* **Chembl\_pipeline 설치 및 사용법**

**1. Chembl\_pipeline 설치**

#git에서 chembl\_pipeline 설치 및 anaconda 설치

#base에서 진행

conda activate

git clone https://github.com/chembl/ChEMBL\_Structure\_Pipeline.git

pip install ./ChEMBL\_Structure\_Pipeline

conda install -c conda-forge chembl\_structure\_pipeline

#python script 작성

vi standard.py

import csv

import os

import sys

import glob

from rdkit import Chem

from chembl\_structure\_pipeline import standardizer, checker

# CSV 파일 처리 함수 정의

def process\_csv(input\_csv, output\_csv):

    with open(input\_csv, 'r') as infile, open(output\_csv, 'w', newline='') as outfile:

        # 콤마 구분자 사용

        reader = csv.reader(infile, delimiter=',')

        writer = csv.writer(outfile)

        # 첫 번째 row에 헤더 추가

        writer.writerow(['Name', 'Input SMILES', 'Canonical SMILES'])

        # 각 row 처리

        for row in reader:

            # 빈 줄이나 잘못된 형식의 행 건너뛰기

            if len(row) < 2:

                continue

            name, smiles = row[0], row[1]

            try:

                #print(f"Processing: {name} - {smiles}")

                # SMILES를 Mol 객체로 변환 (sanitize=False로 RDKit 검증 생략)

                mol = Chem.MolFromSmiles(smiles, sanitize=False)

                if mol is None:

                    raise ValueError(f"SMILES 변환 실패: {smiles}")

                #print(f"SMILES successfully converted to Mol object for {name}")

                # Mol 객체를 MOL 블록으로 변환

                mol\_block = Chem.MolToMolBlock(mol)

                #print(f"Mol object successfully converted to MolBlock for {name}")

                # 표준화된 MOL 블록 생성

                std\_mol\_block = standardizer.standardize\_molblock(mol\_block)

                #print(f"Standardization successful for {name}")

                # 부모 화합물 추출

                parent\_mol\_block, \_ = standardizer.get\_parent\_molblock(std\_mol\_block)

                #print(f"Parent compound extracted successfully for {name}")

                # 체크 과정 (생략 가능)

                issues = checker.check\_molblock(parent\_mol\_block)

                if issues:

                    print(f"Issues found during checking for {name}: {issues}")

                # 부모 화합물 MOL 블록을 RDKit Mol 객체로 변환

                parent\_mol = Chem.MolFromMolBlock(parent\_mol\_block, sanitize=False)

                if parent\_mol is None:

                    raise ValueError(f"MOL 블록 변환 실패: {parent\_mol\_block}")

                #print(f"Parent MolBlock successfully converted to Mol object for {name}")

                # 표준화된 canonical SMILES 생성

                canonical\_smiles = Chem.MolToSmiles(parent\_mol, canonical=True)

                print(f"Canonical SMILES generated successfully for {name}")

            except Exception as e:

                # 변환 실패 시 canonical\_smiles에 "X" 표기

                canonical\_smiles = "X"

                print(f"Error occurred while processing {name}: {e}")

            # CSV에 결과 기록

            writer.writerow([name, smiles, canonical\_smiles])

# 디렉토리의 모든 CSV 파일을 처리하는 함수 정의

def process\_directory(directory):

    # 디렉토리 내 모든 CSV 파일 찾기

    csv\_files = glob.glob(os.path.join(directory, '\*.csv'))

    if not csv\_files:

        print(f"No CSV files found in directory: {directory}")

        return

    for csv\_file in csv\_files:

        # 각 CSV 파일에 대해 "\_out.csv" 파일명 생성

        base\_name = os.path.basename(csv\_file)

        file\_name, \_ = os.path.splitext(base\_name)

        output\_file = os.path.join(directory, f"{file\_name}\_out.csv")

        print(f"Processing {csv\_file} -> {output\_file}")

        process\_csv(csv\_file, output\_file)

# 스크립트 실행

if \_\_name\_\_ == "\_\_main\_\_":

    # 명령줄 인자 처리

    if len(sys.argv) == 1:

        # 명령줄 인자가 없을 경우 현재 디렉토리에서 실행

        current\_dir = os.getcwd()

        print(f"현재 디렉토리: {current\_dir}")

        process\_directory(current\_dir)

    elif len(sys.argv) == 2:

        # 특정 디렉토리가 주어졌을 경우 해당 디렉토리에서 실행

        target\_dir = sys.argv[1]

        # 디렉토리가 실제로 존재하는지 확인

        if os.path.isdir(target\_dir):

            print(f"지정된 디렉토리: {target\_dir}")

            process\_directory(target\_dir)

        else:

            print(f"오류: 지정된 디렉토리가 존재하지 않습니다 - {target\_dir}")

    else:

        print("사용법: python standard.py [directory\_path (선택사항)]")

**2. Chembl\_pipeline 실행**

#standard.py 가 있는 directory 에서 진행

#input file 저장

vi conversion\_test.csv

22707,O=Cl(=O)(=O)F

22999,FBr(F)F

23009,FBr(F)(F)(F)F

23039,FCl(F)F

26183,F[Si-2](F)(F)(F)(F)F.[Zn+2]

26184,F[Si-2](F)(F)(F)(F)F

26193,F[Si-2](F)(F)(F)(F)F.[Na+].[Na+]

26210,[NH4+].[NH4+].F[Si-2](F)(F)(F)(F)F

26352,F[Si-2](F)(F)(F)(F)F.[Ba+2]

33159,F[Ge-2](F)(F)(F)(F)F.[Mg+2]

:wq

#실행 명령어

python standard.py