|  |  |
| --- | --- |
| Three brushes with paint tube and a swash of paint | AIRE  (Architecture Intake Review Engine) |

|  |
| --- |
| V1 (for Early Engagement) |
| Creates an output folder and many intake form Word files with proper names based on an operational plan Excel file. |
| Methods |
| - open\_home\_menu() -> str  - fill\_in\_template(doc: DocxTemplate, record:dict, date\_time: datetime) -> DocxTemplate  - generate\_file\_name(record: dict, sheet\_name: str) -> str  - generate\_templates(sheet\_name: str) -> None  - get\_statistics() -> None |
| Libraries |
| - os.path: to read an input file, make an output folder, and save output files  - pathlib: to read an input file, make an output folder, and save output files  - datetime: to get today's date and time  - pandas: to read an Excel file as a Pandas dataframe  - numpy: for record in df.to\_dict(orient="records")[:]: (this is a numpy record)  - docxtpl: to create document template files  - alive\_progress: to make a progress bar for generating many intake form Word files  - pprint: to print long lists or dictionaries pretty  - json: to export statistics data as a JSON file |
| Process order \*needs update |
| 1.if \_\_name\_\_ == '\_\_main\_\_':  1.Prompt a sheet name  2.read\_data(sheet\_name)  1.Prompt an operational plan Excel file name  2.Get today's date and time  3.Create an output folder  4.Read the operational plan Excel file as a Pandas dataframe  5.Pandas(orients="record) -> Dictionary (This is the rows of the Pandas dataframe)  6.for row in rows:  1.extract each cell in a row  2.Create a DocxTemplate file using docxtpl  3.Write today's date time and time  4.Make a proper name - def letter(sheet\_name): is used here  5.Save the DocxTemplate file in the output folder |

|  |
| --- |
| V2 (for pre-AGP0 and AGP0) |
| Creates a report Word file based on a rubric Excel file. |
| Methods |
| - open\_home\_menu() -> None  - generate\_pre\_agp\_0\_report() -> None  - generate\_agp\_0\_report() -> None  - get\_assessment\_data() -> list  - get\_similarity(rubric\_text: str, rationale\_text: str) -> float  - get\_rubric\_descriptions() -> list  - get\_today\_date\_time() -> datetime  - get\_risk\_level(risk\_score: int) -> str  - get\_attribute\_score(attribute\_level: str) -> int  - fill\_in\_template(initiative\_name: str, attribute\_levels: list, risk\_score: int, rationales: list, corporate\_or\_cluster: str) -> None  - check\_file\_exist(file\_name: str) -> None  - generate\_output\_folder() -> None:  - generate\_output\_template(output\_template\_name: str) -> DocxTemplate:  - save\_output\_template(doc: DocxTemplate) -> None:  - open\_output\_template() -> None: |
| Libraries |
| - os: to read an input file, make an output folder, and save output files  - pathlib: to read an input file, make an output folder, and save output files  - datetime: to get today's date and time  - docxtpl: to create document template files  - pandas: to use a dataframe  - openpyxl: Excel automation  - difflib: computes similarity |
| Process order \*needs update |
| 1.if \_\_name\_\_ == '\_\_main\_\_':  1.welcome() - print a welcome message  2.menu()  1.Choose one of 3 tasks  - 1)Pre AGP 0 Assessment -> user\_input(ini\_name)  - 2)AGP 0 Assessment -> (To be developed)  - 3)Exit -> exit()  3.user\_input(ini\_name)  1.Prompt an ?operational plan Excel file name  2.Extract cells from the Excel file using openpyxl  3.report(crit\_assessment, score, rationale, name, corporate\_cluster)  4.report(crit\_assessment, score, rationale, name, corporate\_cluster)  1.Create an output folder  2.Create a DocxTemplate file using docxtpl  3.options = criteria()  5.criteria()  1.Read a rubric Excel file as a Pandas dataframe  2.Pandas(orients="record) -> Dictionary (This is the rows of the Pandas dataframe)  3.for row in rows:  1.options.append(row["Description"])  4.return options  6.report(crit\_assessment, score, rationale, name, corporate\_cluster)  1.Make a dictionary 'context' using options and other arguments(crit\_assessment,...)  1.similar() - SequenceMatcher(None, a, b).ratio()  2.Save the DocxTemplate file in the output folder |

|  |  |
| --- | --- |
| Libraries | |
| Name | Documentation Link |
| datetime | [datetime — Basic date and time types — Python 3.10.7 documentation](https://docs.python.org/3/library/datetime.html) |
| os.path | [os.path — Common pathname manipulations — Python 3.10.7 documentation](https://docs.python.org/3/library/os.path.html) |
| pathlib | [pathlib — Object-oriented filesystem paths — Python 3.10.7 documentation](https://docs.python.org/3/library/pathlib.html) |
| numpy | [NumPy documentation — NumPy v1.23 Manual](https://numpy.org/doc/stable/) |
| pandas | [pandas documentation — pandas 1.4.4 documentation (pydata.org)](https://pandas.pydata.org/docs/) |
| docxtpl | [Welcome to python-docx-template’s documentation! — python-docx-template 0.9.x documentation (docxtpl.readthedocs.io)](https://docxtpl.readthedocs.io/en/latest/)  [python-docx — python-docx 0.8.11 documentation](https://python-docx.readthedocs.io/en/latest/)  [Jinja — Jinja Documentation (3.1.x) (palletsprojects.com)](https://jinja.palletsprojects.com/en/3.1.x/) |
| openpyxl | [openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.0.10 documentation](https://openpyxl.readthedocs.io/en/stable/) |
| difflib | [difflib — Helpers for computing deltas — Python 3.10.7 documentation](https://docs.python.org/3/library/difflib.html) |

|  |  |  |
| --- | --- | --- |
| Log | | |
| Date | Name | Description |
| 9/9/2022 | Dongwon Lee | First draft |
|  |  |  |
|  |  |  |