AIRE (Architecture Intake Review Engine) documentation

Details are subject to change.

Requirements

* Early Engagement support function
* Generate intake forms based on an operational plan file and a template + put date and the name of the intake form at the additional page for tracking (as solution branches might change the name)
* Compare old and new operational plan files (on every generation, need to archive used operational plan data for comparison (xlsx, pickle, json, etc))
* Generate a comparison report – xlsx (shows changed cells) and word (shows changed initiative numbers, details, and so on)
* Extract data from the operational plan based on the column name, not by the index number, so that even if new columns are added to the operational plan, intake forms can be generated without issue.
* Pre-AGP 0 support function
* Generate an assessment report based on an assessment file and a template
* Improve the similarity algorithm (semantic analysis, etc)
* AGP 0 support function
* Check the AGP0 requirements of an AGP 0 submission (zip file)

Folder Structure

AIRE

+-- src

| main.py

| +-- early\_engagement

| early\_engagement.py

| generate\_comparison\_reports.py

| generate\_intake\_forms.py

| generate\_intake\_form.py (helper function for generate\_intake\_forms.py)

| generate\_excel\_comparion\_report.py (helper function for generate\_comparison\_reports.py)

| generate\_word\_comparion\_report.py (helper function for generate\_comparison\_reports.py)

| +-- pre\_agp0

| pre\_agp0.py

| (TBD)

| +-- agp0

| agp0.py

| (TBD)

+-- data

| +-- input

| +-- early\_engagement

| +-- pre\_agp0

| +-- agp0

| +-- output

| +-- early\_engagement

| +-- op\_archive

| +-- pre\_agp0

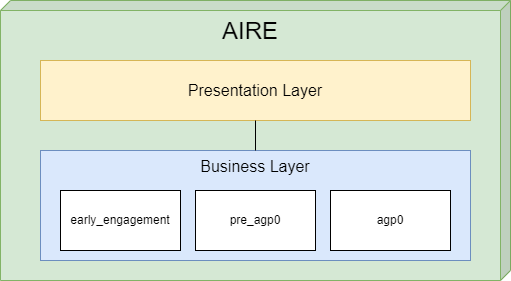
| +-- agp0

+-- docs

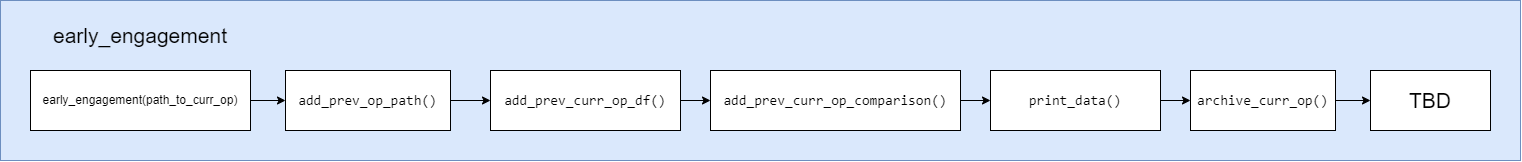
| AIRE docs.docx

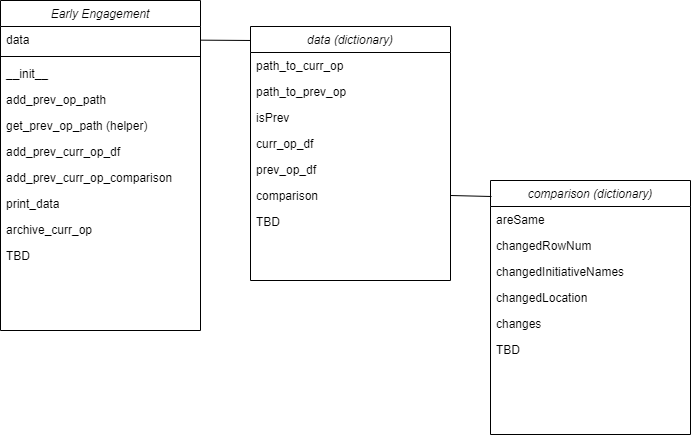
| AIRE flow.png

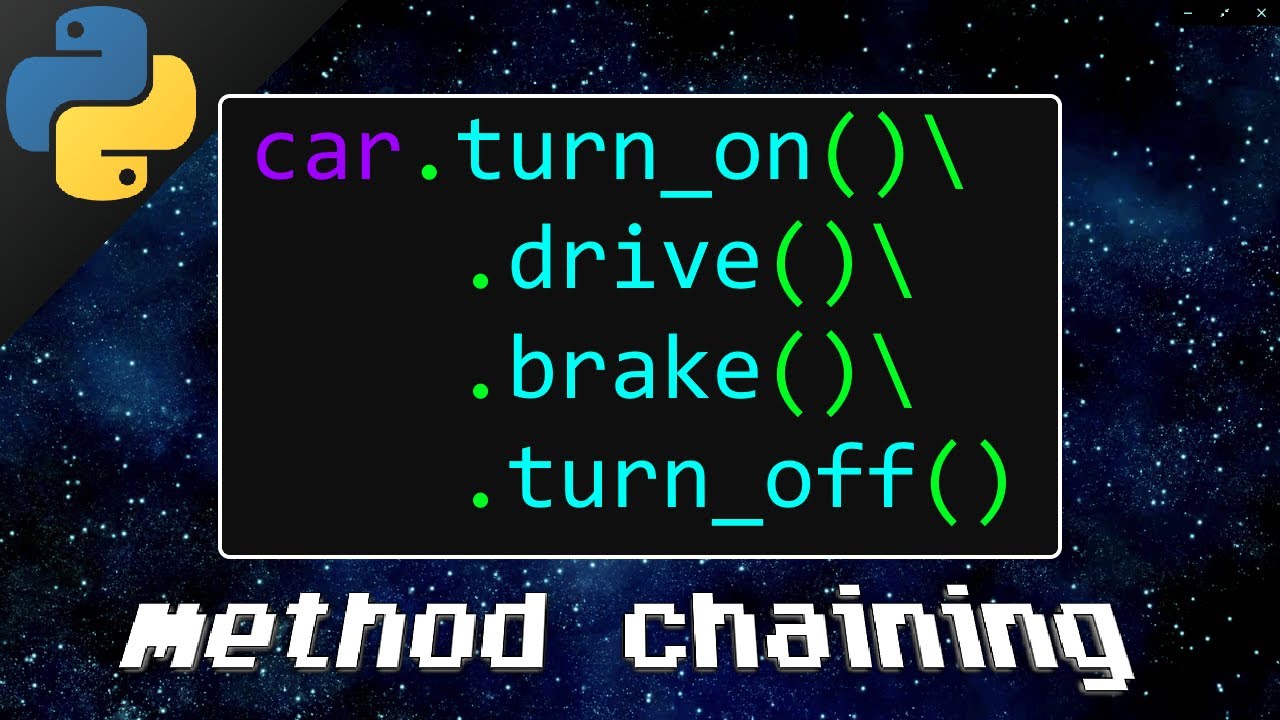
Architecture



Business Layer – early\_engagement







Python Method Chaining

Method chaining is a style of programming in which invoking multiple method calls occurs sequentially. It removes the pain of assigning variables at each intermediate step as each call performs action on same object and then returns the object to next call.

Method chaining has two useful benefits

1. It can reduce the length of the overall code as countless variables do not have to be created.
2. It can increase the readability of the code since methods are invoked sequentially.