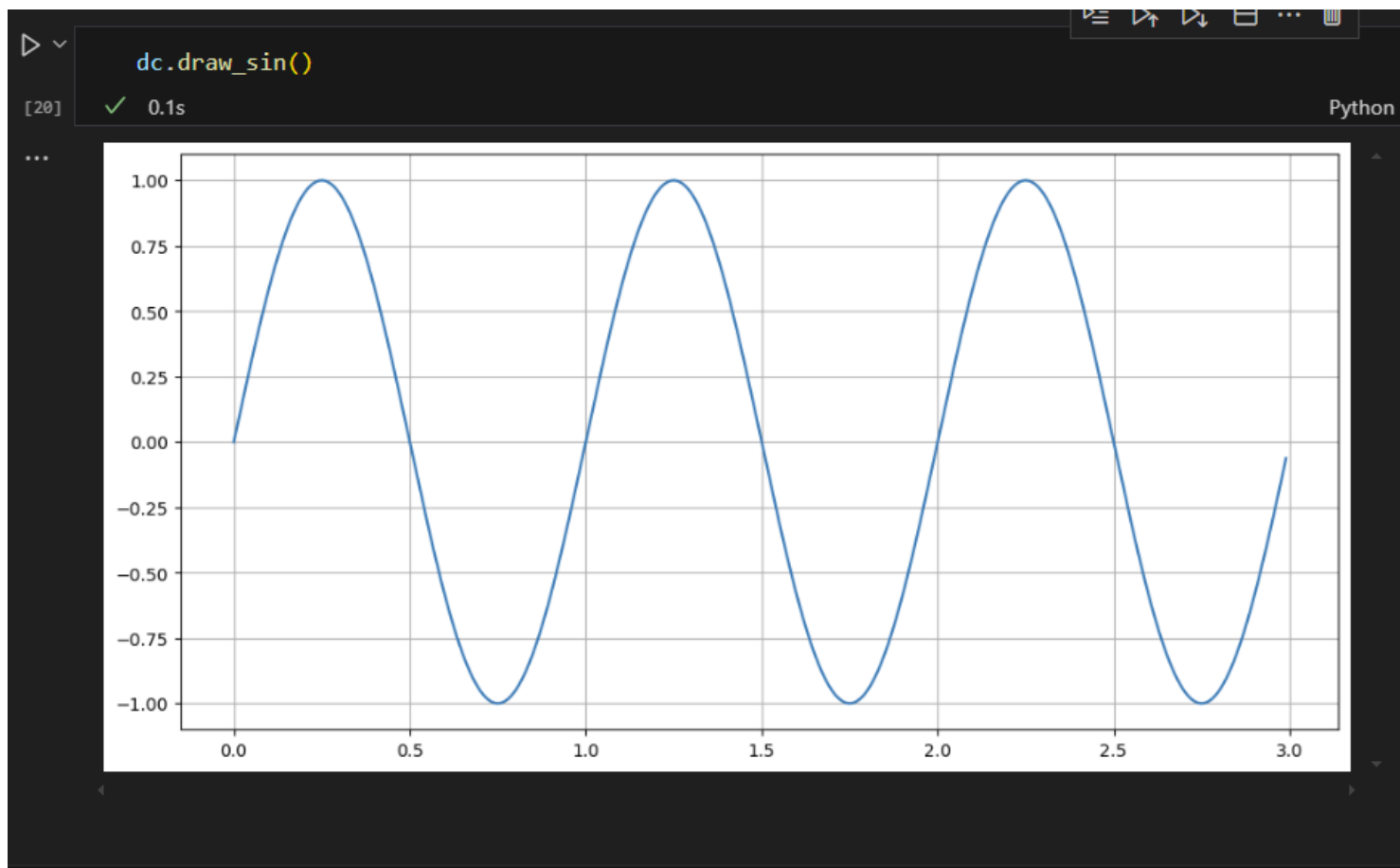
❖ Class로 상속을 이용한 Cos함수 그리기

- dc 객체 생성 후 cos함수 그리기



❖ Class로 상속을 이용한 Cos함수 그리기

- dc 객체가 상속 받은 Sin함수 그리기



❖ 메서드 오버라이딩

- 오버라이딩을 통해 제목 및 각 축에 대한 설명 추가

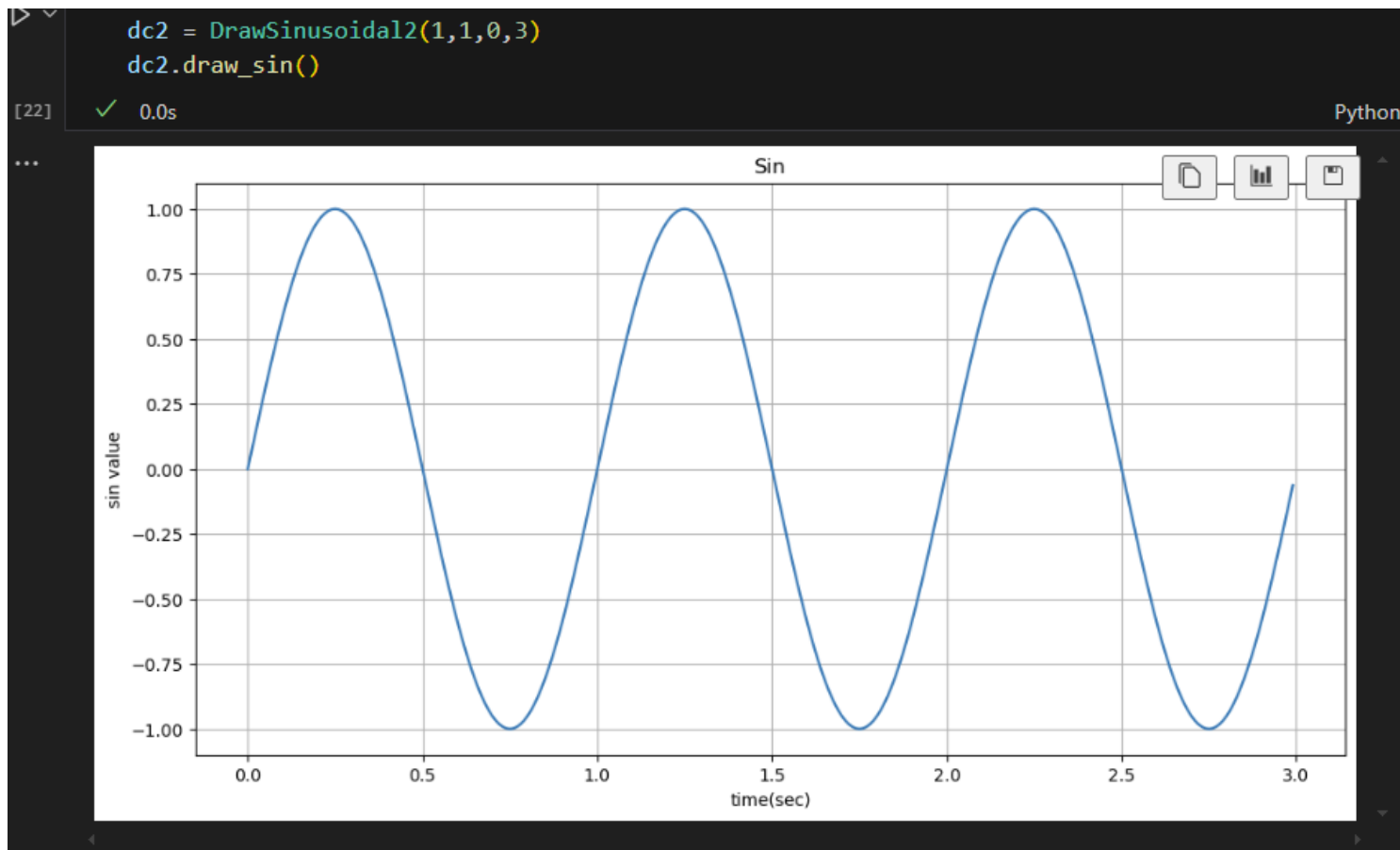
```
class DrawSinusoidal2(DrawSinusoidal):  
    def draw_sin(self):  
        y = self.calc_sin()  
        plt.figure(figsize=(12,6))  
        plt.plot(self.t, y)  
        plt.title('Sin')  
        plt.ylabel('sin value')  
        plt.xlabel('time(sec)')  
        plt.grid()  
        plt.show()
```

[21] ✓ 0.0s

Python

❖ 메서드 오버라이딩

- dc2 객체 생성 및 그래프 그리기



❖ 클래스의 Super() 사용 → 부모 클래스 호출

- Sinusoidal 함수를 그리는 클래스를 만드는 동안 시간 간격(Time Span)은 0.01fh 항상 동일하게 사용해 왔음
- DrawSinusoidal3 클래스에 시간 간격을 ts라는 변수로 설정하여 처리하도록 변경

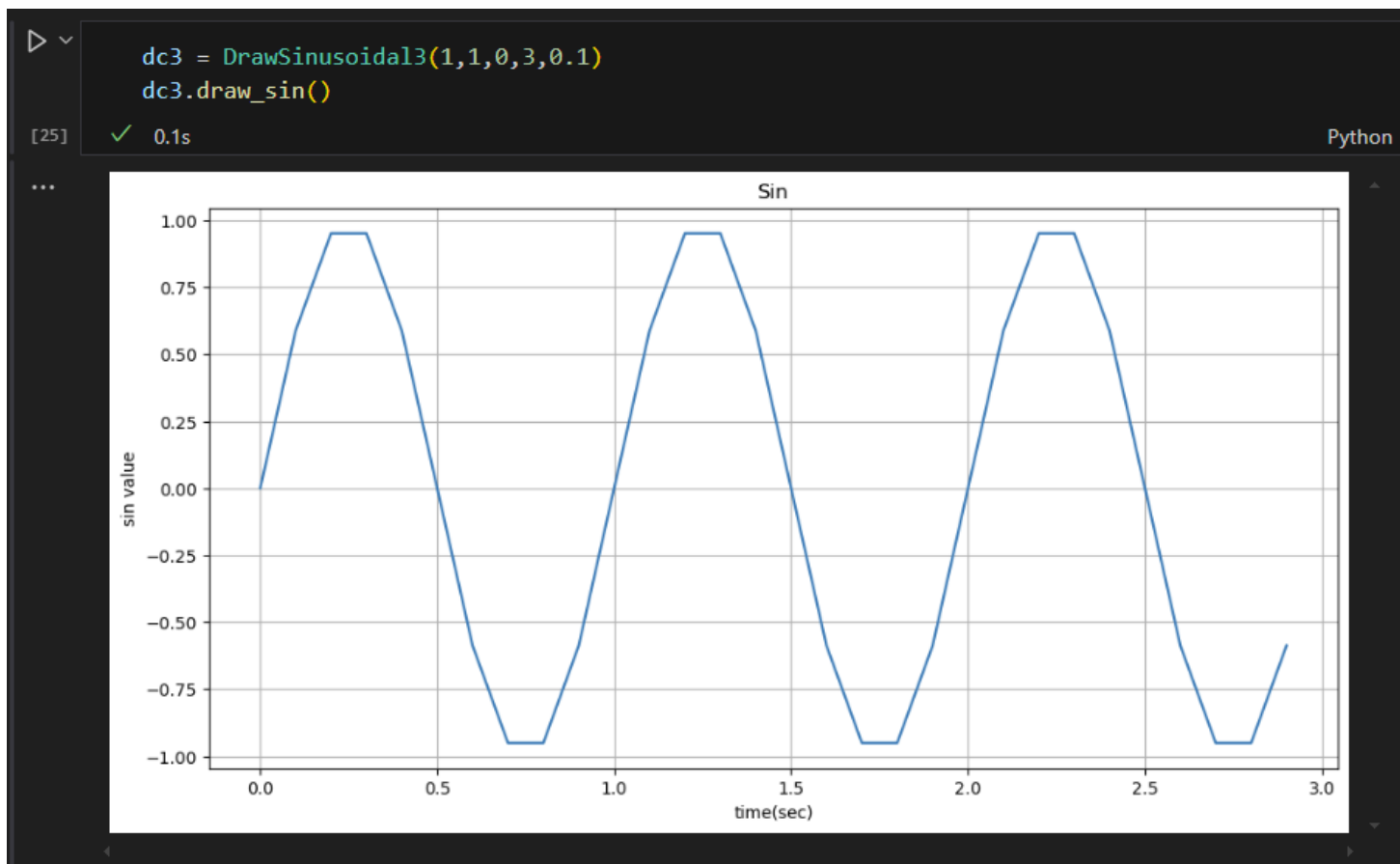
```
class DrawSinusoidal3(DrawSinusoidal2):  
    def __init__(self, amp, freq, bias, end_time, ts):  
        super().__init__(amp, freq, bias, end_time)  
        self.ts = ts  
  
    def calc_sin(self):  
        self.t = np.arange(0, self.end_time, self.ts)  
        return self.amp * np.sin(2*np.pi*self.freq*self.t) + self.bias  
  
    def draw_sin(self):  
        y = self.calc_sin()  
        plt.figure(figsize=(12,6))  
        plt.plot(self.t, y)  
        plt.title('Sin')  
        plt.ylabel('sin value')  
        plt.xlabel('time(sec)')  
        plt.grid()  
        plt.show()
```

[23] ✓ 0.0s

Python

❖ 클래스의 Super() 사용 → 부모 클래스 호출

- dc3 객체를 생성하여 ts를 0.1로 설정하고 그래프 그리기



❖ Homework

1. 정해진 구역 내에서 랜덤하게 주행하는 거북이를 클래스로 설계하기
2. Turtlesim을 Service Server로 대상을 정하고 5가지 이상의 서비스를 요청할 수 있는 Client를 클래스로 설계하기



Q & A
