Attributes Tool 최종 코드 분석

rich : 터미널 환경에서의 텍스트 출력을 예쁘게 해주는 라이브러리

pathlib : 파일 위치 찾기와 파일 입출력을 도와주는 라이브러리

typing : 인터프리터 언어인 파이썬에 타입 지정을 할 수 있게 해주는 라이브러리

docx : document를 만들어주는 라이브러리

docx.table : document에 표를 작성하는 라이브러리

```python
@dataclass
class AttributeTable:
    headers:list
    attribute:int
    shortname:int
    occursIn:int
    filename:str
    category:str
```

```python
@dataclass
class Attribute:
    """ Datastruture for an attribute, including shortname, longname, category etc.
    """
    shortname: str          # Lower case variant of the short name
    shortnameOrig: str
    attribute: str
    occurences:int
    occursIn:Set
    categories:Set
    documents:Set

    def asDict(self) -> dict:
        """ Return this dataclass as a dictionary.
        """
        return {    'shortname' :   self.shortnameOrig,
                    'attribute' :   self.attribute,
                    'occursIn'  :   sorted([ v for v in self.occursIn ]),
                    'categories':   sorted([ v for v in self.categories ]),
                    'documents' :   sorted([ v for v in self.documents ])
                }
```

AttributeTable과 Attribute를 dataclass로 지정한다. 또한 asDict함수를 통하여 Attribute 클래스를 딕셔너리 형식으로 리턴한다.

```
Attributes      = Dict[str, Attribute]
AttributesSN    = Dict[str, List[str]]
```

Attributes변수는 str key와 Attribute value를 가진 딕셔너리로 만들고 AttributesSn은 str key와 str list 를 value로 갖는 딕셔너리로 선언한다.

```
console = Console()
```

console 변수를 선언하고 출력을 위한 콘솔창을 띄운다.

```
attributeTables:List[AttributeTable] = [

    # TS-0004
    AttributeTable(headers=['Parameter Name', 'XSD long name', 'Occurs in', 'Short Name'],  attribute=1, shortname=3, occursIn=2,
    AttributeTable(headers=['Root Element Name', 'Occurs in', 'Short Name'],                attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name'],                   attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Resource Type Name', 'Short Name'],                            attribute=0, shortname=1, occursIn=-1,
    AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name'],                      attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Member Name', 'Short Name'],                                   attribute=0, shortname=1, occursIn=-1,
```

```
    # TS-0022
    AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name', 'Notes'],      attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name', 'Notes'],         attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['ResourceType Name', 'Short Name'],                        attribute=0, shortname=1, occursIn=-1,

    # TS-0023
    AttributeTable(headers=['Resource Type Name', 'Short Name'],                       attribute=0, shortname=1, occursIn=-1,
    AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name'],              attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Argument Name', 'Occurs in', 'Short Name'],               attribute=0, shortname=2, occursIn=1,

    # TS-0032
    AttributeTable(headers=['Attribute Name', 'Short Name'],                           attribute=0, shortname=1, occursIn=-1,
    AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name', 'Notes'],      attribute=0, shortname=2, occursIn=1,
    AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name', 'Notes'],         attribute=0, shortname=2, occursIn=1,
```

attributeTables라는 변수를 AttributeTable의 리스트로 정의하고 속성을 정의한 AttributeTable들을 리스트에 넣어준다. 주석을 보면 알수있듯이 각 주석은 처리할 파일들을 의미한다.

파일들은 다음과 같은 역할을 한다.

TS-0004 - Service Layer Core Protocol (서비스 계층 핵심 프로토콜)

TS-0022 - Field Device Configuration (필드 장치 구성)

TS-0023 - SDT based Information Model and Mapping for Vertical Industries (수직산업에 대한 SDT 기반 정보모델 및

맵핑)

TS-0032 - MAF and MEF Interface Specification (MAF 및 MEF 인터페이스 규격)

```
findAttributeTable(table:Table, filename:str) -> Union[AttributeTable, None]
```

파일로 입력받은 문서로부터 AttributeTable을 찾고 리턴해준다. 만일 찾지 못한다면 None을 리턴해주는 함수

```
processDocuments(documents:list[str], outDirectory:str, csvOut:bool) -> Tuple[Attributes, AttributesSN]
```

```
docs                        = {}
ptasks                      = {}
attributes:Attributes       = {}
attributesSN:AttributesSN   = {}
```

docs - 읽은 문서의 딕셔너리 형태

ptasks - 진행한 progress의 딕셔너리

attributes:Attributes - short name 맵핑 ->속성 정의

attributesSN:AttributesSN - 속성이름 맵핑 -> short name의 리스트

with Progress(...) as progress - 콘솔창에서 진행상황을 보여주고 progress라는 변수로 리턴한다.

stopProgress(msg:str='') - 프로그레스에서 작업을 중단하고 콘솔창에 출력한다. msg를 인자로 받아도 되고 안 받아도 된다. 받게 될 경우 msg를 콘솔창에 출력하여 progress를 멈춘 이유를 알려준다.

```
readTask = progress.add_task(f'Reading document{"s" if len(documents)>1 else ""} ...', total=len(documents))
```

task 준비를 위해 progress에 task를 올리고 readTask에 반환한다.

```
for d in documents:
    if not (dp := Path(d)).exists():
        stopProgress(f'[red]Input document "{d}" does not esist')
        return None, None
    if not dp.is_file():
        stopProgress(f'[red]Input document "{d}" is not a file')
        return None, None
    try:
        docs[d] = Document(d)
        ptasks[d] = progress.add_task(f'Processing {d} ...', total = 1000)
        progress.update(readTask, advance=1)
    except docx.opc.exceptions.PackageNotFoundError as e:
        stopProgress(f'[red]Input document "{d}" is not a .docx file')
        return None, None
    except Exception as e:
        stopProgress(f'[red]Error reading file "{d}"')
        console.print_exception()
        return None, None
```

본격적으로 문서를 읽는 부분이다. for문을 통해 문서를 돌아가면서 모두 읽으며 각 오류케이스를 설정해놓았고 오류가 나거나 파일이 존재하지않으면 stopProgress를 통하여 작업을 중단한다. try: 부분이 실질적으로 문서를 읽는 과정인데 문서를 add_task를 통해 processing하고 update를 통해 progress 를 업데이트 해주는 역할을 한다. docs[d]=Document(d)를 통하여 문서를 읽을때마다 docs에 저장한다.

```
checkTask       = progress.add_task('Checking results ...', total=2)
writeTask       = progress.add_task('Writing files ...', total = 2+len(documents) if csvOut else 2)
```

문서를 모두 읽고 남은 작업을 진행한다.

```
for docName, doc in docs.items():
```

읽고 저장한 문서들을 변환하는 과정이다.

```
for table in doc.tables:
```

문서의 table들을 for문으로 돌아가며 progress에 업데이트 한다.

테이블을 찾지못하면 continue 한다.

```
  for r in table.rows[1:]:
      cells = r.cells
      if cells[0].text.lower().startswith('note:') or len(r.cells) != headersLen:
          continue
```

note:로 시작하거나 cell과 header의 길이가 맞지 않다면 스킵한다.

```
attributeName   = unidecode(cells[snt.attribute].text).strip()
shortnameOrig   = unidecode(cells[snt.shortname].text.replace('*', '').strip())
shortname       = shortnameOrig.lower()
occursIn        = map(str.strip, unidecode(cells[snt.occursIn].text).split(',')) if snt.occursIn > -1 else ['n/a']
```

이름을 추출하고 약간의 변환을 strip이나 lower를 통해 수행한다.

만약 snt.occursIn 이 -1 크지 않다면 ['n/a'] 로 반환한다.

```
if not shortname:
        continue
```

shorname 이 없다면 스킵한다.

```
if shortname in attributes:
```

shorname 이 attributes에 있다면 엔트리를 만들거나 업데이트 한다.

```
countDuplicates = 0
for shortname, attribute in attributes.items():
    countDuplicates += 1 if attribute.occurences > 1 else 0
progress.update(checkTask, advance=1)
countDuplicatesSN = 0
for sns in attributesSN.values():
    countDuplicatesSN += 1 if len(sns) > 1 else 0
```

중복이나 shorname 속성의 중복을 count한다.

그후로 attributes.json을 생성하여 작성하고 출력형식이 정해져 있다면 csv파일을 만든다.

```
progress.stop()
console.print(f'Processed short names:              {len(attributes)}')
if countDuplicates > 0:
    console.print(f'Duplicate definitions:               {countDuplicates}')
if countDuplicatesSN > 0:
    console.print(f'Duplicate definitions (short names): {countDuplicatesSN}')
```

progress를 멈추고 console 창에 shorname, 중복 정의, 중복 정의(shortname)의 개수를 출력하고 이 뒤의 return 함수를 통해 attributes와 attributesSN을 리턴한다.

```
def printAttributeTables(attributes:Attributes, attributesSN:AttributesSN, duplicatesOnly:bool = True) -> None:
```

찾은 속성들을 콘솔창에 출력하며 선택적으로 중복 엔트리만 출력하게 할 수 있다.

```
def printAttributeCsv(attributes:Attributes, outDirectory:str = None) -> None:
```

csv파일에 찾은 속성들을 작성한다.

```
def printDuplicateCsv(attributes:Attributes, attributesSN:AttributesSN, outDirectory:str = None) -> None:
```

두 개의 csv파일을 작성한다. 중복된 속성들이나 같은 속성의 중복된 shortname을 작성한다.

```
if __name__ == '__main__':
```

추가로 받은 명령인자들을 처리하는 부분이다.

dest는 함수의 적용 위치를 지정하는 파라미터이다.

metavar은 help= 에서 도움말 메시지를 생성할 때 표시되는 이름을 변경할 수 있다

action='store' 는 추가옵션를 받을지 정하는 부분이며 추가 옵션을 받지 않고 단지 옵션의 유/무만 필요할 경우 action='store_true'를 사용한다.

help는 인자 사용법에 대한 도움말을 출력한다.

--outdir 혹은 -o의 인자는 파일을 저장할 위치를 정하는 인자들이다.

--csv 혹은 -c의 인자는 csv파일을 출력을 결정하는 인자들이다.

--list 혹은 -l은 찾은 속성들의 리스트 출력을 결정하는 인자들이다.

--list-duplicates 혹은 -ld는 중복된 리스트 출력을 결정하는 인자들이다.

```
attributes, attributesSN = processDocuments(sorted(args.document), args.outDirectory, args.csvOut)
```

processDocuments를 통해 attributes와 attributesSN의 값을 받는다.

```
os.makedirs(args.outDirectory, exist_ok = True)

attributes, attributesSN = processDocuments(sorted(args.document), args.outDirectory, args.csvOut)
if not attributes:
    exit(1)
if args.list or args.listDuplicates:
    printAttributeTables(attributes, attributesSN, args.listDuplicates)
    if args.csvOut:
        printAttributeCsv(attributes, args.outDirectory)
        if args.listDuplicates:
            printDuplicateCsv(attributes, attributesSN, args.outDirectory)
```

makedirs를 통해 디렉토리를 만든다. exist_ok가 true이기 때문에 outDirectory가 이미 존재해도 에러없이 넘어간다.

args.list or args.listDuplicates를 입력받은 경우엔 printAttributeTables를 실행하여 찾은 속성들을 콘솔창에 출력한다. 또한 csvOut을 입력받았따면 csv파일을 출력하며 listDuplicates를 입력받았다면 중복되는 속성들까지 csv파일로 출력한다.