Audrey's Prerequisite code.ipynb

The script's goal is to create a conflict matrix for courses in a given semester, identifying prerequisites (under the "and" & "or" parsing) and guaranteeing 0 co-enrollment between a class and its prerequisites. This analysis helps understand course conflicts, prevents simultaneous enrollment in a course and any of its prerequisites, and aids in optimizing scheduling and planning.

Overview of Code (So Far):

1. Data Filtration and Preprocessing:

- **pd.read_excel**: Load and preprocess raw course data from Excel file
  - Future Improvement: Include course data for both fall and spring semesters (so the adjacency matrix accurately reflects all prereqs, as some may only be available in 1 semester)
- **unique_exam_course_ids:** Filters courses by creating a unique set of Course IDs, ensuring only relevant courses are processed.

2. Semester-Specific Data Extraction:

- **requests.get:** Retrieves subject and course data from Cornell's class roster API for term FA24.
- **csv.writer:** Extracts and stores prerequisite/corequisite information in a CSV file (all_subjects_FA24_courses_requisites.csv).

3. Prerequisite Parsing:

- **parse_prerequisites:** Custom regex function that cleans strings by removing prefixes and punctuation, and extracts course codes using regular expressions

4. Dictionary Creation:

- **exam_mapping:** Maps course names to their corresponding course IDs (parsed prerequisites are in subject+catalog → MATH 1920 form, so this function intends to standarize these name to use Course IDs for consistency).
- **prereq_dict:** Constructs an initial mapping from each course ID to its parsed prerequisites (these parsed prerequisites are still in "string" form aka. (subject+catalog) form.)
- **updated_prereq_dict:** Updates the dictionary by matching parsed prerequisites with existing course IDs for consistency (standardizes dictionary by converting parsed prerequisites to course ID form. For ex., the prerequisite MATH 1920 is not 314159 (not

actual ID, just example)). Thus, the final dictionary is consistent, matching course IDs to their prerequisites, which are also in prerequisite form.

5. Adjacency Matrix Construction:

- **np.zeros, np.fill_diagonal:** Builds adjacency matrix representing direct prerequisite relationships and adds self-loops to signify direct course connections.

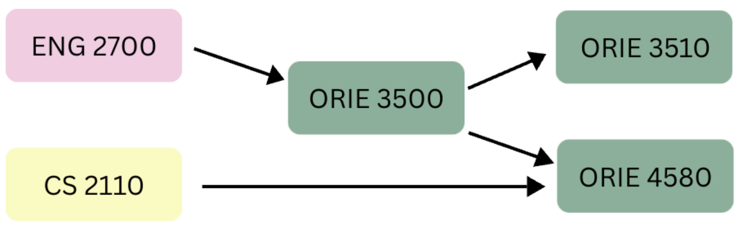
6. Computing Transitive Closure:

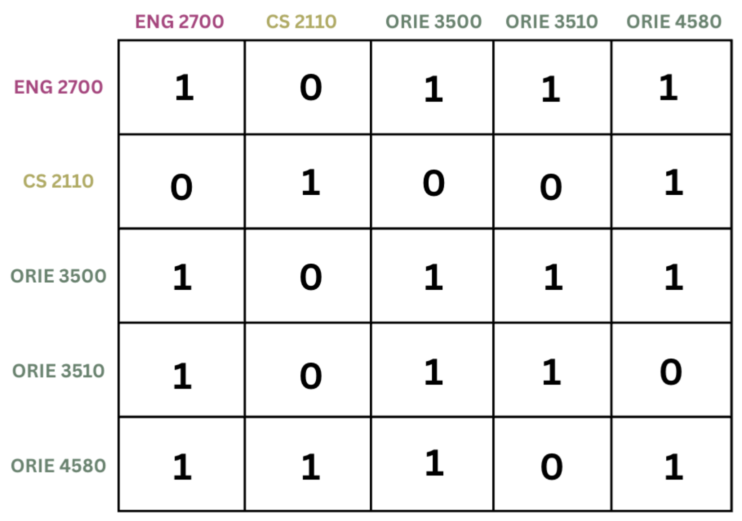
- **Floyd-Warshall algorithm:** Computes a comprehensive reachability matrix indicating indirect prerequisite relationships.

7. Conflict Matrix Generation:

- **conflict_matrix:** Constructs a matrix indicating direct or indirect prerequisite conflicts.
- **conflict_df.to_csv:** Outputs conflict matrix in CSV format (conflict_matrix.csv) for analysis.

Visualization for Poster: Canva Link





	ENG 2700	CS 2110	ORIE 3500	ORIE 3510	ORIE 4580
ENG 2700	1	0	1	1	1
CS 2110	0	1	0	0	1
ORIE 3500	1	0	1	1	1
ORIE 3510	1	0	1	1	0
ORIE 4580	1	1	1	0	1