♦

DansDiffraction_cif_to_xrd대량실행코드

소스코드

```
import Dans_Diffraction as dif
import os
from tqdm import tqdm
import argparse
import sys
import pandas as pd

def main():
    parser = argparse.ArgumentParser(description="argparse
r")
    parser.add_argument("--start_index", type=int)
    parser.add_argument("--last_index", type=int)
    args = parser.parse_args()
    start_index = args.start_index
    last_index = args.last_index

    output_log_path = "/home/dydtkddhkdwk/XRDdatas/outputlo
g"
    save_path = "/home/dydtkddhkdwk/XRDdatas/coremof_xrd"
    cifs_path = "/home/dydtkddhkdwk/PYRASPA/databases/corem
of_database/2019-11-01-ASR-public_12020/structure_10143"

    cifs = [x for x in os.listdir(cifs_path) if x.endswith
(".cif")][start_index:last_index]  # 수정: 슬라이싱 구문에 콜론
(:) 추가

    dic = {}
    dic["coremof"] = []
```

```
    dic["twotheta"] = []
    dic["intensive"] = []

    # 로그 파일 생성 또는 이어 쓰기
    log_file_path = os.path.join(output_log_path, "[total]c
oremof_henry_include_xrd_%s_%s.output" % (start_index, last
_index))

    for coremof in tqdm(cifs):
        coremof_path = os.path.join(cifs_path, coremof)

        # 로그 파일에 현재 처리 중인 coremof 정보를 기록
        with open(log_file_path, 'a') as log_file:
            log_file.write(f"Processing: {coremof}\n")

        try:
            xtl = dif.Crystal(coremof_path)
            sys.stdout = open(os.devnull, 'w')  # 출력 방지
            xtl.Scatter.setup_scatter(scattering_type='x-ra
y', energy_kev=8.0)
            sys.stdout = sys.__stdout__

            twotheta, iten, reflections = xtl.Scatter.powde
r(units='twotheta')
            dic["coremof"].append(coremof)
            dic["twotheta"].append(twotheta.tolist())
            dic["intensive"].append(iten.tolist())
            # dic["reflections"].append(reflections.tolist
())

        except Exception as e:
            # 에러 발생 시 로그 파일에 기록
            with open(log_file_path, 'a') as log_file:
                log_file.write(f"Error processing {coremo
f}: {str(e)}\n")

    # 결과를 CSV로 저장
    output_csv = os.path.join(save_path, "[total]coremof_he
```

```
nry_include_xrd_%s_%s.csv" % (start_index, last_index))
    pd.DataFrame(dic).to_csv(output_csv)

if __name__ == "__main__":
    main()
```

WSL이나 리눅스 환경에서 실시할 subprocess + joblib코드

```
import subprocess
from joblib import Parallel, delayed
import os
path = "/home/dydtkddhkdwk/XRDdatas/codes/"
# 분할 작업을 수행하는 함수
def run_coremof(start_index, last_index):
    # Conda 환경을 활성화하고 스크립트를 실행하는 명령을 작성
    command = f"python {path}coremof_cif_to_xrd.py --start_
index {start_index} --last_index {last_index}"

    # 명령을 출력하여 디버깅할 수 있도록 함
    print(f"Running: {command}")

    # Conda 환경에서 명령을 실행
    result = subprocess.run(["bash", "-c", command], captur
e_output=True, text=True)

    # 오류가 발생하면 에러 메시지 출력
    if result.returncode != 0:
        print(f"Error running coremof_cif_to_xrd.py for ind
ices {start_index} to {last_index}")
        print(result.stderr)
    else:
        print(f"Completed: indices {start_index} to {last_i
ndex}")

def main():
    # CIF 파일 총 개수를 확인
    cifs_path = "//home/dydtkddhkdwk/PYRASPA/databases/core
mof_database/2019-11-01-ASR-public_12020/structure_10143"
```

```
    total_cifs = len([x for x in os.listdir(cifs_path) if
x.endswith(".cif")])
    print(total_cifs)
    # 인덱스 구간을 10개씩 나눔
    index_ranges = [(i, min(i + 50, total_cifs)) for i in r
ange(0, total_cifs, 50)]
    # 병렬 처리: 10개의 CPU로 작업 분할
    Parallel(n_jobs=4)(delayed(run_coremof)(start, end) for
start, end in index_ranges)

if __name__ == "__main__":
    main()
```