IITP DABT DataCollector

설계서

문서 버전: 1.0.0
작성일: 2025-11-26

(주)스위트케이

문서 History

버전	일자	작성자	변경 내용
1.0.0	2025-11-26	(주)스위트케이	최초 작성

목차

1. 개요

1.1. 프로젝트 소개

IITP DABT DataCollector는 CSV 파일에 포함된 이미지 URL 정보를 기반으로 대량의 이미지를 자동으로 다운로드하는 데이터 수집 도구입니다.

DataCollector는 구조화된 CSV 데이터에서 이미지 링크를 추출하여, 날짜 기반 폴더 구조로 체계적으로 저장하며, 멀티스레드 기반의 병렬 다운로드를 통해 효율적인 데이터 수집을 지원합니다.

주요 특징으로는 크로스 플랫폼(Windows, Linux/Ubuntu) 지원을 위한 안전한 파일명 처리, 다양한 이미지 URL 형식 지원(일반 HTTP/HTTPS URL 및 Base64 인코딩된 Data URL), 실패한 다운로드에 대한 상세한 에러 리포트 생성 등이 있습니다.

1.2. 핵심 기능

- **CSV 기반 이미지 URL 처리**: 표준화된 CSV 형식(No, Type, Title, Img-link)에서 이미지 링크 추출
- **멀티스레드 병렬 다운로드**: 설정 가능한 스레드 수를 통한 동시 다운로드 처리로 성능 최적화
- **날짜 기반 폴더 구조**: 실행일 기준 YYYY-MM-DD 형식의 폴더에 이미지 자동 분류 저장
- **크로스 플랫폼 파일명 생성**: Windows 및 Linux/Unix 파일시스템 제약을 고려한 안전한 파일명 자동 생성
- **다양한 URL 형식 지원**:
  - 일반 HTTP/HTTPS 이미지 URL
  - Base64 인코딩된 Data URL (data:image/* 형식)
  - 여러 URL이 하나의 셀에 포함된 경우 자동 분리
- **확장자 자동 감지**: URL에 확장자가 없는 경우 HTTP 헤더의 Content-Type을 분석하여 확장자 추론
- **에러 추적 및 리포트**: 실패한 다운로드에 대한 상세 정보를 포함한 에러 CSV 파일 자동 생성
- **에러 분석 유틸리티**: 에러 리포트를 분석하여 실패 원인별 개수 집계 및 샘플 데이터 제공
- **유연한 HTTP 설정**: SSL 검증, 커스텀 헤더, HEAD 요청 사전 검증 등 다양한 네트워크 환경 대응

2. 프로젝트 전체 소스 구조 및 설명

2.1. 디렉토리 구조

```
01.img_downloader/
├── downloader.py          # 메인 다운로더 프로그램
├── analyze_errors.py      # 에러 분석 유틸리티
├── requirements.txt       # Python 패키지 의존성 목록
├── env.sample             # 환경 변수 설정 샘플 파일
├── README.md              # 프로젝트 사용 가이드
└── logs/                  # 로그 파일 저장 디렉토리
    └── image_downloader_YYYY-MM-DD.log
```

주요 파일 설명:

- **downloader.py**: CSV 파일을 읽어 이미지를 다운로드하는 핵심 모듈. 환경 변수 로드, CSV 파싱, 멀티스레드 다운로드, 에러 처리 등 전체 프로세스를 담당합니다.
- **analyze_errors.py**: 다운로드 실패 시 생성되는 에러 CSV 파일을 분석하여, 실패 원인별 개수 집계와 샘플 데이터를 제공하는 독립 실행 유틸리티입니다.
- **requirements.txt**: 프로젝트 실행에 필요한 외부 Python 패키지 목록을 정의합니다.
- **env.sample**: 환경 변수 설정 예시 파일로, 실제 운영 시 `.env` 파일로 복사하여 사용합니다.
- **logs/**: 실행 로그가 날짜별로 저장되는 디렉토리입니다. 로그 파일명은 `image_downloader_YYYY-MM-DD.log` 형식을 따릅니다. (서버 환경에서는 일반적으로 `/home/iitp-app/IITP-DABT-DataCollector/logs/` 디렉토리에 저장됨)

2.2. 의존성 패키지

프로젝트는 다음과 같은 외부 패키지를 사용합니다:

- **requests (2.32.3)**: HTTP/HTTPS 요청을 처리하는 라이브러리. 이미지 다운로드를 위한 GET 요청, HEAD 요청, 세션 관리, SSL 검증 등을 담당합니다.
- **python-dotenv (1.0.1)**: `.env` 파일에서 환경 변수를 로드하는 라이브러리. 설정값을 코드와 분리하여 관리할 수 있도록 지원합니다.

이 두 패키지는 Python 표준 라이브러리만으로는 부족한 HTTP 통신 및 환경 설정 관리 기능을 제공합니다. 나머지 기능들(CSV 처리, 파일 시스템 작업, 멀티스레딩, 로깅 등)은 Python 표준 라이브러리로 구현되어 있어 추가 의존성이 최소화되었습니다.

2.3. 환경 설정

프로젝트는 `.env` 파일을 통해 모든 설정값을 관리합니다. 주요 설정 항목은 다음과 같습니다:

- **LOG_LEVEL**: 로깅 레벨 설정 (DEBUG, INFO, WARNING, ERROR, CRITICAL)
- **ROOT_DIR**: 이미지가 저장될 루트 디렉토리 경로
- **THREADS**: 병렬 다운로드에 사용할 스레드 수

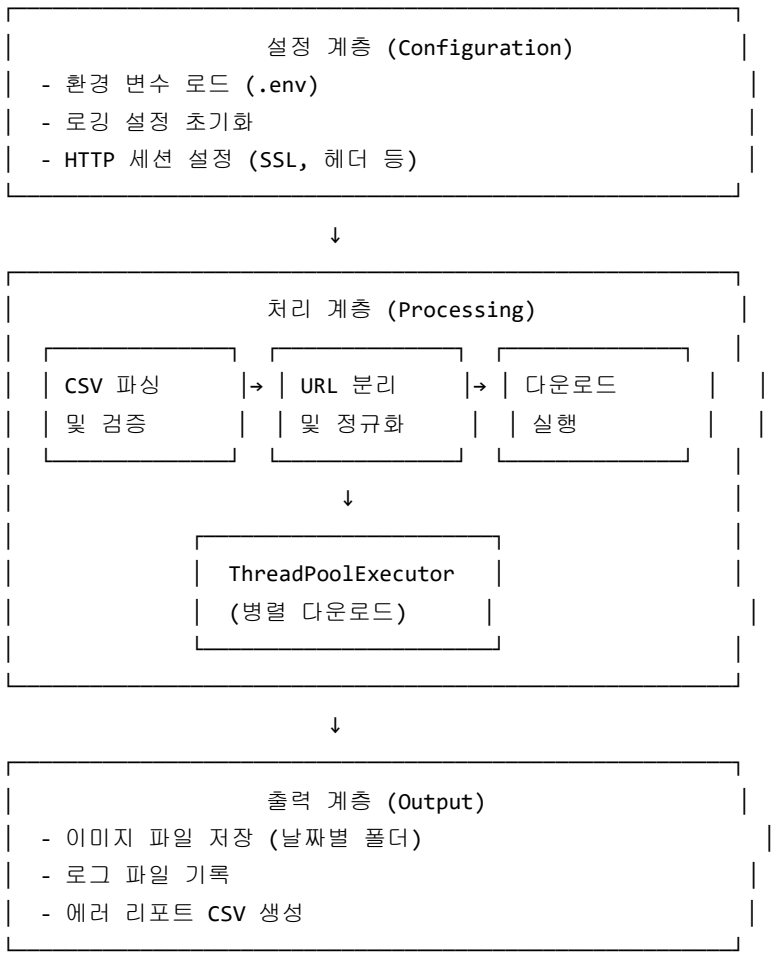
- **URL_CSV_PATH**: 처리할 CSV 파일의 전체 경로
- **HEAD_CHECK**: 다운로드 전 HEAD 요청으로 이미지 타입 검증 여부
- **VERIFY_SSL**: HTTPS 요청 시 SSL 인증서 검증 여부
- **REQUEST_HEADERS_JSON**: 추가 HTTP 헤더를 JSON 형식으로 설정

환경 변수는 스크립트 실행 디렉토리와 현재 작업 디렉토리에서 순차적으로 검색되며, `URL_CSV_PATH` 는 필수 항목이고 나머지는 기본값이 제공됩니다.

3. 프로젝트의 소프트웨어 아키텍처

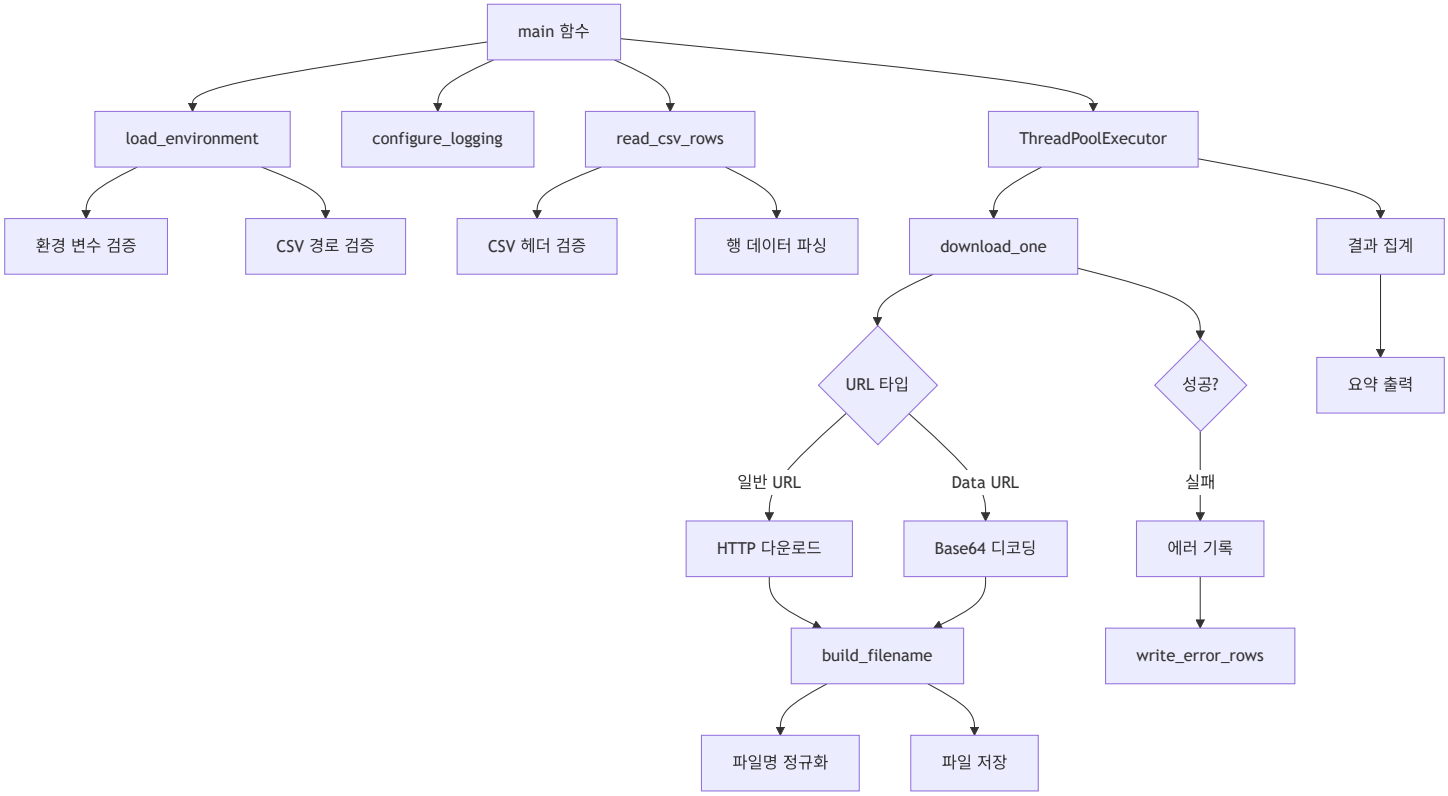
3.1. 시스템 구성도

시스템은 크게 세 가지 계층으로 구성됩니다:



3.2. 모듈 구조

시스템의 주요 모듈과 함수 구조는 다음과 같습니다:

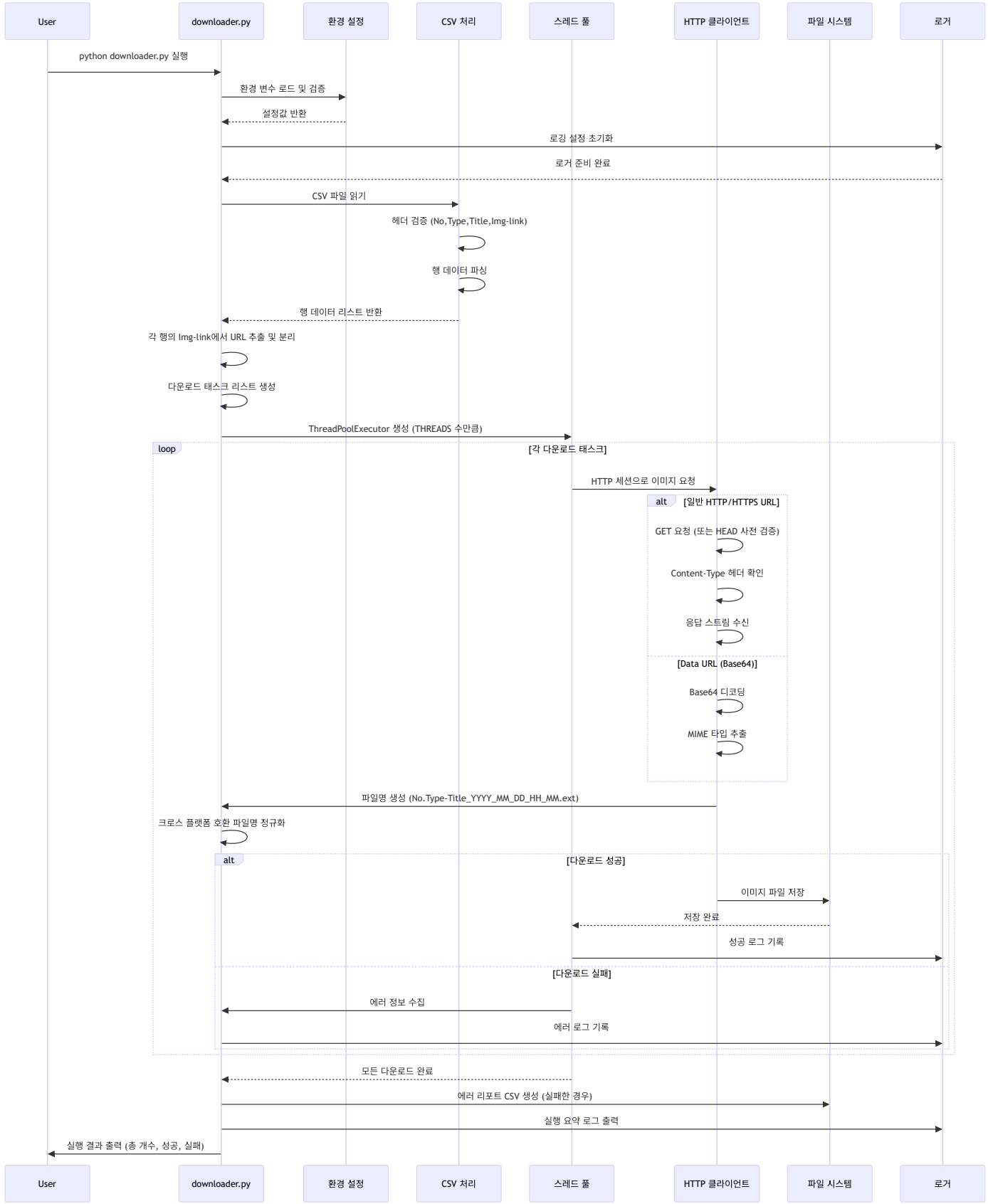


모듈별 역할:

- **환경 설정 모듈**: `load_environment()`, `configure_logging()` - 시스템 초기 설정
- **CSV 처리 모듈**: `read_csv_rows()` - CSV 파일 읽기 및 검증
- **URL 처리 모듈**: `split_urls()`, `fix_malformed_scheme()` - URL 파싱 및 정규화
- **다운로드 모듈**: `download_one()`, `download_data_url()` - 이미지 다운로드 실행
- **파일명 처리 모듈**: `build_filename()`, `sanitize_for_windows()` - 크로스 플랫폼 호환 안전한 파일명 생성
- **에러 처리 모듈**: `write_error_rows()` - 실패 내역 기록
- **확장자 추론 모듈**: `infer_extension_from_headers()`, `infer_extension_from_mime()` - Content-Type 기반 확장자 감지

4. 전체 시스템 연동 Flow

시스템의 전체 실행 흐름은 다음과 같습니다:



주요 처리 단계:

1. **초기화 단계**: 환경 변수 로드, 로깅 설정, HTTP 세션 준비
2. **입력 처리 단계**: CSV 파일 읽기, 헤더 검증, URL 추출 및 분리
3. **다운로드 실행 단계**: 멀티스레드로 병렬 다운로드 수행
4. **결과 처리 단계**: 성공/실패 집계, 에러 리포트 생성, 요약 정보 출력

5. 전체 시스템의 기능 상세 설명

5.1. 환경 변수 로드 및 검증

기능 개요:
시스템 시작 시 `.env` 파일에서 설정값을 로드하고 필수 항목의 유효성을 검증합니다.

5.1.1. 처리 흐름

1. **환경 변수 로드**: 현재 작업 디렉토리와 스크립트 디렉토리에서 `.env` 파일을 순차적으로 검색하여 로드합니다.
2. **필수 항목 검증**:
   - `URL_CSV_PATH` : 반드시 제공되어야 하며, 파일 확장자가 `.csv` 인지 확인합니다.
   - 경로는 사용자 홈 디렉토리 표기(~)를 실제 경로로 변환하고 절대 경로로 정규화합니다.
3. **선택 항목 기본값 설정**:
   - `LOG_LEVEL` : 기본값 "INFO"
   - `ROOT_DIR` : 미지정 시 현재 작업 디렉토리의 `downloads` 폴더로 설정 (서버 환경에서는 일반적으로 `/home/iitp-app/IITP-DABT-DataCollector/downloads`)
   - `THREADS` : 기본값 "8", 양의 정수여야 함
   - `HEAD_CHECK` : 기본값 "false" (boolean 값으로 파싱: "true", "false", "1", "0", "yes", "no", "y", "n", "on", "off" 등 지원)
   - `VERIFY_SSL` : 기본값 "true" (boolean 값으로 파싱: "true", "false", "1", "0", "yes", "no", "y", "n", "on", "off" 등 지원)
   - `REQUEST_HEADERS_JSON` : 기본값 빈 문자열
4. **검증 실패 처리**: 필수 항목이 없거나 잘못된 형식인 경우 `ValueError` 또는 `SystemExit` 예외를 발생시켜 프로그램을 종료합니다.

5.1.2. 예외 처리

- CSV 파일이 존재하지 않거나 확장자가 `.csv` 가 아닌 경우 즉시 종료
- `THREADS` 값이 양의 정수가 아닌 경우 예외 발생

5.2. CSV 파일 읽기 및 검증

기능 개요:
CSV 파일을 읽어 표준 형식에 맞는지 검증하고, 행 데이터를 파싱하여 딕셔너리 리스트로 반환합니다.

5.2.1. 입력 CSV 데이터 포맷

입력 CSV 파일은 다음 형식을 따라야 합니다:

헤더 (필수, 첫 번째 행):

- 정확히 `No,Type,Title,Img-link` (대소문자 구분, 공백 없음)
- 순서는 반드시 위와 동일해야 함

데이터 행:

- 각 행은 4개의 필드를 포함해야 함
- 필드 구분자는 콤마(,) 사용
- 필드값에 콤마가 포함된 경우 따옴표로 감싸야 함 (표준 CSV 규칙)

컬럼 설명:

- **No**: 행 번호 또는 식별자 (문자열, 숫자 모두 가능)
- **Type**: 이미지 유형 분류 (예: "교육", "행사" 등)
- **Title**: 이미지 제목 또는 설명
- **Img-link**: 이미지 URL 또는 여러 URL (구분자: 줄바꿈, 콤마, 세미콜론, 탭, 공백)

인코딩:

- UTF-8 BOM 인코딩 지원 (Excel에서 저장한 CSV 파일 호환)
- UTF-8 인코딩도 지원

예시:

```
No,Type,Title,Img-link
1,교육,장애인직업교육프로그램,https://example.com/image1.jpg
2,행사,세미나안내,https://example.com/img2.jpg,https://example.com/img3.jpg
3,교육,온라인강의,...
```

5.2.2. 처리 흐름

1. **파일 인코딩 처리**: UTF-8 BOM 인코딩을 지원하여 Excel에서 저장한 CSV 파일도 정상적으로 읽을 수 있습니다.
2. **헤더 검증**:
   - CSV의 첫 번째 행이 정확히 `No,Type,Title,Img-link` (대소문자 구분)인지 확인합니다.
   - 헤더가 일치하지 않으면 프로그램을 종료합니다.
3. **행 데이터 파싱**:
   - 각 행의 필드값에서 앞뒤 공백과 줄바꿈 문자를 제거합니다.
   - `Title` 필드의 내부 연속 공백을 단일 공백으로 정규화합니다.
   - 빈 필드는 빈 문자열로 처리합니다.
4. **데이터 반환**: 검증된 행 데이터를 딕셔너리 리스트로 반환합니다.

5.2.3. 예외 처리

- CSV 파일에 헤더가 없는 경우 `ValueError` 발생
- 헤더 형식이 일치하지 않는 경우 `SystemExit` 로 프로그램 종료
- 파일 읽기 오류 시 예외 전파

5.3. 이미지 URL 파싱 및 분리

기능 개요:
CSV의 `Img-link` 필드에서 하나 이상의 URL을 추출하고, 다양한 형식의 URL을 정규화합니다.

5.3.1. 처리 흐름

1. **Data URL 처리**:
   - `data:image/...` 형식의 Base64 인코딩된 이미지 URL을 먼저 추출합니다.
   - Data URL 내부의 콤마는 URL의 일부이므로 분리 기준에서 제외합니다.
2. **일반 URL 분리**:
   - 줄바꿈(`\n` , `\r`), 탭(`\t`), 콤마(`,`), 세미콜론(`;`), 공백을 구분자로 사용하여 URL을 분리합니다.
   - URL 인코딩된 줄바꿈(`%0A`)도 실제 줄바꿈으로 변환하여 처리합니다.
3. **URL 정규화**:
   - 잘못된 스킴 형식(`http:/example.com` → `http://example.com`)을 자동 수정합니다.
   - `http://` , `https://` , `data:image/` 로 시작하는 URL만 유효한 것으로 간주합니다.
4. **중복 제거**: 동일한 URL이 여러 번 나타나는 경우 첫 번째 발생만 유지하여 중복 다운로드를 방지합니다.
5. **인덱스 할당**: 하나의 행에 여러 URL이 있는 경우, 각 URL에 1부터 시작하는 인덱스를 부여하여 파일명에 반영합니다.
6. **빈 URL 처리**: `Img-link` 필드가 비어있거나 유효한 URL이 추출되지 않은 경우, 빈 URL로 태스크를 생성하여 "Empty Img-link" 에러로 기록됩니다.

5.3.2. 지원하는 URL 형식

- 단일 URL: `https://example.com/image.jpg`
- 여러 URL (줄바꿈 구분): `https://example.com/img1.jpg\nhttps://example.com/img2.jpg`
- 여러 URL (콤마 구분): `https://example.com/img1.jpg,https://example.com/img2.jpg`
- Data URL: `...`
- 혼합 형식: 일반 URL과 Data URL이 함께 포함된 경우

5.4. 이미지 다운로드 처리

기능 개요:
각 이미지 URL에 대해 HTTP 요청을 수행하거나 Data URL을 디코딩하여 이미지 데이터를 획득합니다.

5.4.1. 일반 HTTP/HTTPS URL 처리

1. **HEAD 요청 (선택적)**:
   - `HEAD_CHECK=true` 인 경우, 다운로드 전에 HEAD 요청으로 `Content-Type` 헤더를 확인합니다.
   - `Content-Type` 이 `image/*` 로 시작하지 않으면 다운로드를 건너뜁니다.
   - HEAD 요청이 실패해도 GET 요청으로 폴백(fallback)합니다.
2. **GET 요청**:
   - `requests.Session` 을 사용하여 스트리밍 방식으로 이미지를 다운로드합니다.
   - 기본 User-Agent: `img-downloader/1.0 (+https://example.local)` 가 자동으로 설정됩니다.
   - 타임아웃 설정: 연결 10초, 읽기 60초
   - SSL 검증은 `VERIFY_SSL` 설정에 따라 제어됩니다.
   - 커스텀 헤더는 `REQUEST_HEADERS_JSON` 에서 로드하여 세션에 적용됩니다 (기본 User-Agent를 덮어쓸 수 있음).
3. **응답 처리**:
   - HTTP 상태 코드가 2xx가 아니면 예외를 발생시켜 실패로 처리합니다.
   - 응답 본문을 8KB 청크 단위로 읽어 디스크에 저장합니다.

IITP DABT DataCollector 설계서

5.4.2. Data URL 처리

1. **URL 파싱**: `data:image/base64,` 형식에서 MIME 타입과 데이터 부분을 분리합니다.
2. **디코딩**:
   - `base64` 플래그가 있으면 Base64 디코딩을 수행합니다.
   - 없으면 URL 디코딩(`unquote_to_bytes`)을 수행합니다.
3. **MIME 타입 추출**: MIME 타입에서 확장자를 추론하여 파일 저장 시 사용합니다.

5.4.3. 예외 처리

- 네트워크 오류, 타임아웃, HTTP 오류 등은 모두 캐치하여 에러 메시지로 기록합니다.
- 실패한 다운로드는 에러 리포트에 포함됩니다.

5.5. 파일명 생성 및 저장

기능 개요:
CSV 행 데이터와 타임스탬프를 기반으로 크로스 플랫폼 호환 파일명을 생성하고 이미지를 저장합니다.

5.5.1. 파일명 생성 규칙

1. **기본 형식**: `{No}.{Type}-{Title}_ {YYYY_MM_DD_HH_MM}{인덱스}.{확장자}`
   - 예: `1.교육-장애인직업교육프로그램_ 2025_01_27_14_30.jpg`
   - Title과 타임스탬프 사이에 공백 하나가 포함됩니다 (_).
   - 같은 행에 여러 URL이 있는 경우 인덱스가 추가됩니다: `..._14_30_1.jpg` , `..._14_30_2.jpg`
2. **Title 길이 제한**:
   - `Title` 필드가 120자를 초과하면 자동으로 잘라냅니다.
   - 전체 파일명이 240자를 초과하면 추가로 잘라냅니다.
3. **크로스 플랫폼 호환 처리**:
   - Windows 및 Linux/Unix에서 문제가 될 수 있는 문자(` : " / \ | ? *` 및 제어 문자)를 언더스코어(_)로 치환합니다.
   - 연속된 공백을 단일 공백으로 정규화합니다.
   - 파일명 끝의 점이나 공백을 제거합니다.
4. **확장자 처리**:
   - URL 경로에 확장자가 있으면 우선 사용합니다.
   - `.do` 확장자는 무시하고 Content-Type에서 추론합니다.
   - 확장자가 없으면 HTTP 응답의 `Content-Type` 헤더를 분석하여 추론합니다.
     - 주요 이미지 타입(jpeg, jpg, png, gif, webp, bmp, tiff)은 미리 정의된 맵핑 테이블을 사용합니다.
     - 그 외의 타입은 Python의 `mimetypes` 모듈을 사용하여 추론합니다.
     - `.jpe` 확장자는 자동으로 `.jpg` 로 정규화됩니다.
   - 추론이 불가능한 경우 기본값 `.jpg` 를 사용합니다.

5.5.2. 저장 경로

- 루트 디렉토리: `ROOT_DIR` 환경 변수로 지정
- 날짜 폴더: `ROOT_DIR/YYYY-MM-DD/` 형식으로 자동 생성
- 예: `/home/iitp-app/IITP-DABT-DataCollector/downloads/2025-08-27/1.교육-제목_ 2025_01_27_14_30.jpg`

15 / 22

5.5.3. 중복 파일명 처리

- 동일한 파일명이 이미 존재하는 경우, 파일시스템의 기본 동작에 따라 덮어쓰기됩니다.
- 인덱스 접미사로 구분되는 경우 중복 가능성이 낮습니다.

5.6. 에러 처리 및 로깅

기능 개요:
다운로드 실패 시 상세한 에러 정보를 수집하고, 로그 파일과 에러 리포트 CSV에 기록합니다.

5.6.1. 에러 정보 구조

각 실패한 다운로드에 대해 다음 정보를 수집합니다:

- 원본 CSV 행 데이터: `No`, `Type`, `Title`, `Img-link`
- 실패한 URL: 실제 다운로드를 시도한 URL
- URL 인덱스: 행 내에서 몇 번째 URL인지 (1부터 시작)
- 에러 메시지: 실패 원인에 대한 설명

5.6.2. 에러 리포트 CSV 데이터 포맷

하나 이상의 다운로드가 실패한 경우, 원본 CSV 파일과 같은 디렉토리에 `{원본파일명}_errorRow.csv` 파일이 생성됩니다. 에러 리포트는 UTF-8 BOM 인코딩으로 저장되어 Excel에서 바로 열 수 있습니다.

에러 리포트 CSV 파일은 다음 형식을 따릅니다:

헤더 (첫 번째 행):

- `No,Type,Title,Img-link,url_index,url,error`

컬럼 설명:

- **No**: 원본 CSV의 행 번호 또는 식별자
- **Type**: 원본 CSV의 Type 필드값
- **Title**: 원본 CSV의 Title 필드값
- **Img-link**: 원본 CSV의 Img-link 필드값 (전체)
- **url_index**: 실패한 URL이 해당 행의 Img-link에서 몇 번째 URL인지 (1부터 시작)
- **url**: 실제로 다운로드를 시도했으나 실패한 URL
- **error**: 실패 원인에 대한 에러 메시지

인코딩:

- UTF-8 BOM 인코딩 (Excel 호환)

예시:

```
No,Type,Title,Img-link,url_index,url,error
5,교육,프로그램안내,https://example.com/img1.jpg,1,https://example.com/img1.jpg,404 Client Error: Not Found
10,행사,세미나,https://example.com/img2.jpg,1,https://example.com/img2.jpg,Timeout
```

참고 사항:

- 하나의 행에 여러 URL이 있고 일부만 실패한 경우, 각 실패한 URL마다 별도의 행이 생성됩니다.
- `Img-link` 필드가 비어있거나 URL이 추출되지 않은 경우, `url_index` 는 1이고 `url` 은 빈 문자열이며 `error` 는 "Empty Img-link"입니다.

5.6.3. 로깅

1. **로그 레벨**: 환경 변수 `LOG_LEVEL` 로 제어되며, DEBUG, INFO, WARNING, ERROR, CRITICAL을 지원합니다.
2. **로그 출력 위치**:
   - 콘솔: 표준 출력으로 실시간 진행 상황 표시
   - 파일: 현재 작업 디렉토리의 `logs/image_downloader_YYYY-MM-DD.log` 에 일일 로그 저장 (서버 환경에서는 일반적으로 `/home/iitp-app/IITP-DABT-DataCollector/logs/image_downloader_YYYY-MM-DD.log`)
3. **로그 내용**:
   - 시작 정보: CSV 경로, 출력 디렉토리, 로그 파일 경로, 스레드 수, 설정값
   - 진행 상황: 각 이미지 저장 성공 시 파일 경로 기록
   - 에러 정보: 실패한 다운로드의 행 번호, 제목, URL 인덱스, 에러 메시지
   - 실행 요약: 총 링크 수, 성공 수, 실패 수, 에러 리포트 파일 경로

5.6.4. 예외 처리

- 예상치 못한 예외가 발생하면 전체 스택 트레이스를 로그에 기록하고 프로그램을 종료합니다.
- 시스템 종료 메시지(`SystemExit`)는 콘솔에도 출력됩니다.

5.7. 에러 분석 유틸리티

기능 개요:

`analyze_errors.py` 는 다운로드 실패 후 생성된 에러 리포트 CSV를 분석하여 실패 원인별 개수 집계와 샘플 데이터를 제공합니다.

5.7.1. 처리 흐름

1. **에러 리포트 읽기**:
   - 명령줄 인자로 전달된 `*_errorRow.csv` 파일을 읽습니다.
   - 필수 헤더(`No, Type, Title, Img-link, error`) 존재 여부를 검증합니다.
2. **에러 필터링**:
   - `error` 필드가 비어있거나 "Empty Img-link"인 항목은 제외합니다.
   - 실제 다운로드 시도 후 실패한 항목만 분석 대상으로 포함합니다.
3. **에러 원인별 개수 집계**:
   - 동일한 에러 메시지를 가진 항목들을 그룹화하여 개수를 집계합니다.
   - 에러 메시지별로 빈도순으로 정렬하여 출력합니다.
4. **샘플 데이터 제공**:
   - 각 에러 유형별로 최대 5개의 샘플 행을 수집합니다.
   - 샘플에는 `No` , `Title` , `Img-link` 정보가 포함됩니다.
5. **요약 CSV 생성**:
   - 에러 리포트와 같은 디렉토리에 `{원본파일명}_summary.csv` 파일을 생성합니다.
   - 컬럼: `reason, count` (에러 원인, 발생 횟수)

5.7.2. 사용 예시

```
python analyze_errors.py /home/iitp-app/IITP-DABT-DataCollector/data/input_errorRow.csv
```

5.7.3. 출력 형식

콘솔에 다음 정보가 출력됩니다:

1. **파일 정보**:
   - 총 행 수: 에러 리포트 CSV 파일의 전체 행 수
   - 실제 에러 행 수: "Empty Img-link"를 제외한 실제 다운로드 실패 행 수
2. **에러 원인별 개수 및 샘플**:
   - 각 에러 메시지별 발생 횟수 (빈도순 정렬)
   - 각 에러 유형별 샘플 데이터 (최대 5개)
     - 샘플에는 `No`, `Title` 정보가 포함됨
3. **요약 파일 정보**:
   - 생성된 요약 CSV 파일의 저장 경로

출력 예시:

```
Total rows in file: 100
Non-empty errors: 15

Distinct failure reasons (excluding 'Empty Img-link'):
- 8 x 404 Client Error: Not Found for url: https://example.com/missing.jpg
    sample -> No=5, Title=교육프로그램
    sample -> No=12, Title=세미나안내
- 5 x Timeout
    sample -> No=20, Title=행사공지
- 2 x Connection refused
    sample -> No=35, Title=강의자료

Summary saved: /home/iitp-app/IITP-DABT-DataCollector/data/input_summary.csv
```

이 유틸리티를 통해 대량의 다운로드 실패 원인을 빠르게 파악하고, 문제가 되는 URL 패턴이나 네트워크 이슈를 식별할 수 있습니다.

6. 부록

A. 환경 변수 상세 설명

다음은 `.env` 파일에 설정 가능한 모든 환경 변수에 대한 상세 설명입니다.

A.1. 필수 환경 변수

URL_CSV_PATH

- **설명**: 처리할 CSV 파일의 전체 경로 (파일명 포함)
- **형식**: 절대 경로 또는 상대 경로 (사용자 홈 디렉토리 표기 `~` 지원)
- **예시**:
  - `/home/iitp-app/IITP-DABT-DataCollector/data/images.csv`
  - `./input.csv`
  - `~/IITP-DABT-DataCollector/data/data.csv`
- **검증**: 파일 확장자가 반드시 `.csv` 여야 함
- **기본값**: 없음 (필수 항목)

A.2. 선택 환경 변수

LOG_LEVEL

- **설명**: 로깅 레벨 설정. 이 레벨 이상의 로그만 기록됨
- **가능한 값**: `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL` (대소문자 구분 없음)
- **기본값**: `INFO`
- **권장값**: 운영 환경에서는 `INFO`, 문제 디버깅 시 `DEBUG`

ROOT_DIR

- **설명**: 다운로드한 이미지가 저장될 루트 디렉토리 경로
- **형식**: 절대 경로 또는 상대 경로
- **예시**:
  - `/home/iitp-app/IITP-DABT-DataCollector/downloads`
  - `./output`
  - `/home/iitp-app/IITP-DABT-DataCollector/images`
- **기본값**: 현재 작업 디렉토리의 `downloads` 폴더 (서버 환경에서는 일반적으로 `/home/iitp-app/IITP-DABT-DataCollector/downloads`)
- **참고**: 실제 저장은 `ROOT_DIR/YYYY-MM-DD/` 하위에 이루어짐

THREADS

- **설명**: 병렬 다운로드에 사용할 스레드 수
- **형식**: 양의 정수
- **기본값**: `8`

- **권장값**:
  - 네트워크 대역폭이 충분한 경우: 8~16
  - 서버 부하를 고려해야 하는 경우: 4~8
  - 매우 느린 네트워크: 2~4
- **주의**: 과도하게 높은 값은 오히려 성능 저하를 유발할 수 있음

HEAD_CHECK

- **설명**: 다운로드 전에 HEAD 요청으로 이미지 타입을 사전 검증할지 여부
- **가능한 값**: `true`, `false`, `1`, `0`, `yes`, `no`, `y`, `n`, `on`, `off` (대소문자 구분 없음)
- **기본값**: `false`
- **사용 시나리오**:
  - `true` : Content-Type이 `image/*` 가 아닌 URL은 다운로드하지 않음 (대역폭 절약)
  - `false` : 모든 URL을 다운로드 시도 (일부 서버는 HEAD를 지원하지 않을 수 있음)

VERIFY_SSL

- **설명**: HTTPS 요청 시 SSL 인증서 검증 여부
- **가능한 값**: `true`, `false`, `1`, `0`, `yes`, `no`, `y`, `n`, `on`, `off` (대소문자 구분 없음)
- **기본값**: `true`
- **주의**:
  - 운영 환경에서는 반드시 `true` 로 설정 권장
  - 자체 서명 인증서를 사용하는 내부 서버의 경우에만 `false` 고려
  - `false` 설정 시 중간자 공격(Man-in-the-Middle)에 취약해질 수 있음

REQUEST_HEADERS_JSON

- **설명**: HTTP 요청에 추가할 커스텀 헤더를 JSON 형식으로 설정
- **형식**: 유효한 JSON 객체 문자열
- **예시**:
  - `{"Referer":"https://example.com"}`
  - `{"User-Agent":"Mozilla/5.0 (compatible; DataCollector/1.0)"}`
  - 참고: 기본 User-Agent는 `img-downloader/1.0 (+https://example.local)` 이며, `REQUEST_HEADERS_JSON` 에서 User-Agent를 지정하면 이를 덮어씁니다.
  - `{"Authorization":"Bearer token123","X-Custom-Header":"value"}`
- **기본값**: 빈 문자열 (추가 헤더 없음)
- **사용 시나리오**:
  - 특정 사이트에서 Referer 헤더를 요구하는 경우
  - 커스텀 인증 토큰이 필요한 경우
  - User-Agent를 변경해야 하는 경우
- **주의**: JSON 형식이 잘못된 경우 경고만 출력하고 무시됨

B. 로그 레벨별 상세 설명

시스템은 설정된 로그 레벨에 따라 다른 수준의 정보를 기록합니다. 로그 레벨은 심각도 순서대로 다음과 같습니다:

```
DEBUG < INFO < WARNING < ERROR < CRITICAL
```

설정한 레벨 이상의 로그만 기록됩니다. 예를 들어 `INFO` 로 설정하면 `INFO` , `WARNING` , `ERROR` , `CRITICAL` 로그가 기록되고 `DEBUG` 로그는 기록되지 않습니다.

B.1. DEBUG 레벨

설명: 가장 상세한 디버깅 정보를 제공합니다. 개발 및 문제 해결 시에만 사용 권장.

기록되는 내용:

- 로깅 설정 초기화 완료 정보: 설정된 로그 레벨과 로그 파일 경로
- HEAD 요청 실패 시 GET 요청으로 폴백하는 경우의 상세 메시지

사용 시나리오:

- HEAD 요청이 실패하는 원인 파악
- 로깅 설정이 올바르게 적용되었는지 확인

B.2. INFO 레벨

설명: 일반적인 실행 정보를 제공합니다. 운영 환경에서 권장되는 기본 레벨입니다.

기록되는 내용:

- 프로그램 시작 정보: CSV 파일 경로, 출력 디렉토리, 로그 파일 경로, 스레드 수, HEAD_CHECK 설정값, VERIFY_SSL 설정값
- 커스텀 HTTP 헤더가 성공적으로 병합된 경우
- 각 이미지 다운로드 성공 시 저장된 파일 경로 (일반 URL 및 Data URL 모두)
- 실행 요약: 총 링크 수, 성공 수, 실패 수, 에러 리포트 파일 경로

사용 시나리오:

- 정상 운영 시 진행 상황 모니터링
- 다운로드 완료 후 결과 확인
- 일일 실행 로그 검토

B.3. WARNING 레벨

설명: 잠재적인 문제나 비정상적인 상황을 알립니다. 프로그램은 계속 실행됩니다.

기록되는 내용:

- `REQUEST_HEADERS_JSON` 환경 변수 파싱 실패 (잘못된 JSON 형식인 경우)
- CSV 파일에서 추출된 URL이 하나도 없는 경우

사용 시나리오:

- 설정 오류가 있지만 프로그램이 계속 실행 가능한 경우
- 일부 비정상적인 상황이 발생했지만 전체 작업에는 영향이 없는 경우

B.4. ERROR 레벨

설명: 다운로드 실패와 같은 오류 상황을 기록합니다. 작업은 계속 진행됩니다.

기록되는 내용:

- 개별 이미지 다운로드 실패: 행 번호(No), 제목(Title), URL 인덱스, 에러 메시지
- 예상치 못한 예외 발생 시 전체 스택 트레이스 (logger.exception 사용)
- 프로그램 실행 중 치명적 오류 발생 시 전체 스택 트레이스

사용 시나리오:

- 일부 이미지 다운로드가 실패한 경우 원인 파악
- 에러 리포트와 함께 실패 원인 분석

B.5. CRITICAL 레벨

설명: CRITICAL 레벨 로그는 실제 소스 코드에서 사용되지 않습니다.

참고 사항:

- 환경 변수 필수 항목 누락, CSV 파일 확장자 오류, CSV 헤더 형식 불일치 등의 오류는 `SystemExit` 예외로 처리되며, 이는 `print()` 함수를 통해 콘솔에 직접 출력됩니다.
- 이러한 오류는 로그 파일이 아닌 표준 출력으로만 표시되므로, 프로그램 실행 시 콘솔 출력을 확인해야 합니다.
- 예상치 못한 예외는 ERROR 레벨의 `logger.exception()` 으로 기록됩니다.

B.6. 로그 파일 관리

- **저장 위치**: 현재 작업 디렉토리의 `logs/image_downloader_YYYY-MM-DD.log` (서버 환경에서는 일반적으로 `/home/iitp-app/IITP-DABT-DataCollector/logs/image_downloader_YYYY-MM-DD.log`)
- **인코딩**: UTF-8
- **로그 파일명 형식**: 실행 날짜 기준으로 일일 로그 파일 생성
- **로그 포맷**: `%(asctime)s [%(levelname)s] %(message)s`
  - 예:

    ```
    2025-08-27 14:30:15,123 [INFO] Saved: /home/iitp-app/IITP-DABT-DataCollector/downloads/2025-01-27/image.jpg
    ```
- **로그 롤오버**: 날짜가 변경되면 새로운 로그 파일이 생성됨 (기존 파일은 유지)