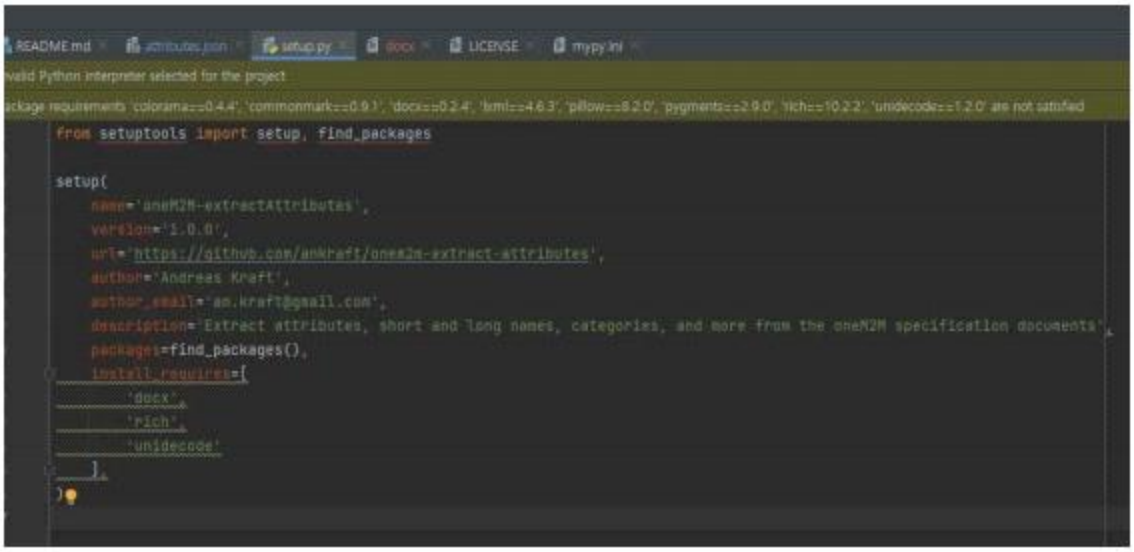
1. setup.py 에 대하여

-파이썬에서 사용 가능한 다양한 패키지를 설치하는데 사용된다.
-파이썬에 수동으로 패키지를 설치해야하는 경우 주로 사용한다.




```
from setuptools import setup, find_packages

setup(
name='oneM2M-extractAttributes',
version='1.0.0',
url='https://github.com/ankraft/onem2m-extract-attributes',
author='Andreas Kraft',
author_email='an.kraft@gmail.com',
description='Extract attributes, short and long names, categories, and more from
the oneM2M specification documents',
packages=find_packages(),
        install_requires=[
                'docx',
                'rich',
                'unidecode'
        ],
)
```

from setuptools import setp, find_packages

* 파이썬 문법 from 파일명(라이브러리) import 함수이름
>>from은 import에서 사용했던 calender.함수명() 이름에서 앞에 부분인 파일명을 없애고 함수명만 사용해서 간단하게 쓰기위해 사용한다.
>>import는 이미 만들어진 파이썬 프로그램 파일, 라이브러리안에 있는 파일등을 사용할 수 있게 해주는 명령어.

*from setuptools *import setup, find_packages는 결국 setuptools파일을 setup 함수명으로 사용하겠다는 것이다.

-setuptools파일에 있는 기능들 name, version, url, author, author_email, description,

의 기능들에 해당하는 정보를 대입한다.

-대부분 setuptools.find_packages 함수를 사용하여 자동으로 포함될 package들을 찾게 하고 대신에 제외되어야 할 package들을 exclude 인자를 통해 설정한다.

-packages=find_packages()하여 패키지를 찾고 install_requires(=모듈 의존성 관리)를 통해서 docx, rich, unidecode (docx만인식한다는 말 예전 doc는 인식은 안함, rich 라이브러리 파이썬 텍스트 하는 친구, unidecode 유니코드)

라는 말이다.

=https://github.com/ankraft/onem2m-extract-attributes 에서 oneM2M-extractAttributes 이름인 애를 1.0.0버전으로 설치 한다.
attributes 툴은 oneM2M 사양 문서에서 속성, 짧은 이름 및 긴 이름, 범주 등을 추출한다는 설명과 인식모듈은 docx, rich, unidecode로 관리하도록 해놓았다.

2. Readme.md에 대해

(md란 마크 다운 언어 방언을 통해 마크 다운 문서 파일로 사용되는 텍스트 파일을 말함.)

- Install

python-docx 모듈을 설치해야 합니다
-> pip install python-docx

python-docx가 아닌 docx를 설치할 경우 오류 발생

- Extract Attributes Tool의 기능

oneM2M의 문서에서 속성, 범주 등을 추출합니다.

- Installation

이 스크립트를 실행하기 위해서 Python 3.8 이상이 필요하다.
pyenv와 같은 가상환경 사용을 추천한다.

pyenv가하는 일 ...
사용자별로전역 Python 버전을 변경할수 있습니다.
프로젝트별 Python 버전에 대한 지원을 제공합니다.
Python 버전을환경으로 재정의할 수 있습니다. 변수.
한 번에 여러 버전의 Python에서 명령을 검색합니다. 이것은tox를 사용하여 Python 버전에서 테스트하는 데 도움이 될 수 있습니다.

다음 명령어를 사용하여 추가로 필요한 Python 모듈을 설치한다.

```
python3 -m pip install -r requirements.txt
```

- Running

Preparing the input documents
이 스크립트는 docx 형식의 문서를 받아들인다. 필요한 경우, 기존 문서를 “docx” 문서로 열고 저장하여 형식을 변환해야 한다.
Word의 “doc” 형식과 같은 이전 형식은 지원되지 않는다.

다음 명령을 사용하여 ‘TS-0022-Field_Device_Configuration-V4_2_0.docx’ 문서를 입력으로 읽고 기본 출력 디렉토리에서 속성 정의가 있는 ‘attributes.json’(default) 파일을 생성한다.

```
python3 src/extractAttributes.py TS-0022-Field_Device_Configuration-V4_2_0.docx
```

다음 명령을 사용하여 동일한 디렉토리에 있는 모든 ‘.docx’ 문서를 읽고 처리한 후 ‘attribute.json’ 파일을 생성할 수 있다.

```
python3 src/extractAttributes.py *.docx -o out
```

다음 명령어는 이전 예시와 유사하지만, 각 입력 문서마다 하나씩 CSV 파일을 생성한다.

```
-> python3 src/extractAttributes.py *.docx --csv -o out
```

다음 명령어를 사용하여 속성의 JSON 파일 외에도 화면에 결과를 나열할 수 있다. 중복 정의는 빨간색으로 표시된다.

```
-> python3 src/extractAttributes.py *.docx --list
```

중복 여부만 확인하려는 경우에는 출력된 내용이 너무 많을 수 있다. 따라서 다음 명령을 사용하여 화면에 중복 정의만 나열할 수 있다. 여기에는 여러번 정의된 속성뿐만 아니라 여러 개의 짧은 이름이 정의된 속성도 별도의 테이블에 나열된다.
—>python3 src/extractAttributes.py *.docx --list-duplicates

명령어는 출력 디렉토리에 CSV 파일인 'duplicates.csv' 파일 및
'duplicates_shortnames.csv' 를 생성하는 것을 제외하고 이전 명령과 유사하다.
—>python3 src/extractAttributes.py *.docx --list-duplicates --csv

-oneM2M의 specification documents에서 속성, 짧은 이름 및 긴 이름, 범주 등 을 추출한다.
-이 스크립트는 oneM2M specification documents를 가져와서 short name 정의 테이블을 검색한다. 이후 속성, short name, 범주 및 기타정보를 포함하는 JSON 구조를 생성한다.
-쉽게 맵핑 할 수 있도록 CSV(Comma Separated Values) 파일을 option 기능으로 생성할 수 있으며 중복 특성 정의에 대한 보고서를 생성할 수 있다.
-생성된 output은 API 문서를 지원하거나 응용프로그램에서 속성 맵핑을 자동으로 생성하는 부분에 사용될 수 있다.

-attributes.json에서 자주 보이는 문서는 TS-0004d인데 서비스계층핵심프로토콜 이라고 쓰여있다.

3. attributes.json에 대하여

```
[ {
        "shortname": "op", //짧은 이름 : op
        "attribute": "operation", //속성 : 사업
        "occursIn": // 'occurs in' entry(입력값)들을 나누고 공백을 없앤다. [
            "Request", -> 이게 입력값 들
            "authorizationDecision", -> 이게 입력값 들
            "dynAuthDasRequest", -> 이게 입력값 들
            "operationMonitor", -> 이게 입력값 들
            "request" -> 이게 입력값 들
        ],
```

```
        "categories": [ //카테고리
            "Complex Data Types",
            "Primitive Parameters" (기본 매개변수),
            "Resource Attributes"
        ],
        "documents": [ //문서
            "ts-0004.docx"
        ]
    },
```

----------**여기까지만 본다면**

“shortname” => 속성 정의를 내리는 짧은 말
“attribute" => 속성 그 자체 이름
"occursIn" => 해당 속성에는 무슨 입력 값들이 있는지
"categories" => 해당 속성은 어떤 형태의 엔트리(입력값?)을 가지는지
“doucuments" => tool에 넣을 문서파일
5가지의 단어가 중복되는 것을 볼 수 있다.

■((Primitive Parameters예시))
자바스크립트에는 4가지 Primitive Data Types가 있다.
1. String: 텍스트 값. 따옴표로 지정되는 데이터
-> 예) var car_name = "기아";
2. Number: 숫자 값.
-> 예) var car_year = 2017;
3. Boolean: True(참) 혹은 False(거짓) 값.
-> 예) var new_car = true;
4. undefined: 아무 값이 지정 안 된다.
■((Complex Data Types 예시))
Primitive Data Types에 포함되지 않은 오브젝트(Object) 타입과 함수(Function) 타입의 두 가지가 있다.
-> object = { name: 'NAME', age: 29 }; 뭐 이렇게 타입을 마음대로 할당 가능하다.
-> Function함수 기능은 안에 뭐 기능 넣어서 불러오기 이름() 이렇게 쓰이는 것
■((Resource Attributes 예시)) = 직역: 자원 속성
->선택자.setAttribute(속성명, 속성값) 뭐 이런 구조로 들어가 있는 정보

```
    {
        "shortname": "to",
        "attribute": "to",
        "occursIn": [
            "Request",
            "Response",
```

```
            "authorizationDecision",
            "authorizationPolicy"
        ],
        "categories": [
            "Primitive Parameters",
            "Resource Attributes"
        ],
        "documents": [
            "ts-0004.docx"
        ]
},
```