Rotation 행렬변환행렬

3D, rotation matrix (Rx, Ry, Rz)

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D, rotation matrix (R)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

OpenPose

신체의 특징점 (관절)을 추론 후, 관절들을 이어 주는 방식

3개의 Skeleton: Body, Hand, Face

(op.datum 헤더파일: poseKeypoints(), faceKeypoints(), handKeypoints()로 관절 추론 가능)

OpenPose 모델 종류

종류 마다 출력 관절 수가 다름.

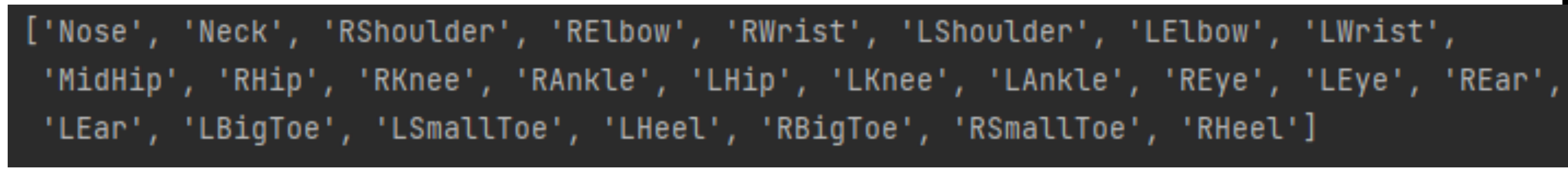
세 종류: BODY-25(25개), COCO(18개), MPII(15개)

BODY-25: 이용 중 (그림 참고)

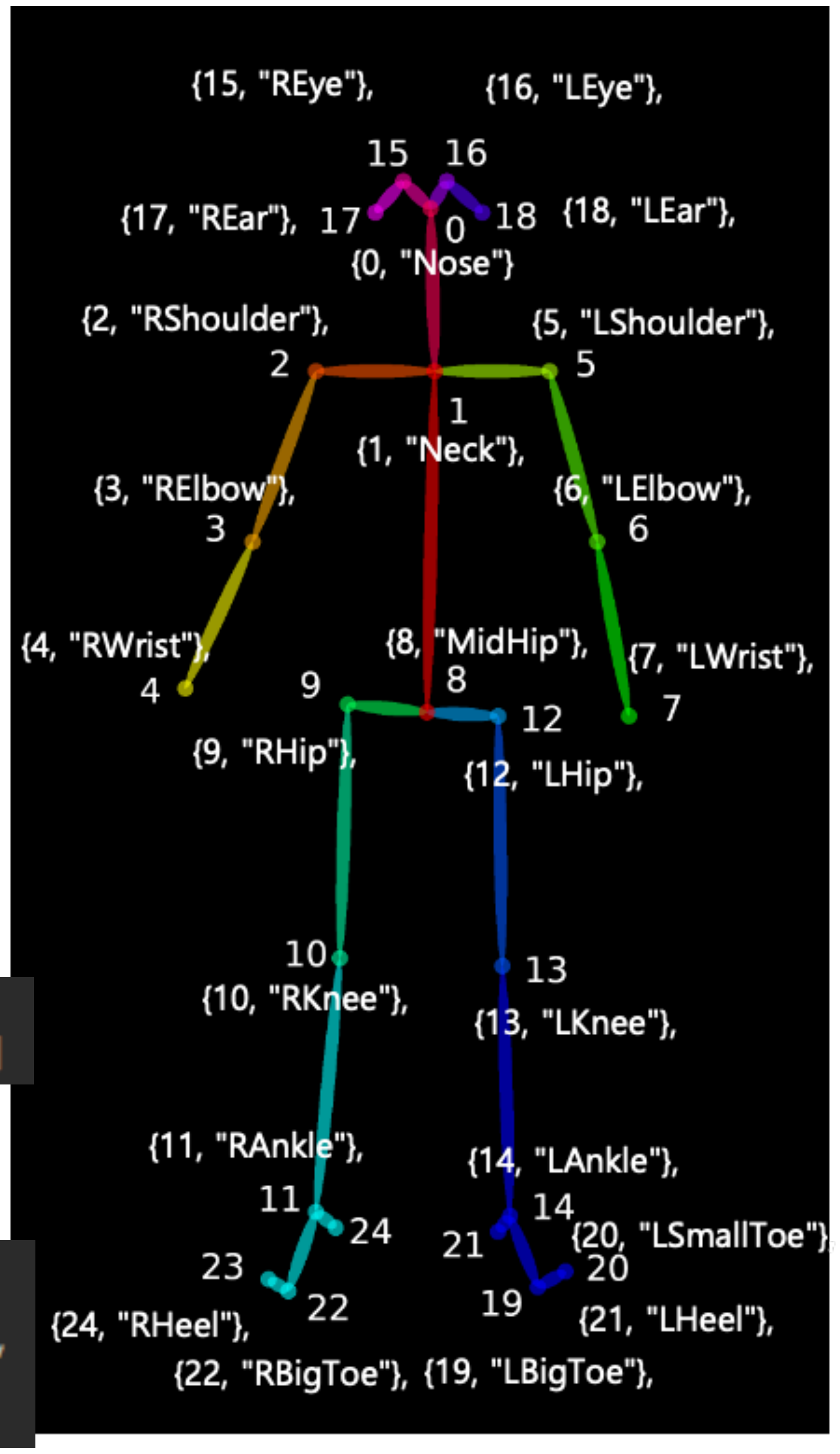
PoseModel

```
class Reader(object):
    BODY_PART_25 = [op.getPoseBodyPartMapping(op.PoseModel.BODY_25)[idx] for idx in range(25)]
```

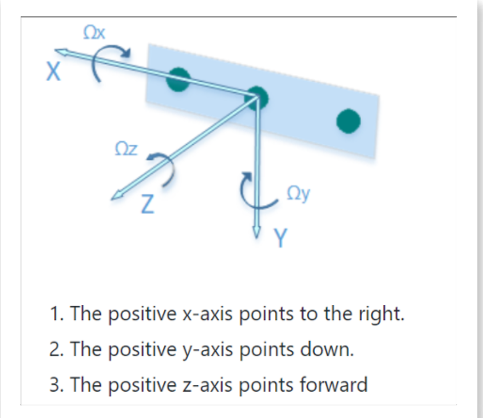
(출력) BODY_PART_25



```
['Nose', 'Neck', 'RShoulder', 'RElbow', 'RWrist', 'LShoulder', 'LElbow', 'LWrist',
 'MidHip', 'RHip', 'RKnee', 'RAnkle', 'LHip', 'LKnee', 'LAnkle', 'REye', 'LEye', 'REar',
 'LEar', 'LBigToe', 'LSmallToe', 'LHeel', 'RBigToe', 'RSmallToe', 'RHeel']
```



skeleton3DView.py



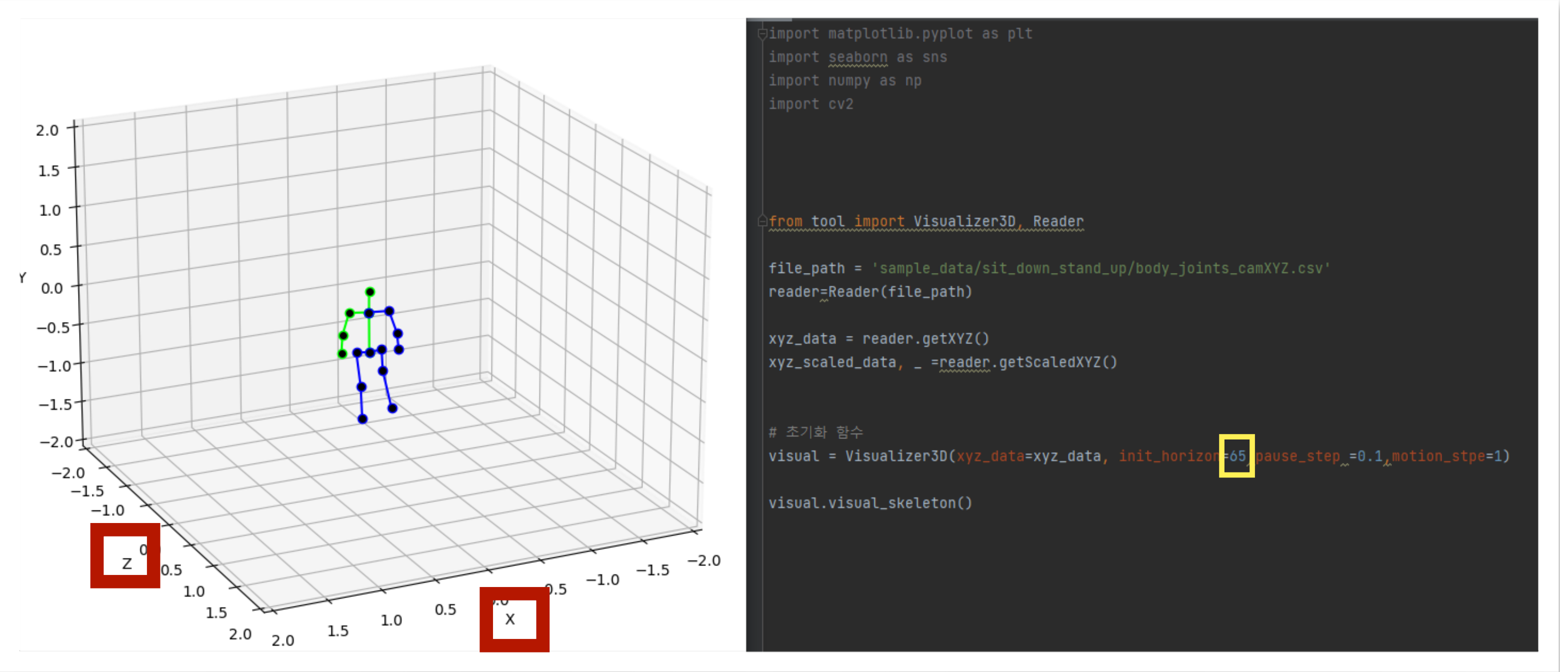
1. The positive x-axis points to the right.
2. The positive y-axis points down.
3. The positive z-axis points forward

5) 원본데이터: (165, 101) body_joint_camXYZ.csv: frame, 관절 part 당 (x,y,z,s)

6) getXYZ() 후: (165, 25, 3) frame수, 관절 25part, (x,y,z) 좌표

8) Visualizer3D()의 파라미터

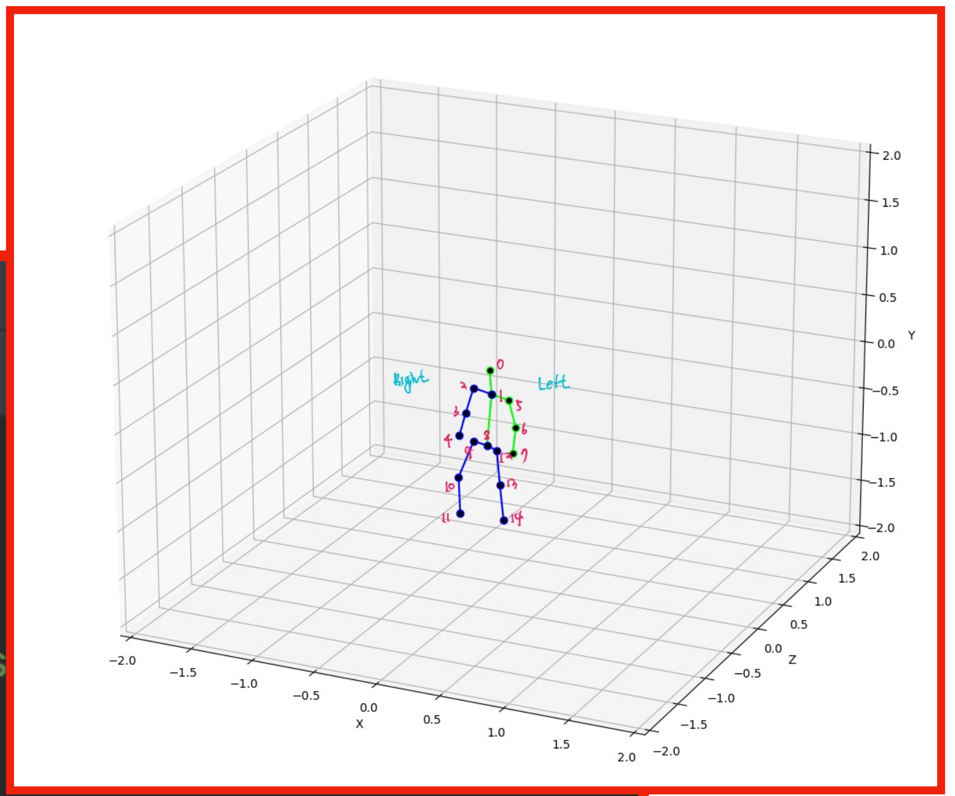
1) xyz_data: body joint 데이터 (스켈레톤)
2) init_horizon: ax.view_init(관찰 각도 지정)
   -65: 양각 65도 기울기, x-z축 반대 (그림 참고: 65일때와 비교)
3) pause_step: (갱신 시간) 0.1초마다 1 frame

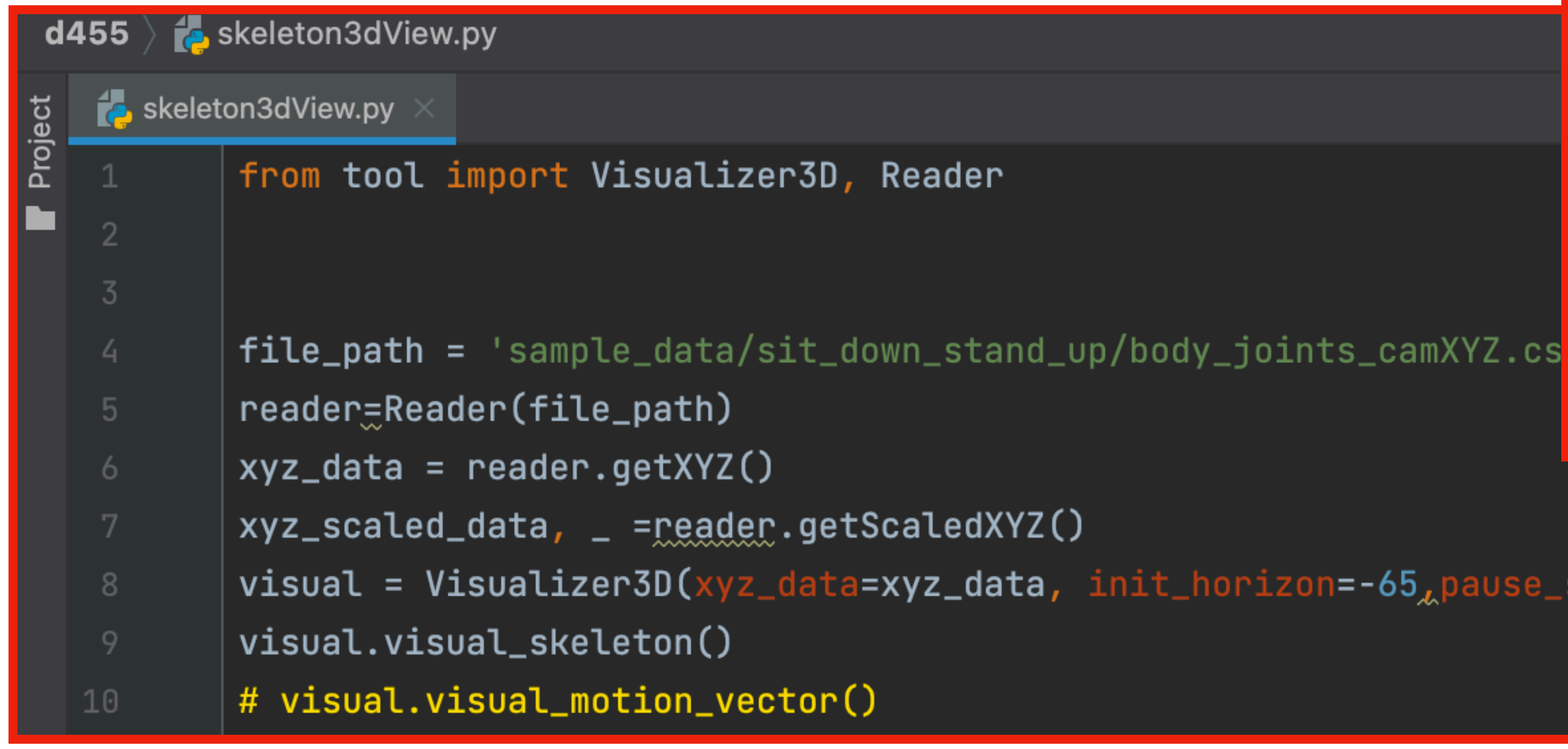


```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import cv2




from tool import Visualizer3D, Reader

file_path = 'sample_data/sit_down_stand_up/body_joints_camXYZ.csv'
reader=Reader(file_path)

xyz_data = reader.getXYZ()
xyz_scaled_data, _ =reader.getScaledXYZ()


# 초기화 함수
visual = Visualizer3D(xyz_data=xyz_data, init_horizon=65,pause_step =0.1,motion_stpe=1)

visual.visual_skeleton()
```

(if init_horizon=65) 값 변화 X, 보는 각도만 바뀜
(스켈레톤 시각화: left joint - right joint 가 뒤집혀 보임)

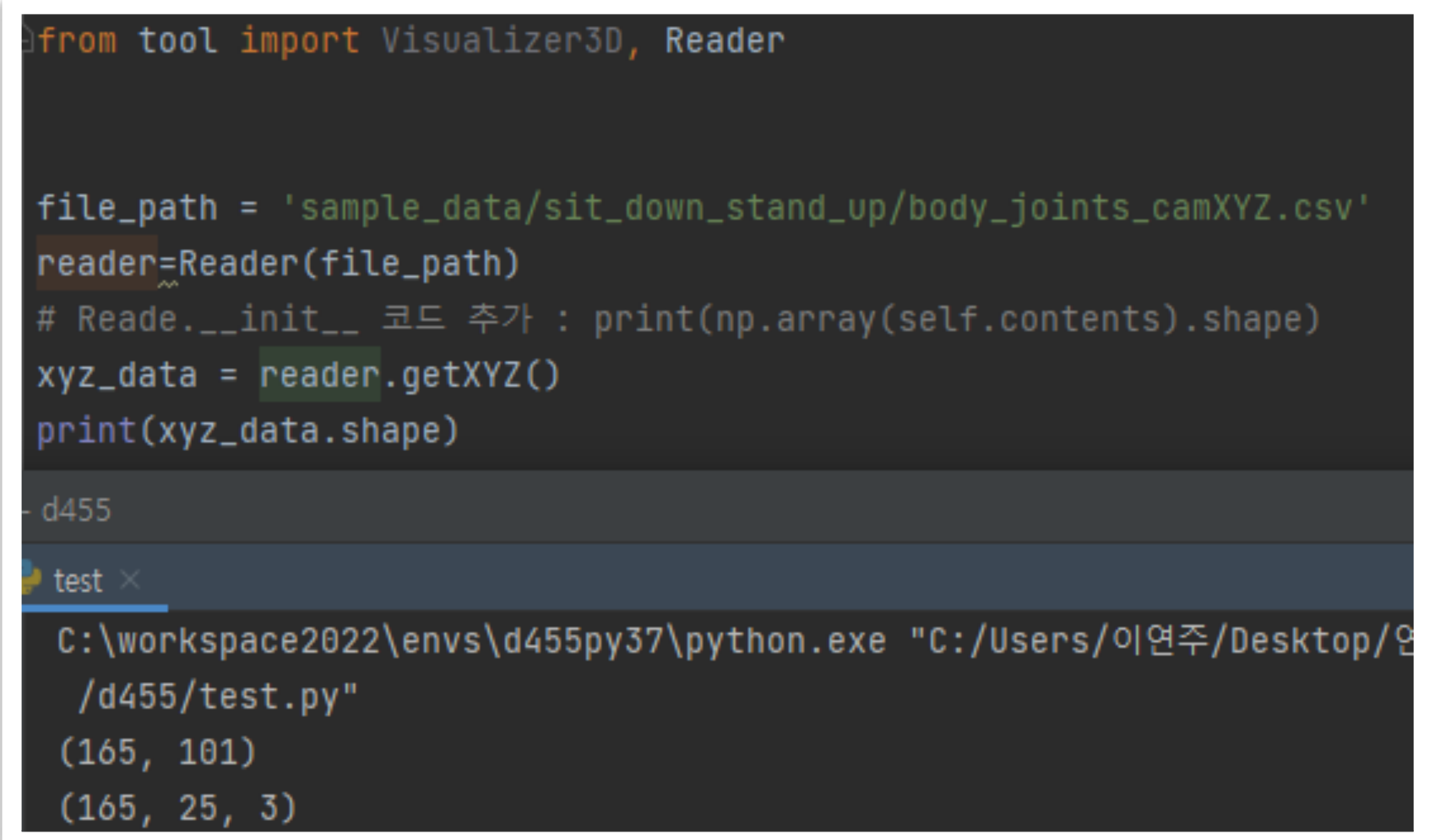
결과





```python
from tool import Visualizer3D, Reader


file_path = 'sample_data/sit_down_stand_up/body_joints_camXYZ.cs
reader=Reader(file_path)
xyz_data = reader.getXYZ()
xyz_scaled_data, _ =reader.getScaledXYZ()
visual = Visualizer3D(xyz_data=xyz_data, init_horizon=-65,pause_step =0.1,motion_stpe=1)
visual.visual_skeleton()
# visual.visual_motion_vector()
```

skeleton3DView.py



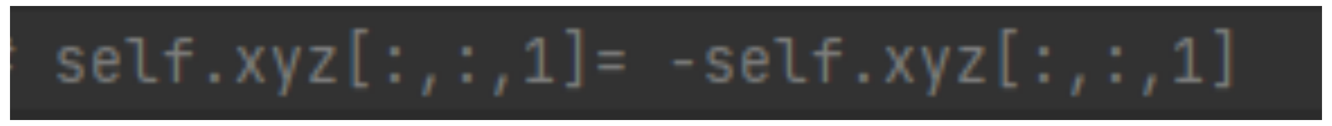
```python
from tool import Visualizer3D, Reader


file_path = 'sample_data/sit_down_stand_up/body_joints_camXYZ.csv'
reader=Reader(file_path)
# Reade.__init__ 코드 추가 : print(np.array(self.contents).shape)
xyz_data = reader.getXYZ()
print(xyz_data.shape)
```

```
C:\workspace2022\envs\d455py37\python.exe "C:/Users/이연주/Desktop/
 /d455/test.py"
(165, 101)
(165, 25, 3)
```

#5 > #6: 2차원 > 3차원 데이터

Visual3D.py

1) y값 * -1 (*값이 변함)



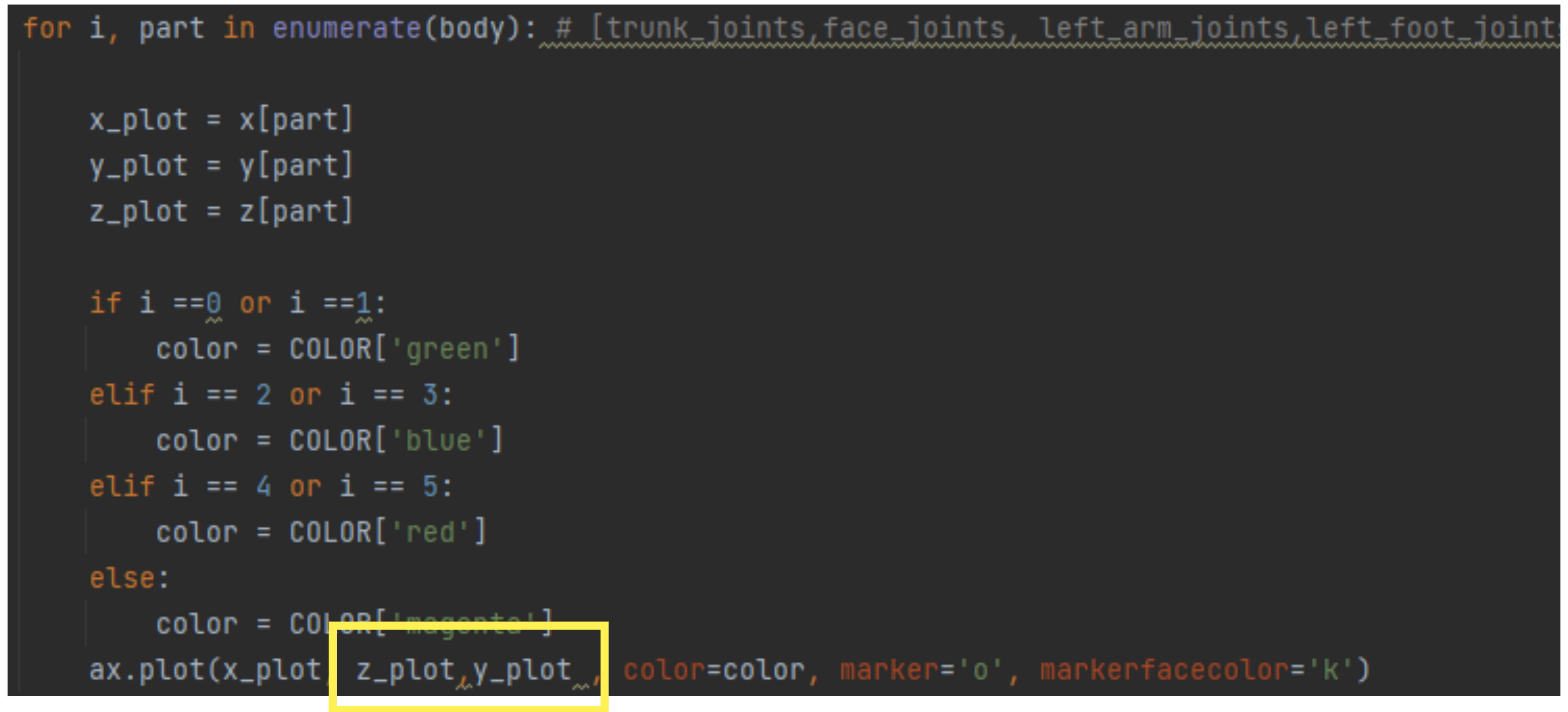
```
self.xyz[:,:,1]= -self.xyz[:,:,1]
```

카메라 좌표: y의 양의 방향: 아래 > 위로 바꿈

2) 시각화 부분

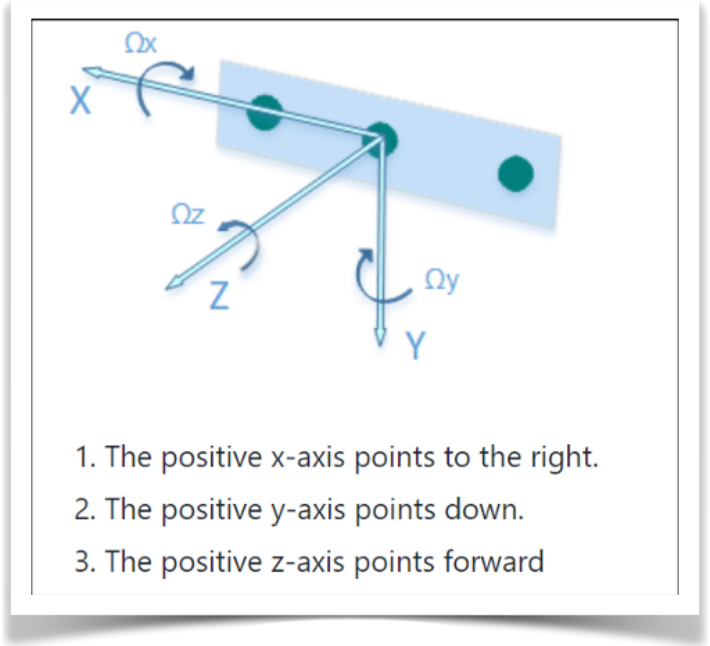
body: body-relation을 가리킴

ax.plot(x, z, y) : y <-> z축 바뀜



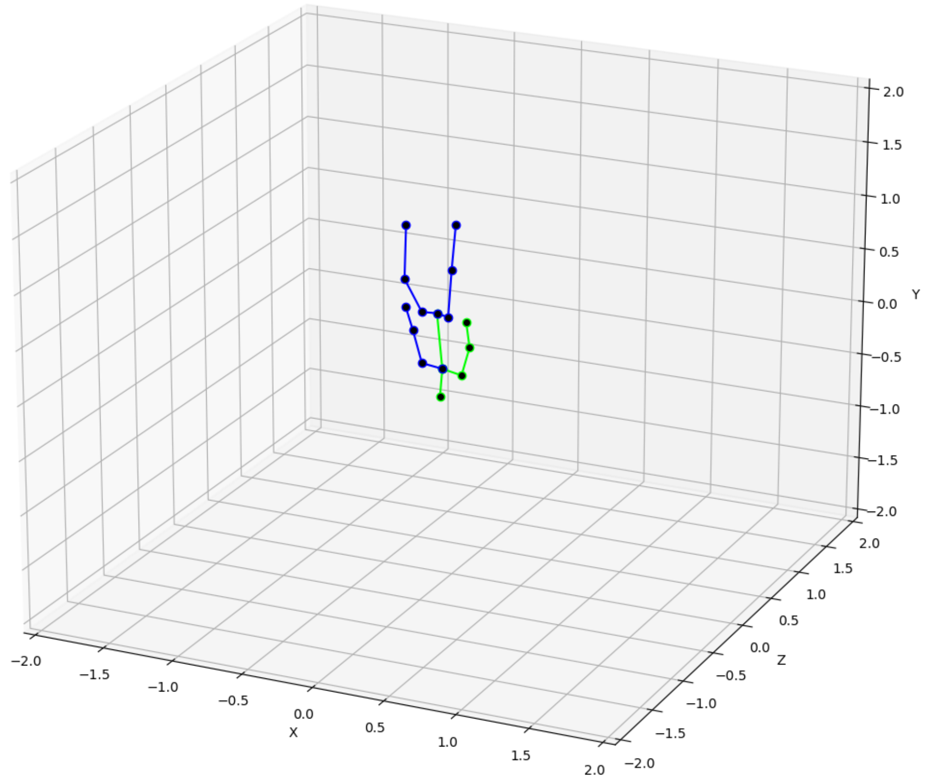
```
for i, part in enumerate(body): # [trunk_joints,face_joints, left_arm_joints,left_foot_joint

    x_plot = x[part]
    y_plot = y[part]
    z_plot = z[part]

    if i ==0 or i ==1:
        color = COLOR['green']
    elif i == 2 or i == 3:
        color = COLOR['blue']
    elif i == 4 or i == 5:
        color = COLOR['red']
    else:
        color = COLOR['magenta']
    ax.plot(x_plot, z_plot,y_plot , color=color, marker='o', markerfacecolor='k')
```

#2) 코드

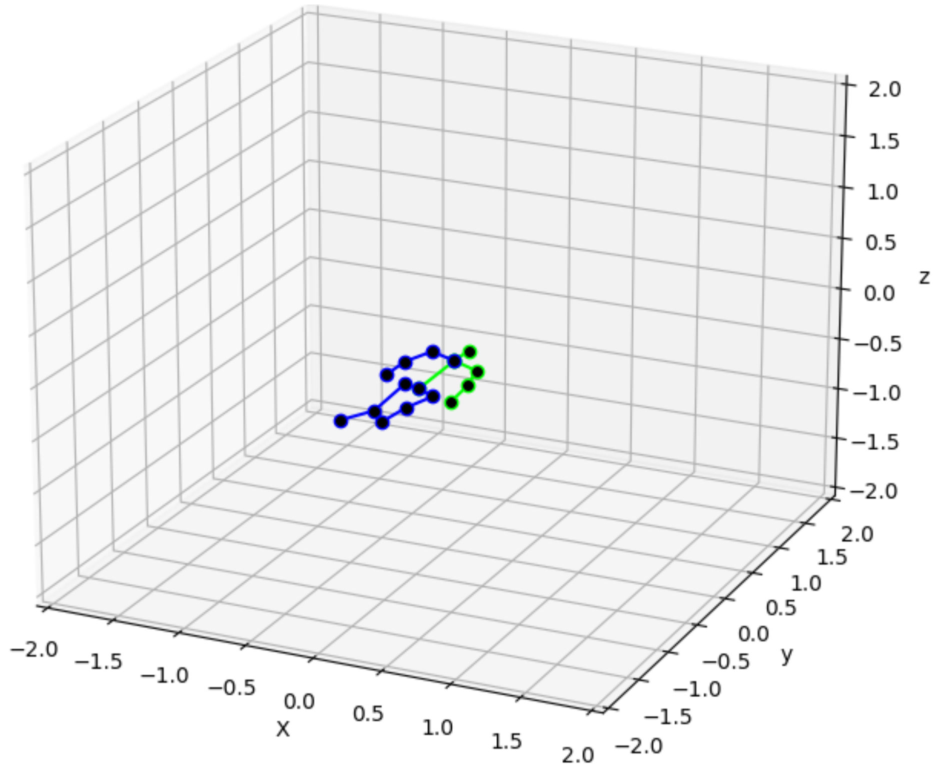


1. The positive x-axis points to the right.
2. The positive y-axis points down.
3. The positive z-axis points forward

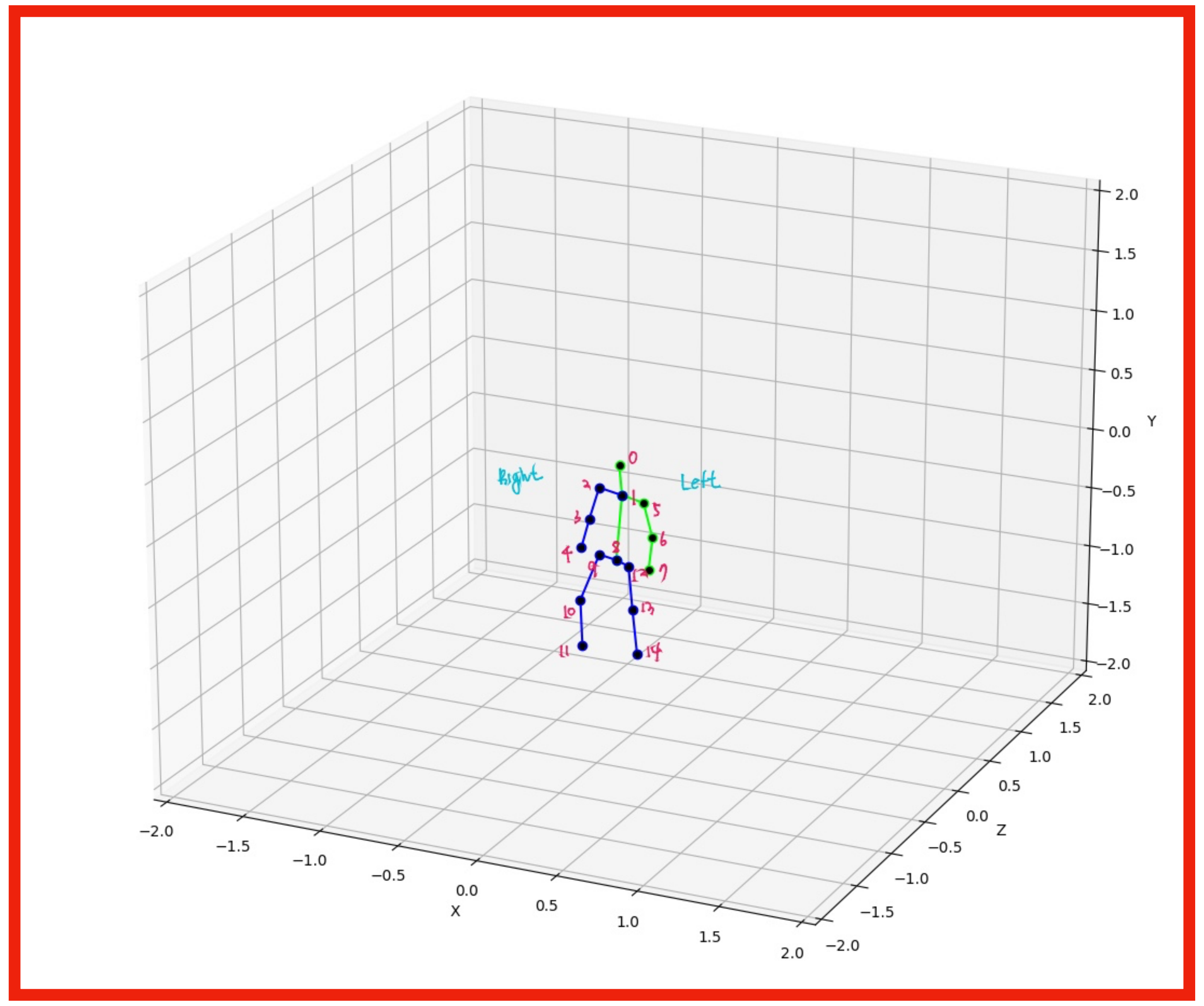
카메라 좌표(d455)



#1) y value * -1 안 한 경우



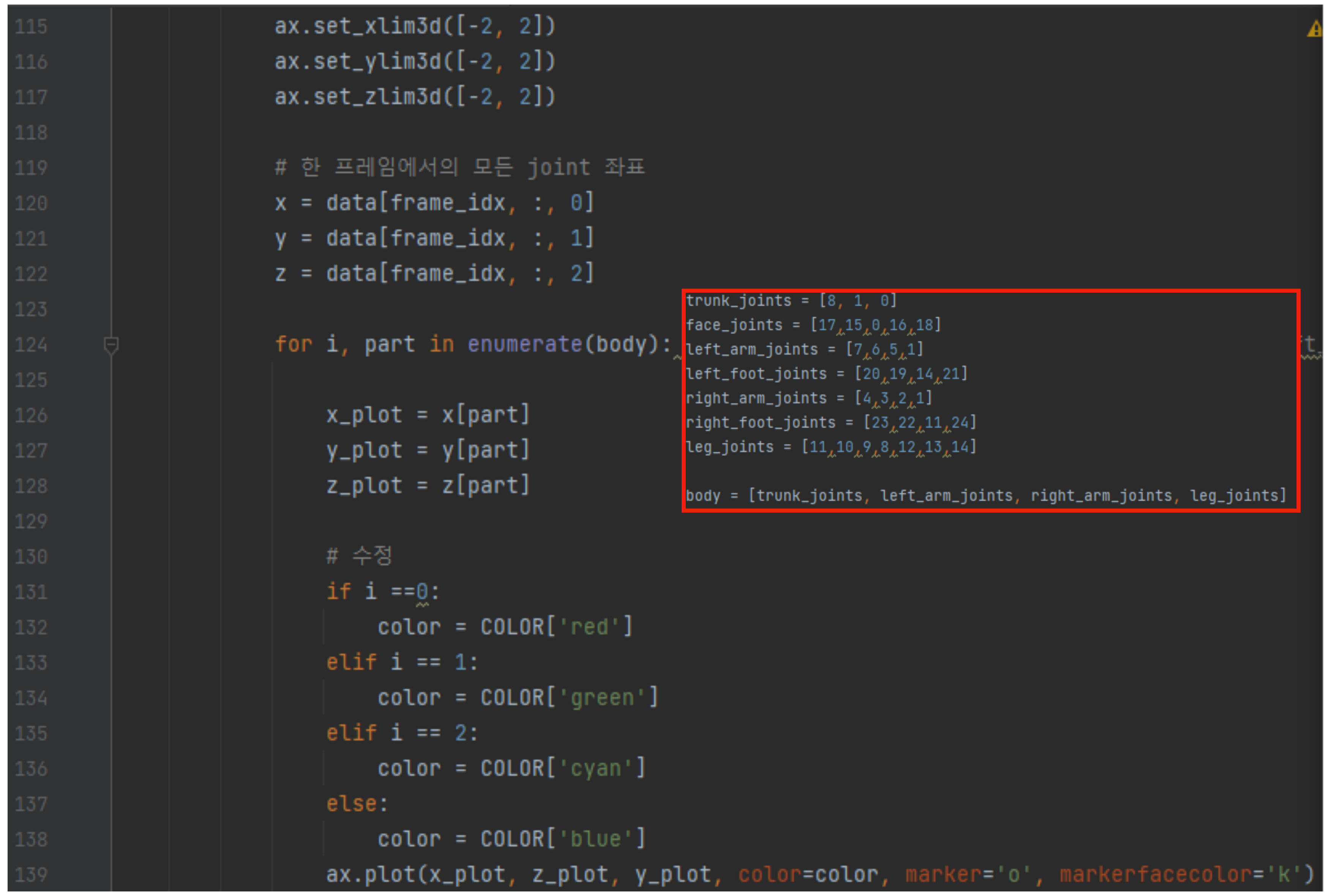
#2) y, z 안 바꾼 경우



결과

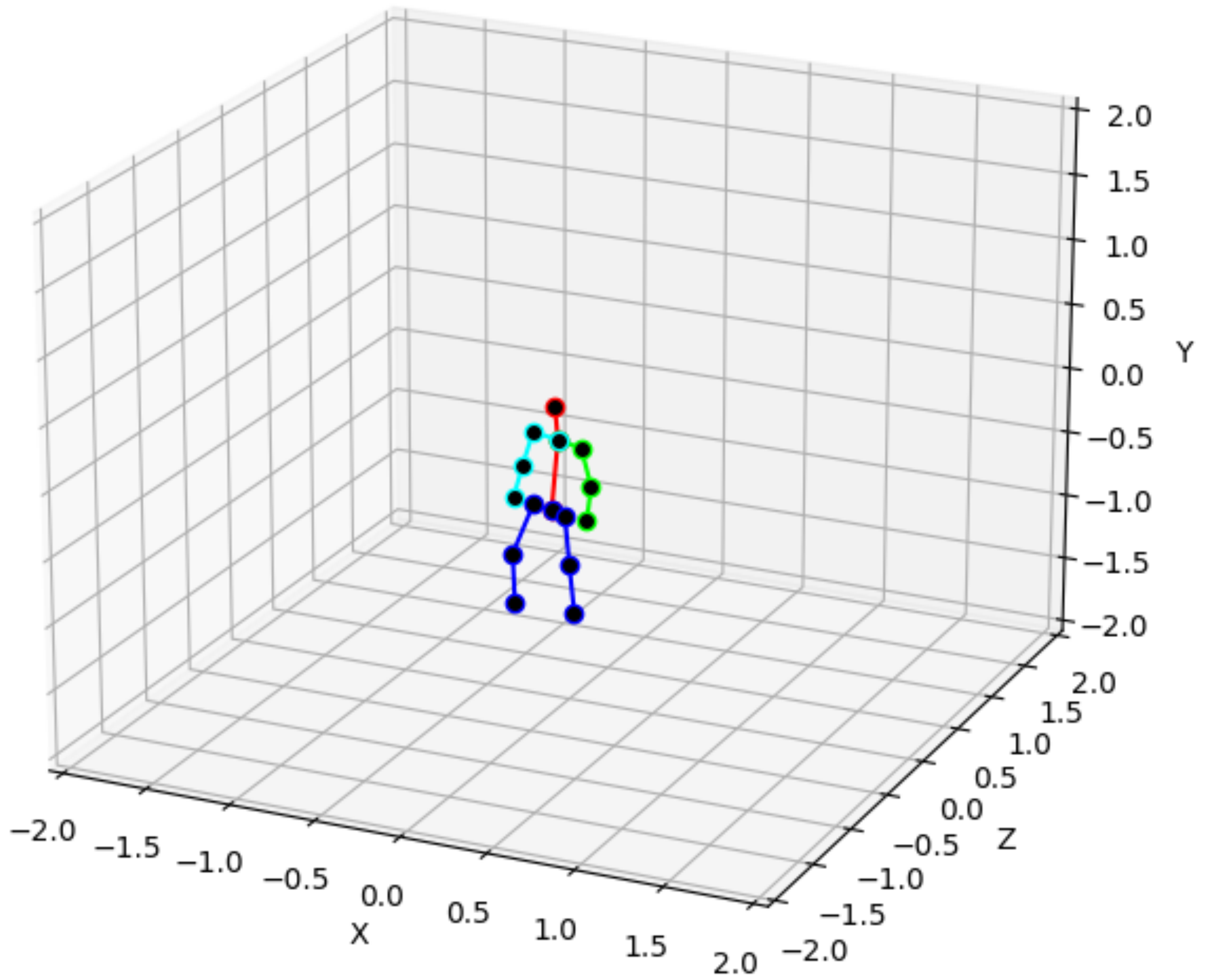
Visual3D.py

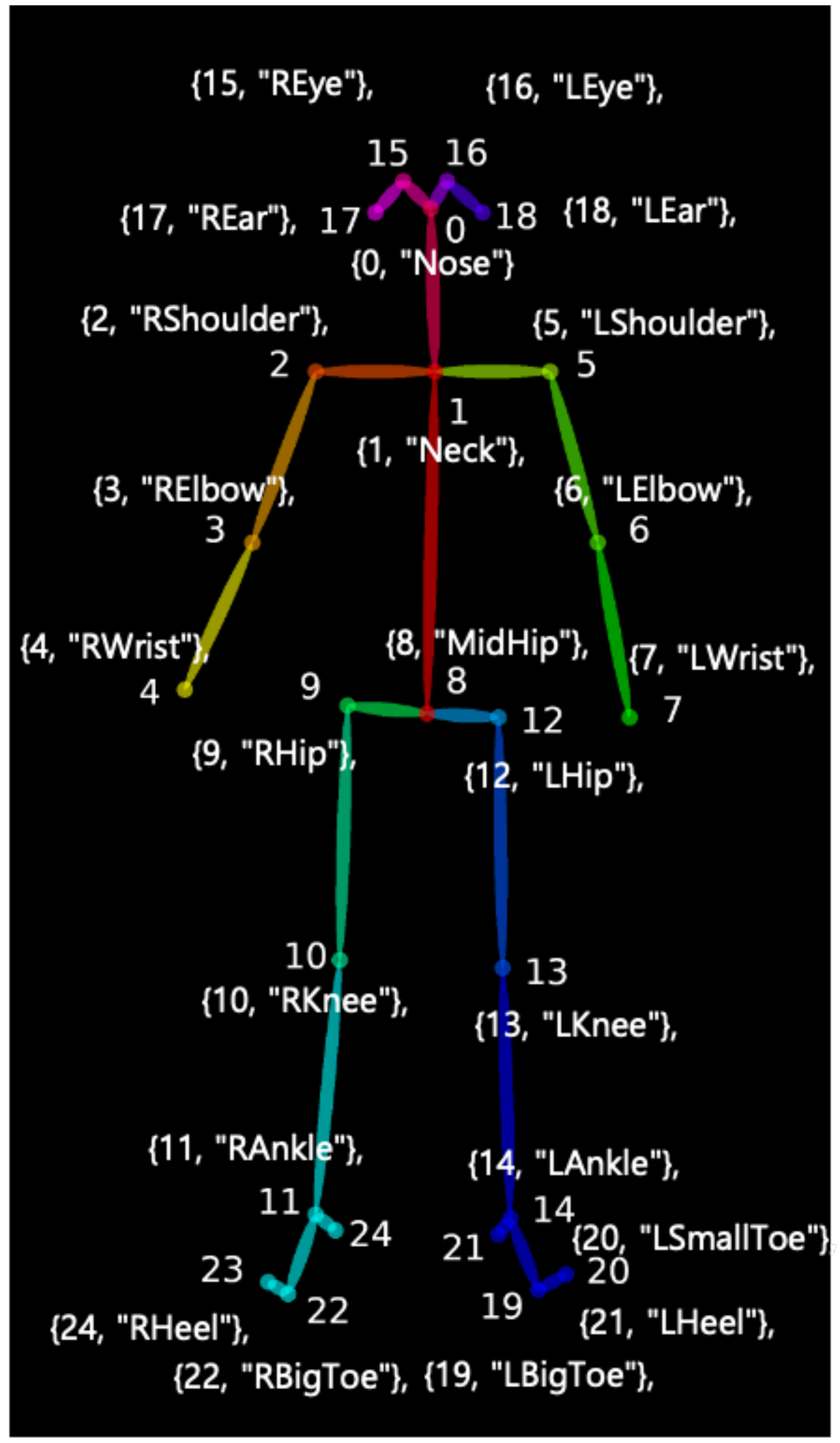
3) (추가 구현) body별로 색깔 다르게



```
ax.set_xlim3d([-2, 2])
ax.set_ylim3d([-2, 2])
ax.set_zlim3d([-2, 2])

# 한 프레임에서의 모든 joint 좌표
x = data[frame_idx, :, 0]
y = data[frame_idx, :, 1]
z = data[frame_idx, :, 2]

for i, part in enumerate(body):

    x_plot = x[part]
    y_plot = y[part]
    z_plot = z[part]

    # 수정
    if i ==0:
        color = COLOR['red']
    elif i == 1:
        color = COLOR['green']
    elif i == 2:
        color = COLOR['cyan']
    else:
        color = COLOR['blue']
    ax.plot(x_plot, z_plot, y_plot, color=color, marker='o', markerfacecolor='k')
```

```
trunk_joints = [8, 1, 0]
face_joints = [17,15,0,16,18]
left_arm_joints = [7,6,5,1]
left_foot_joints = [20,19,14,21]
right_arm_joints = [4,3,2,1]
right_foot_joints = [23,22,11,24]
leg_joints = [11,10,9,8,12,13,14]

body = [trunk_joints, left_arm_joints, right_arm_joints, leg_joints]
```





#3) 색을 바꿔봄 (Visualizer3D.py 내의 visual_skeleton 수정)
앞선 코드에서는 green, blue 색깔의 스켈레톤 시각화, 수정 코드에서는 4가지 색깔의 스켈레톤 시각화

openpose_pyrealsense.py

1) stream format (color, depth)

(1) enable_stream() => configure.enable_stream(매개변수)

- 함수: 매개변수를 이용하여 장치 스트림을 활성화

  (애플리케이션이 특정 구성으로 스트림을 요청할 수 있음)

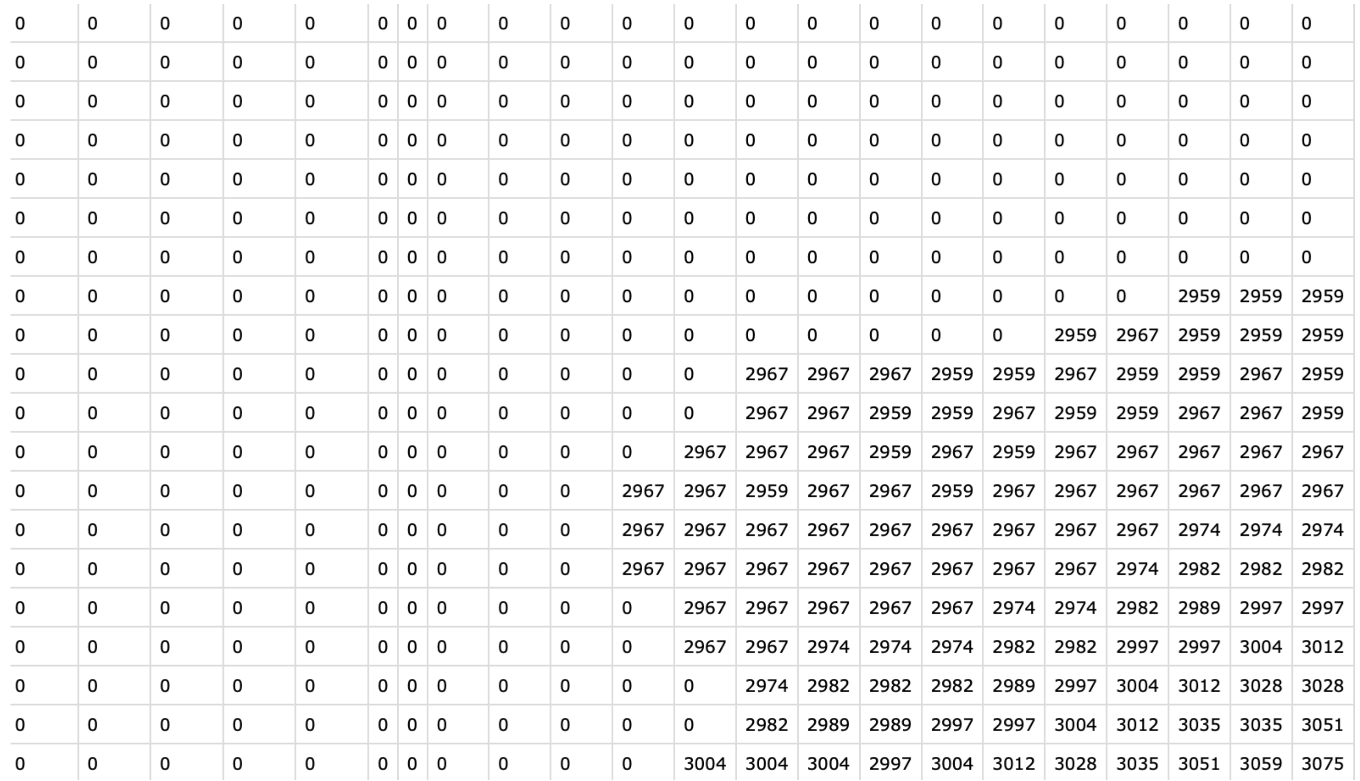
- 매개변수: stream_type: 스트림 유형, resolution: 해상도(w, h),

  format: 스트림 포맷, fps: 프레임속도

(2) 인수 설명

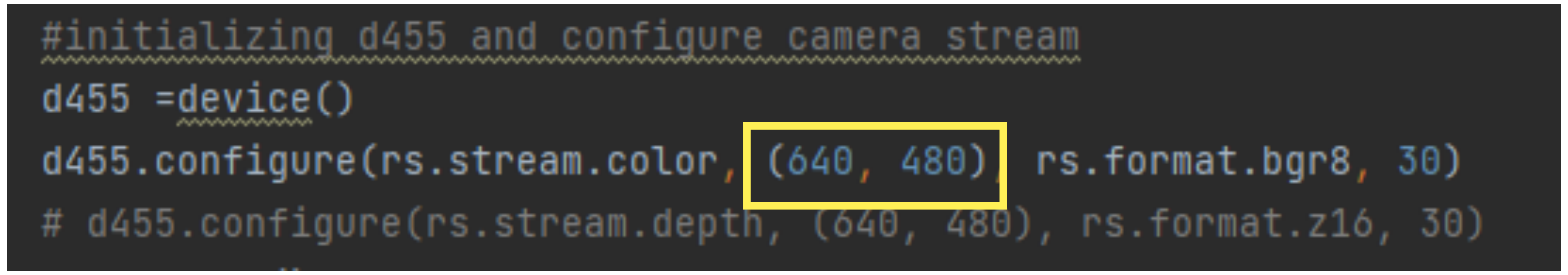
- bgr8: 8비트 B,G,R 채널로 구성된 픽셀(채널당 256개의 색의 사용), suitable for OpenCV!
- z16: 16비트 depth 값(depth 값 = depth scale * pixel value), 미터 단위

z16 - depth_array



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2959	2959	2959
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2959	2967	2959	2959	2959
0	0	0	0	0	0	0	0	0	0	0	0	2967	2967	2967	2959	2959	2967	2959	2959	2967	2959
0	0	0	0	0	0	0	0	0	0	0	0	2967	2967	2959	2959	2967	2959	2959	2967	2967	2959
0	0	0	0	0	0	0	0	0	0	0	2967	2967	2967	2959	2967	2959	2967	2967	2967	2967	2967
0	0	0	0	0	0	0	0	0	0	2967	2967	2959	2967	2967	2959	2967	2967	2967	2967	2967	2967
0	0	0	0	0	0	0	0	0	0	2967	2967	2967	2967	2967	2967	2967	2967	2967	2974	2974	2974
0	0	0	0	0	0	0	0	0	0	2967	2967	2967	2967	2967	2967	2967	2967	2974	2982	2982	2982
0	0	0	0	0	0	0	0	0	0	0	2967	2967	2967	2967	2967	2974	2974	2982	2989	2997	2997
0	0	0	0	0	0	0	0	0	0	0	2967	2967	2974	2974	2974	2982	2982	2997	2997	3004	3012
0	0	0	0	0	0	0	0	0	0	0	0	2974	2982	2982	2982	2989	2997	3004	3012	3028	3028
0	0	0	0	0	0	0	0	0	0	0	0	2982	2989	2989	2997	2997	3004	3012	3035	3035	3051
0	0	0	0	0	0	0	0	0	0	0	3004	3004	3004	2997	3004	3012	3028	3035	3051	3059	3075

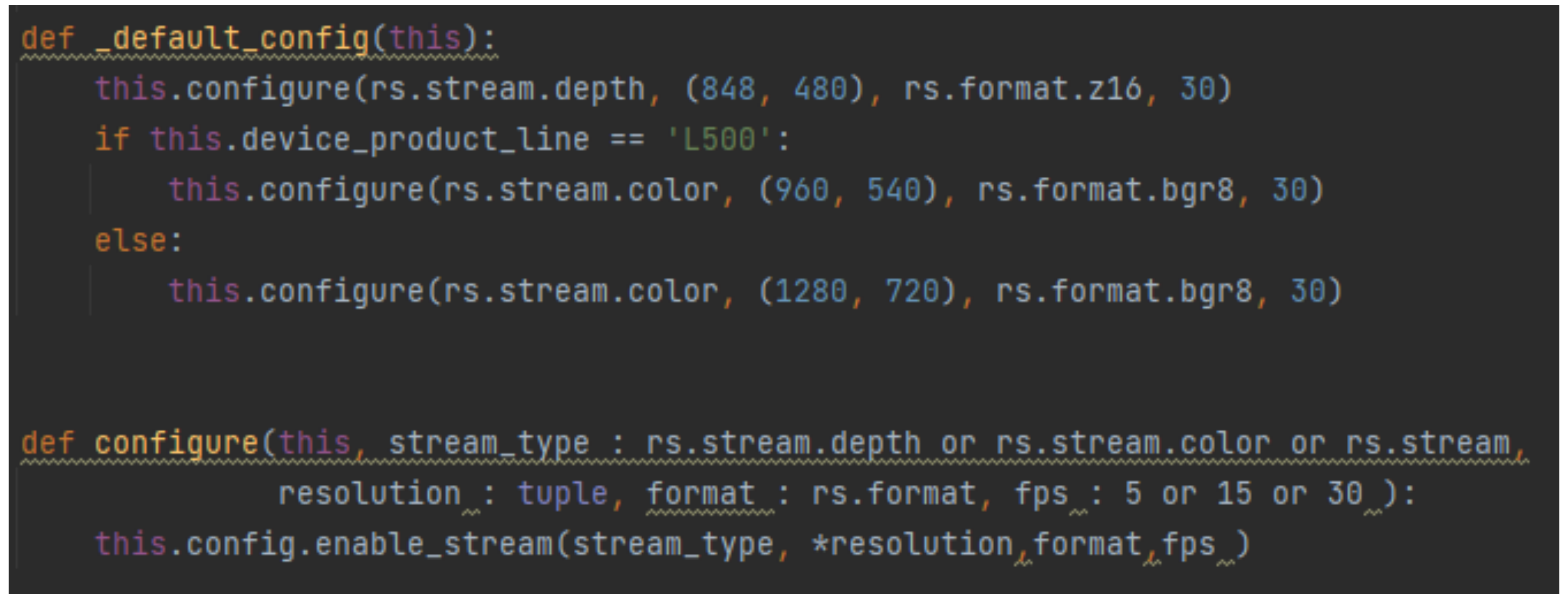
sample_data/sit_down_stand_up/depth_array/0.csv
16bit = 값이 256 아님. 그 이상
2445 = 2445mm



```
#initializing d455 and configure camera stream
d455 =device()
d455.configure(rs.stream.color, (640, 480), rs.format.bgr8, 30)
# d455.configure(rs.stream.depth, (640, 480), rs.format.z16, 30)
```

openpose_pyrealsense.py

stream에서 (y, x) > (x, y) = (480, 640)



```
def _default_config(this):
    this.configure(rs.stream.depth, (848, 480), rs.format.z16, 30)
    if this.device_product_line == 'L500':
        this.configure(rs.stream.color, (960, 540), rs.format.bgr8, 30)
    else:
        this.configure(rs.stream.color, (1280, 720), rs.format.bgr8, 30)


def configure(this, stream_type : rs.stream.depth or rs.stream.color or rs.stream,
              resolution : tuple, format : rs.format, fps : 5 or 15 or 30 ):
    this.config.enable_stream(stream_type, *resolution,format,fps )
```

realsense.py - 다른 예시

openpose_pyrealsense.py

2) 데이터 준비 및 전처리

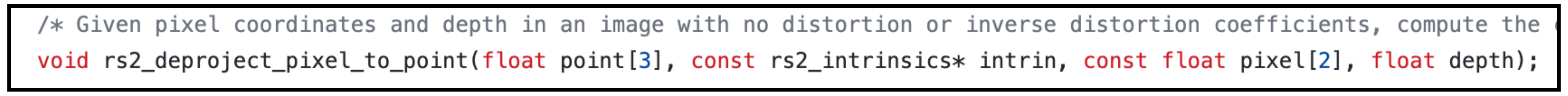
1) depth_scale for cal depth 값 (depth = scale * depth_pixel)

2) colorizer(): color frame 좌표로 깊이 계산

(color, depth 이미지의 정렬이 다르기 때문에 두 이미지를 수평으로 연결함
- 이 과정에서 depth 값 8bit로 변환됨.)

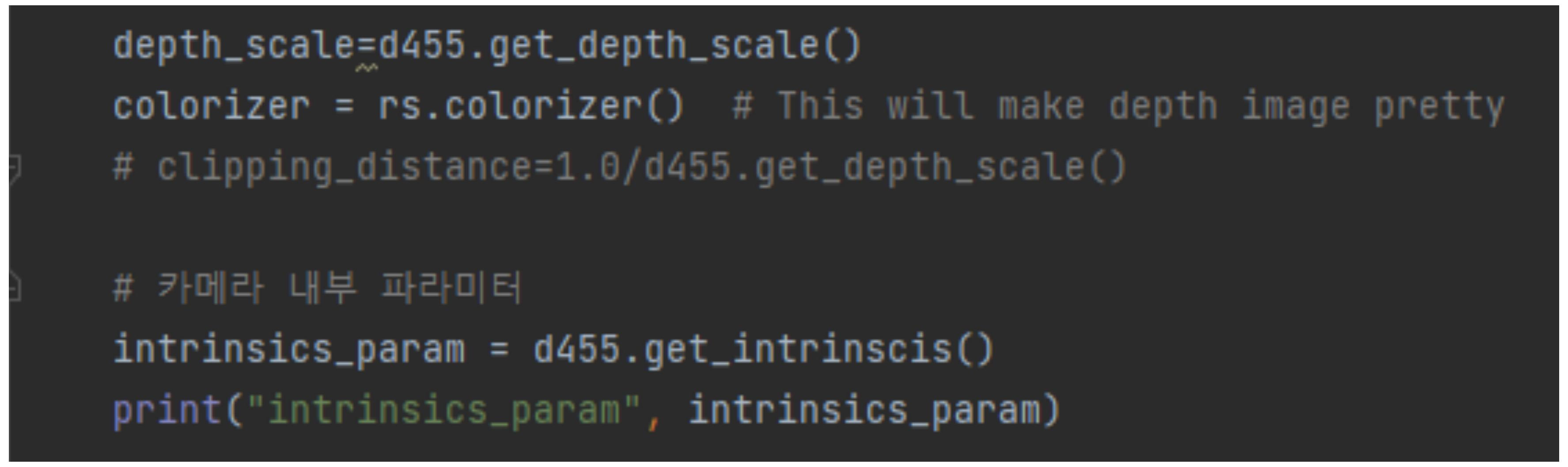
3) get_intrinsic(): 카메라 내부 파라미터 for deprojection

(정리) For Deprojection



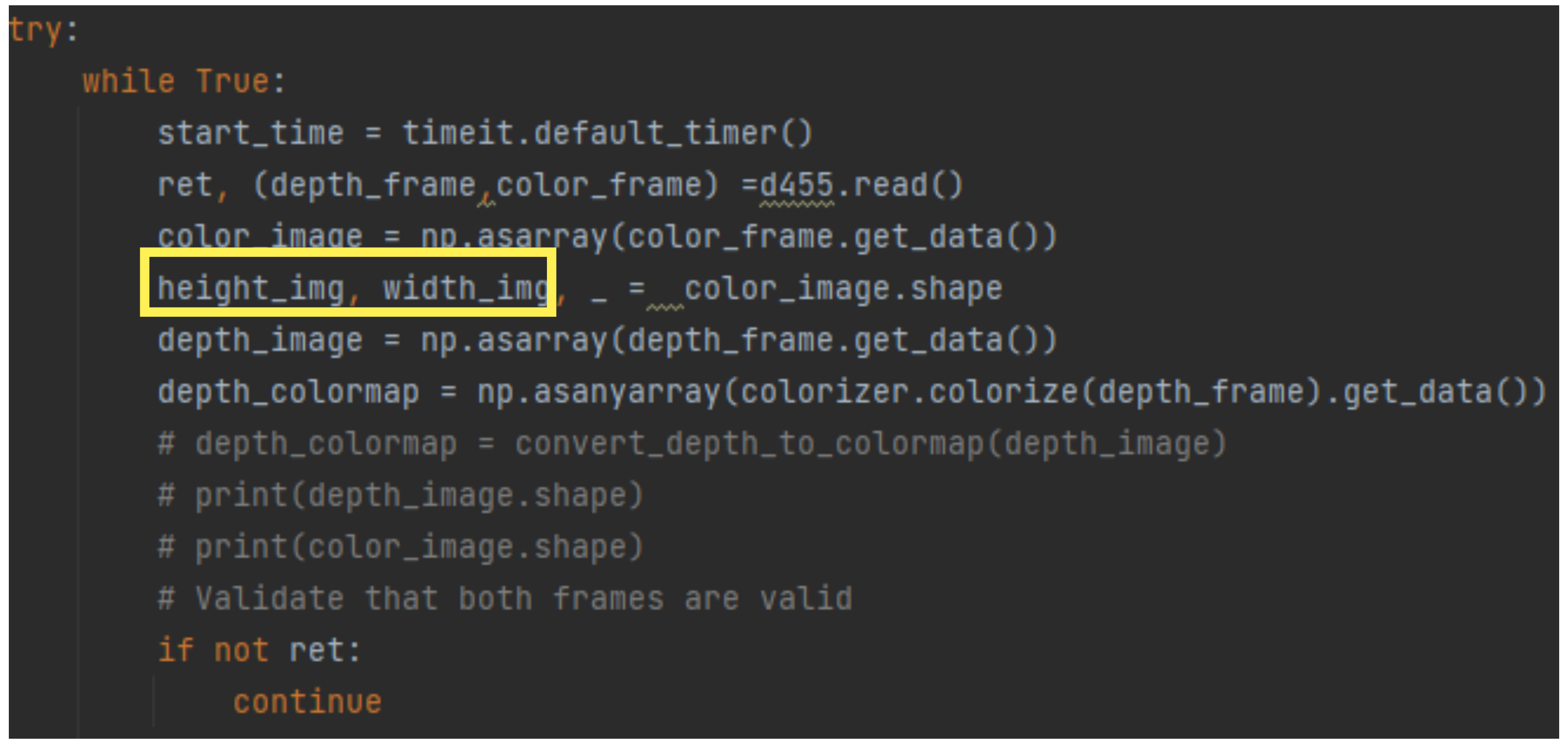
```
/* Given pixel coordinates and depth in an image with no distortion or inverse distortion coefficients, compute the
void rs2_deproject_pixel_to_point(float point[3], const rs2_intrinsics* intrin, const float pixel[2], float depth);
```

- intrin: 카메라 내부 파라미터
- pixel[2]: color stream
- depth: depth stream & depth_scale



```
depth_scale=d455.get_depth_scale()
colorizer = rs.colorizer()  # This will make depth image pretty
# clipping_distance=1.0/d455.get_depth_scale()

# 카메라 내부 파라미터
intrinsics_param = d455.get_intrinscis()
print("intrinsics_param", intrinsics_param)
```

필요한 데이터 준비(1)



```
try:
    while True:
        start_time = timeit.default_timer()
        ret, (depth_frame,color_frame) =d455.read()
        color_image = np.asarray(color_frame.get_data())
        height_img, width_img, _ = color_image.shape
        depth_image = np.asarray(depth_frame.get_data())
        depth_colormap = np.asanyarray(colorizer.colorize(depth_frame).get_data())
        # depth_colormap = convert_depth_to_colormap(depth_image)
        # print(depth_image.shape)
        # print(color_image.shape)
        # Validate that both frames are valid
        if not ret:
            continue
```

필요한 데이터 준비(2)
*height_img, width_img 순서 주의

openpose_pyrealsense.py

*** Person Detected~PersonDetecd사이의 배열은 한 사람의 joint 데이터(x,y,s)임**

2) 주요 Action

(1) datum.poseKeypoints

joints의 x, y, s값 구하기

- 이미지의 각 사람에 대한 신체 포즈(x,y,점수) 위치
- keypoints.shape = [bodies, joints, (x,y,z)]
- keypoints.size = 1(사람) * 25 * 3

(3) np.clip(arr, min, max)

(4) depth = depth[y][x] * depth_scale

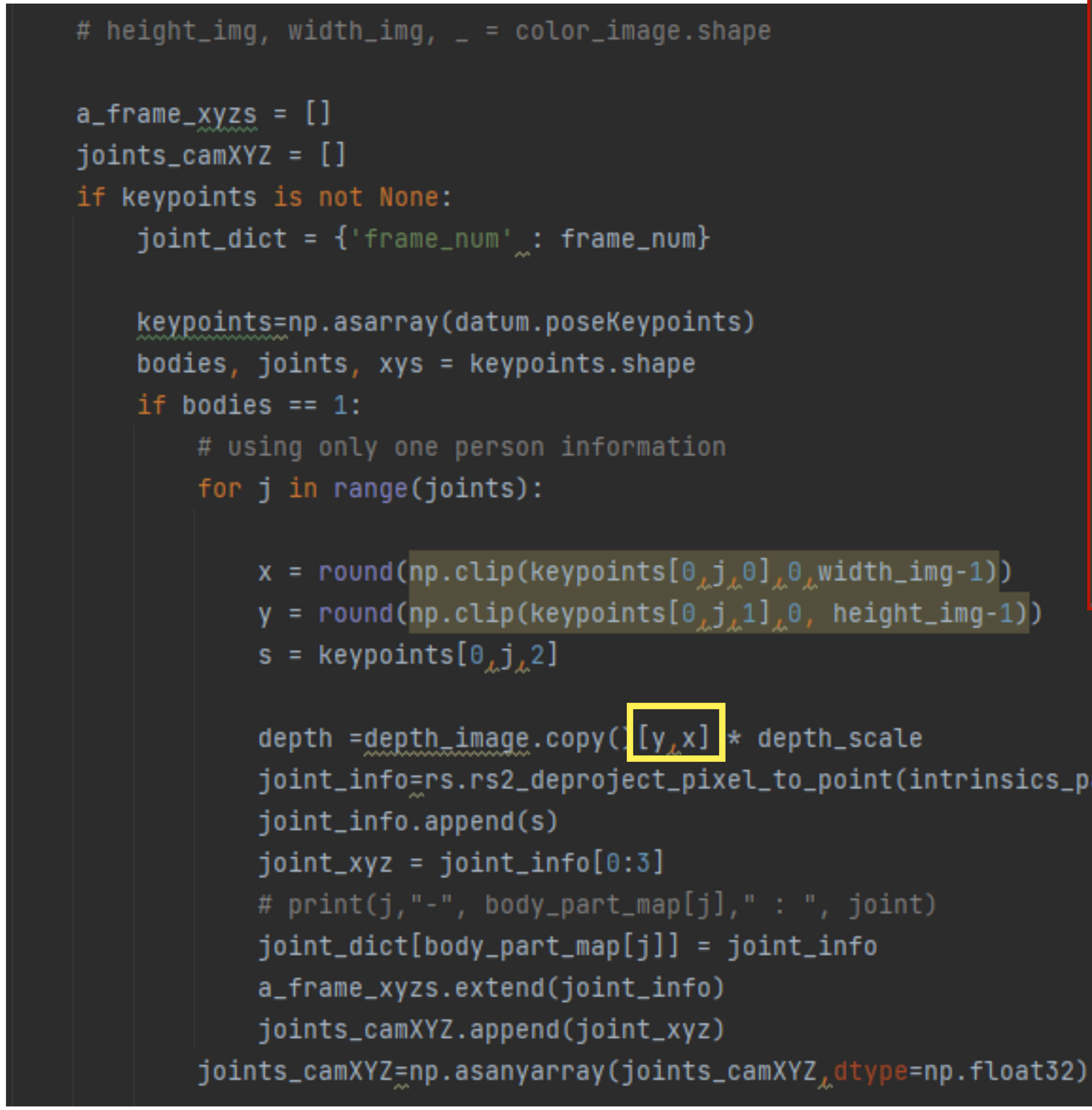
<x,y, depth>

(5) rs2_deproject_pixel_to_point()

(6) joint_xyz 값 저장

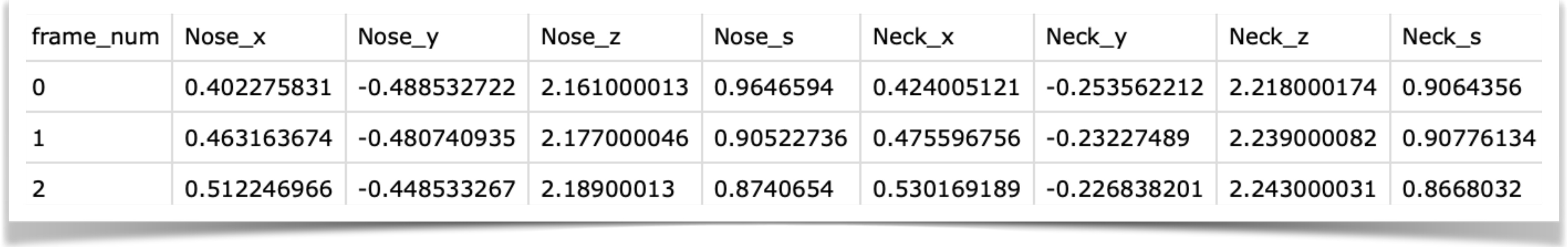
```
   Person Detected
                 x    y  confidences
Nose           352  162     0.877677
Neck           354  186     0.860390
RShoulder      330  186     0.785359
RElbow         300  222     0.852995
RWrist         339  227     0.827675
LShoulder      380  186     0.790244
LElbow         411  224     0.868296
LWrist         380  230     0.827934
MidHip         365  253     0.552375
RHip           346  258     0.581863
RKnee          381  252     0.601171
RAnkle         442  317     0.621208
LHip           383  250     0.542253
LKnee          382  273     0.543372
LAnkle           0    0     0.000000
REye           346  156     0.936202
LEye           359  156     0.894469
REar           338  159     0.700735
LEar           368  159     0.772414
LBigToe          0    0     0.000000
LSmallToe        0    0     0.000000
LHeel            0    0     0.000000
RBigToe        450  333     0.112443
RSmallToe        0    0     0.000000
RHeel          450  319     0.366867

   Person Detected
                 x    y  confidences
Nose           230  112     0.952630
Neck           217  141     0.843662
RShoulder      191  140     0.758305
RElbow         202  192     0.849608
RWrist         214  148     0.797286
LShoulder      245  141     0.860531
```

keypoints 예시



```
# height_img, width_img, _ = color_image.shape

a_frame_xyzs = []
joints_camXYZ = []
if keypoints is not None:
    joint_dict = {'frame_num' : frame_num}

    keypoints=np.asarray(datum.poseKeypoints)
    bodies, joints, xys = keypoints.shape
    if bodies == 1:
        # using only one person information
        for j in range(joints):

            x = round(np.clip(keypoints[0,j,0],0,width_img-1))
            y = round(np.clip(keypoints[0,j,1],0, height_img-1))
            s = keypoints[0,j,2]

            depth =depth_image.copy()[y,x] * depth_scale
            joint_info=rs.rs2_deproject_pixel_to_point(intrinsics_param, (x,y),depth)
            joint_info.append(s)
            joint_xyz = joint_info[0:3]
            # print(j,"-", body_part_map[j]," : ", joint)
            joint_dict[body_part_map[j]] = joint_info
            a_frame_xyzs.extend(joint_info)
            joints_camXYZ.append(joint_xyz)
        joints_camXYZ=np.asanyarray(joints_camXYZ,dtype=np.float32)
```

주요 코드



frame_num	Nose_x	Nose_y	Nose_z	Nose_s	Neck_x	Neck_y	Neck_z	Neck_s
0	0.402275831	-0.488532722	2.161000013	0.9646594	0.424005121	-0.253562212	2.218000174	0.9064356
1	0.463163674	-0.480740935	2.177000046	0.90522736	0.475596756	-0.23227489	2.239000082	0.90776134
2	0.512246966	-0.448533267	2.18900013	0.8740654	0.530169189	-0.226838201	2.243000031	0.8668032

코드에서의 a_frame_xyzs 저장한 결과(.csv)