Student Capacity Forecaster

By Braxton Fair and Cole Harp

Table of contents

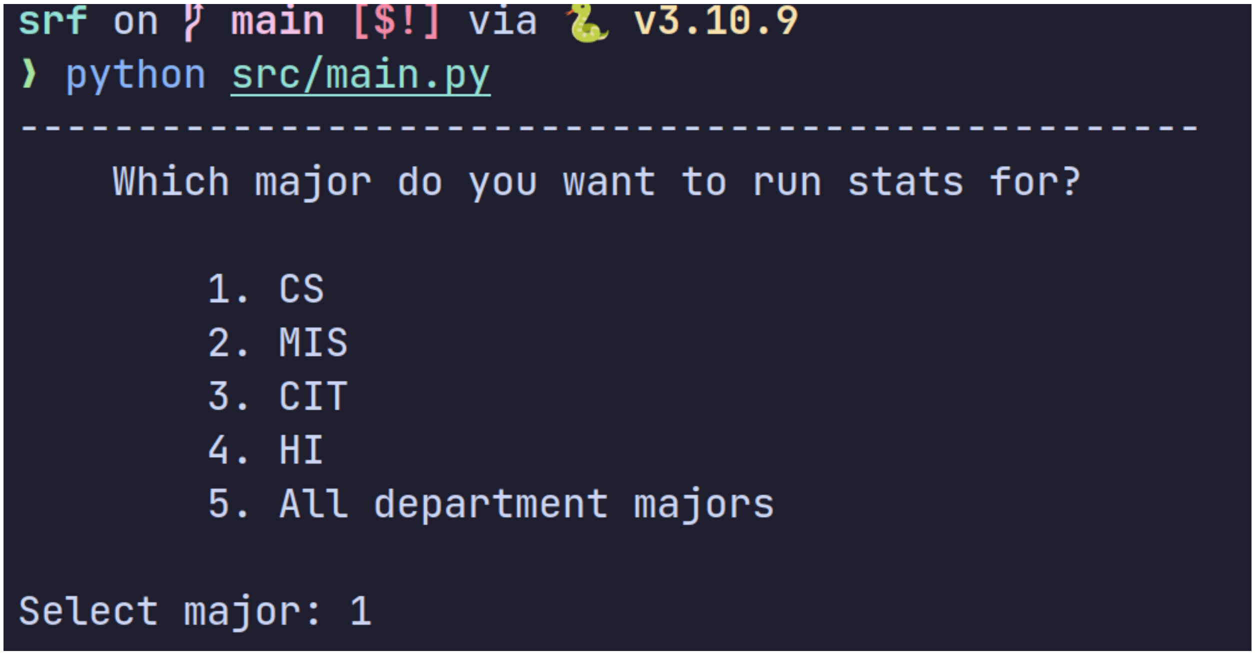
Overview

The Student Retention Forecaster is a proof of concept project with the objective of projecting the movement of student populations within their respective major core classes. This is done by generating test data using relevant and configurable parameters and then generating an intermediate representation in JSON which can then either be used in a capacity projection or outputted to a CSV for viewing.

Dependencies: Currently, we only use pytest as a dependency for some basic testing. Which can be installed either by **pip install -r requirements.txt** or **pip install pytest**.

Example output:

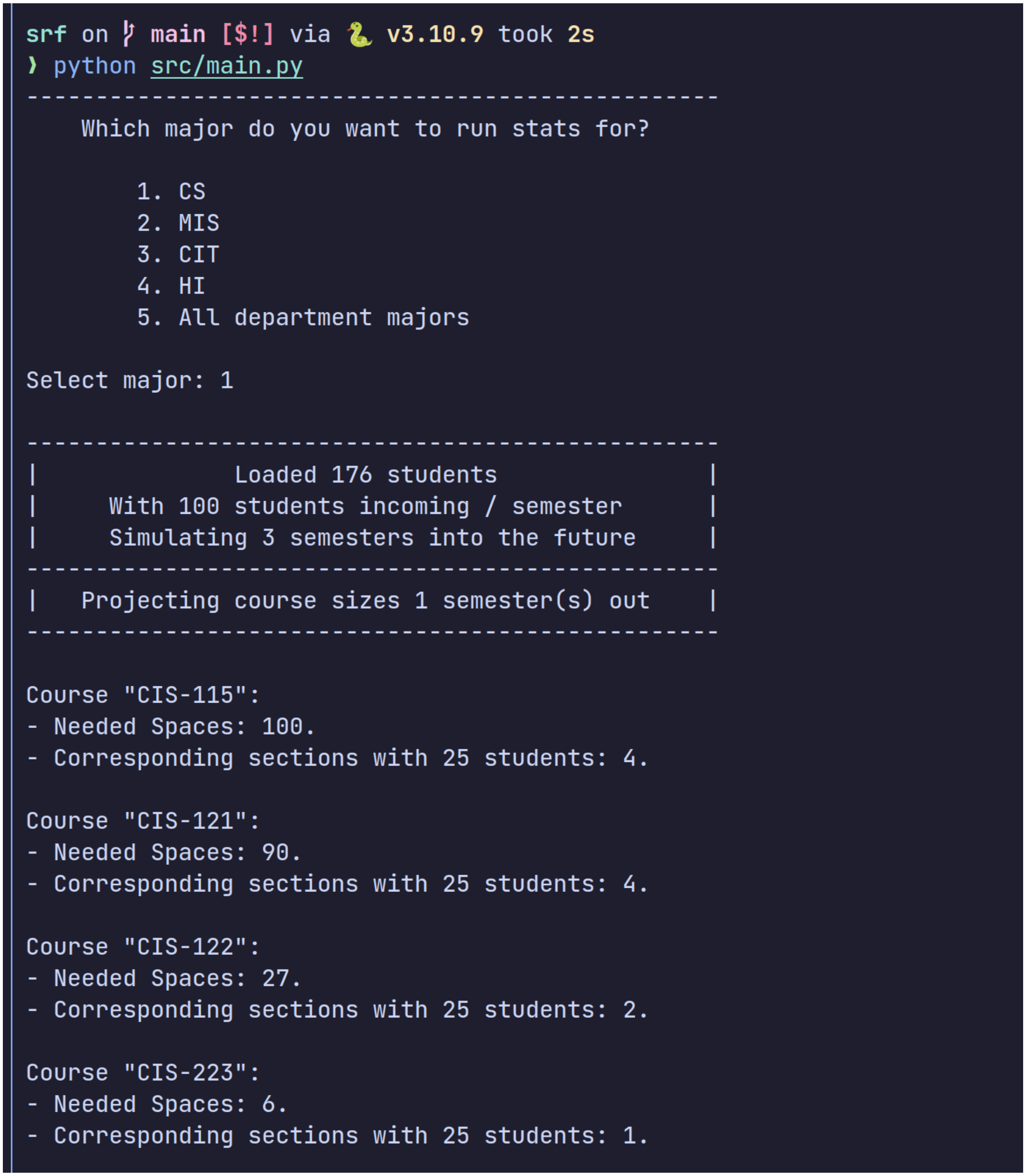
After running **src/main.py**, you'll get a prompt to select a major (or all majors) to evaluate:



```
srf on  main [$!] via 🐍 v3.10.9
❯ python src/main.py
--------------------------------------------------
    Which major do you want to run stats for?

        1. CS
        2. MIS
        3. CIT
        4. HI
        5. All department majors

Select major: 1
```

Upon selecting a major, it will run the forecast for a specified number of semesters (specified in main.py).

Forecast:



```
srf on  main [$!] via 🐍 v3.10.9 took 2s
❯ python src/main.py
--------------------------------------------------
    Which major do you want to run stats for?

        1. CS
        2. MIS
        3. CIT
        4. HI
        5. All department majors

Select major: 1

--------------------------------------------------
|              Loaded 176 students               |
|     With 100 students incoming / semester      |
|     Simulating 3 semesters into the future     |
--------------------------------------------------
|   Projecting course sizes 1 semester(s) out    |
--------------------------------------------------

Course "CIS-115":
- Needed Spaces: 100.
- Corresponding sections with 25 students: 4.

Course "CIS-121":
- Needed Spaces: 90.
- Corresponding sections with 25 students: 4.

Course "CIS-122":
- Needed Spaces: 27.
- Corresponding sections with 25 students: 2.

Course "CIS-223":
- Needed Spaces: 6.
- Corresponding sections with 25 students: 1.
```

The Codebase

Models:

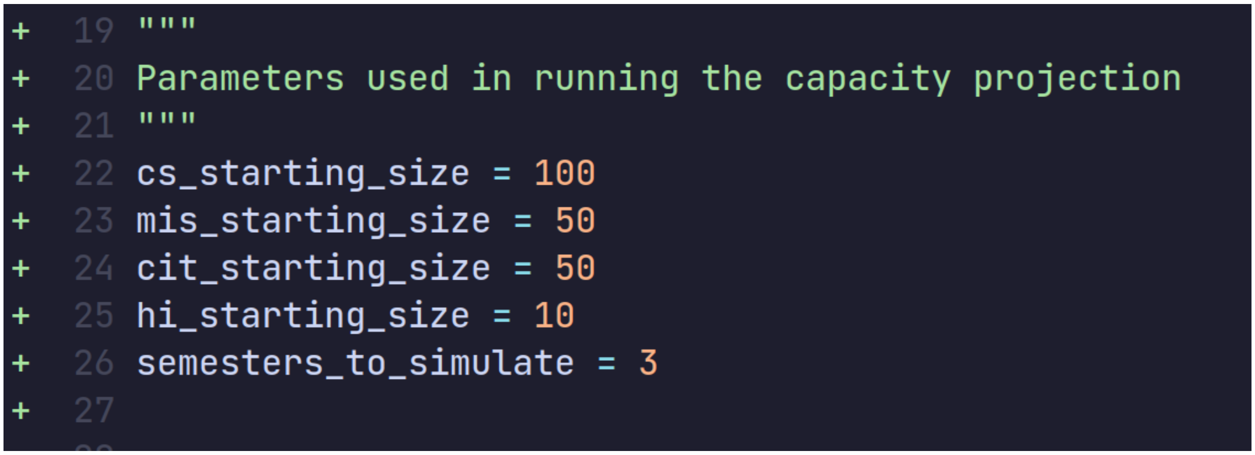
Model	Description
student.py	A model of a student, containing their course history, GPA calculations, DFW rates, and can calculate the highest course taken in a major-specific class.
course.py	A model of a course containing it’s class size, and acts as a doubly-linked-list connecting itself to it’s prerequisites and the courses it is a prerequisite for.
section.py	An unused but modeled section to be taught by a professor within a class. There would be many sections per course.
professor.py	An unused modeled professor. To be used to assign professors to hypothetical sections, but did not have time to complete this work

Other project files

File	Purpose
main.py	The main script that runs the projections
utils.py	A file of helper functions. Currently used to serialize JSON data to student objects and links the class history to it’s respective course objects
student_generator.py	A student data generator file to generate hypothetical student data with a course and grade history. Generates for the CS, CIT, MIS, and HI majors currently. Outputs to **students-(major).json**.
json-to-csv.py	Converts and collapses the **students-(major).json** files into one CSV file
Students-cs.json, students-cit.json, students-mis.json, students-hi.json	The JSON representations of students generated in **student_generator.py**.
compiled_students.csv	The compiled .json student data to a variable-length .csv format

There is also some basic testing inside the **tests** folder, however, due to time constraints, testing was not able to be completed.

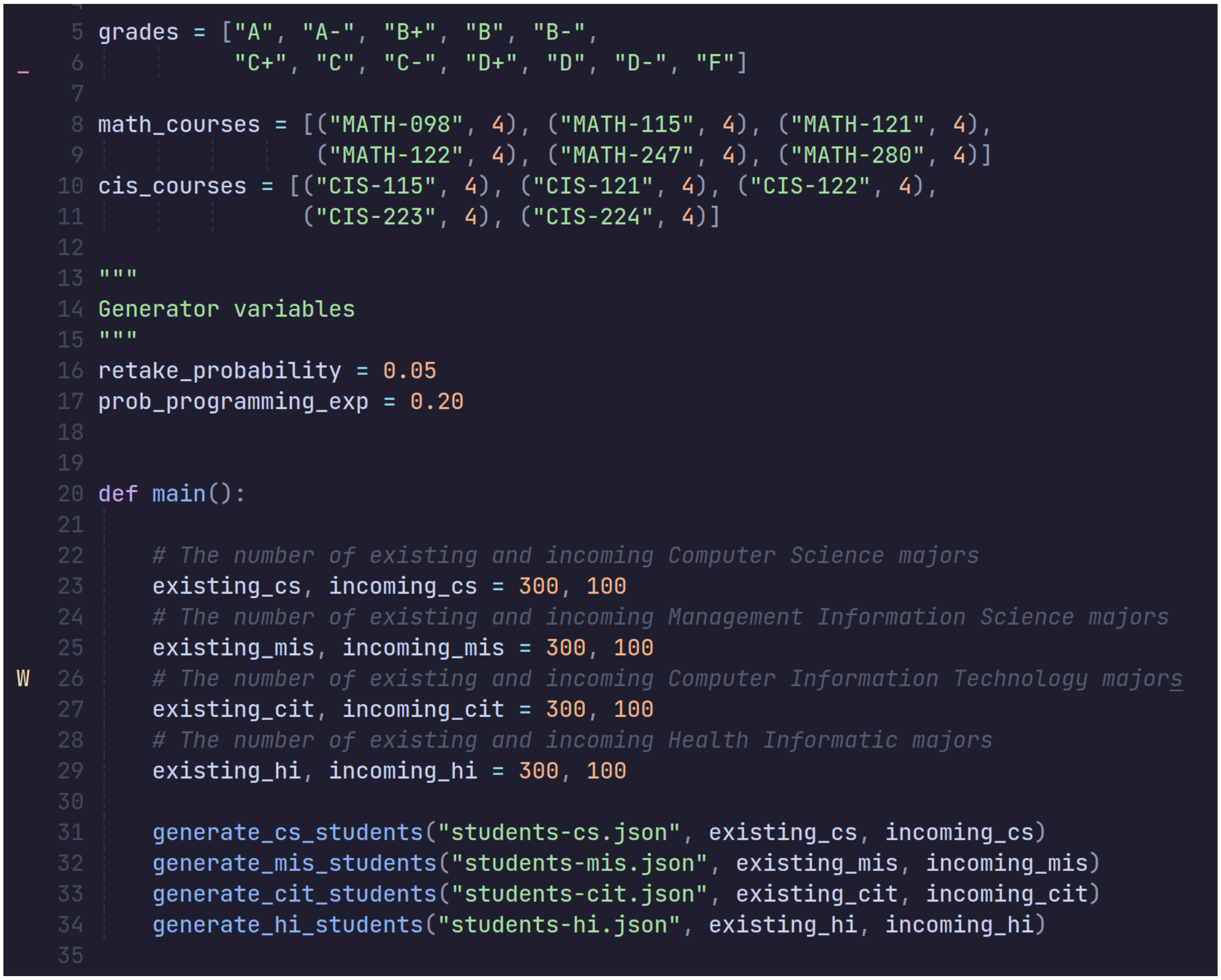
There are parameters in **main.py** that can be utilized in the forecaster, near the top of the file:



```
+  """
+  Parameters used in running the capacity projection
+  """
+  cs_starting_size = 100
+  mis_starting_size = 50
+  cit_starting_size = 50
+  hi_starting_size = 10
+  semesters_to_simulate = 3
+
```

Data Synthesization

As mentioned earlier, data is created within **student_generator.py**, and the output is 4 JSON files, all with ~400 students per major (this total number, down to the number of incoming freshman and existing students can be configurable). You can see the configurable parameters here for data synthesization:



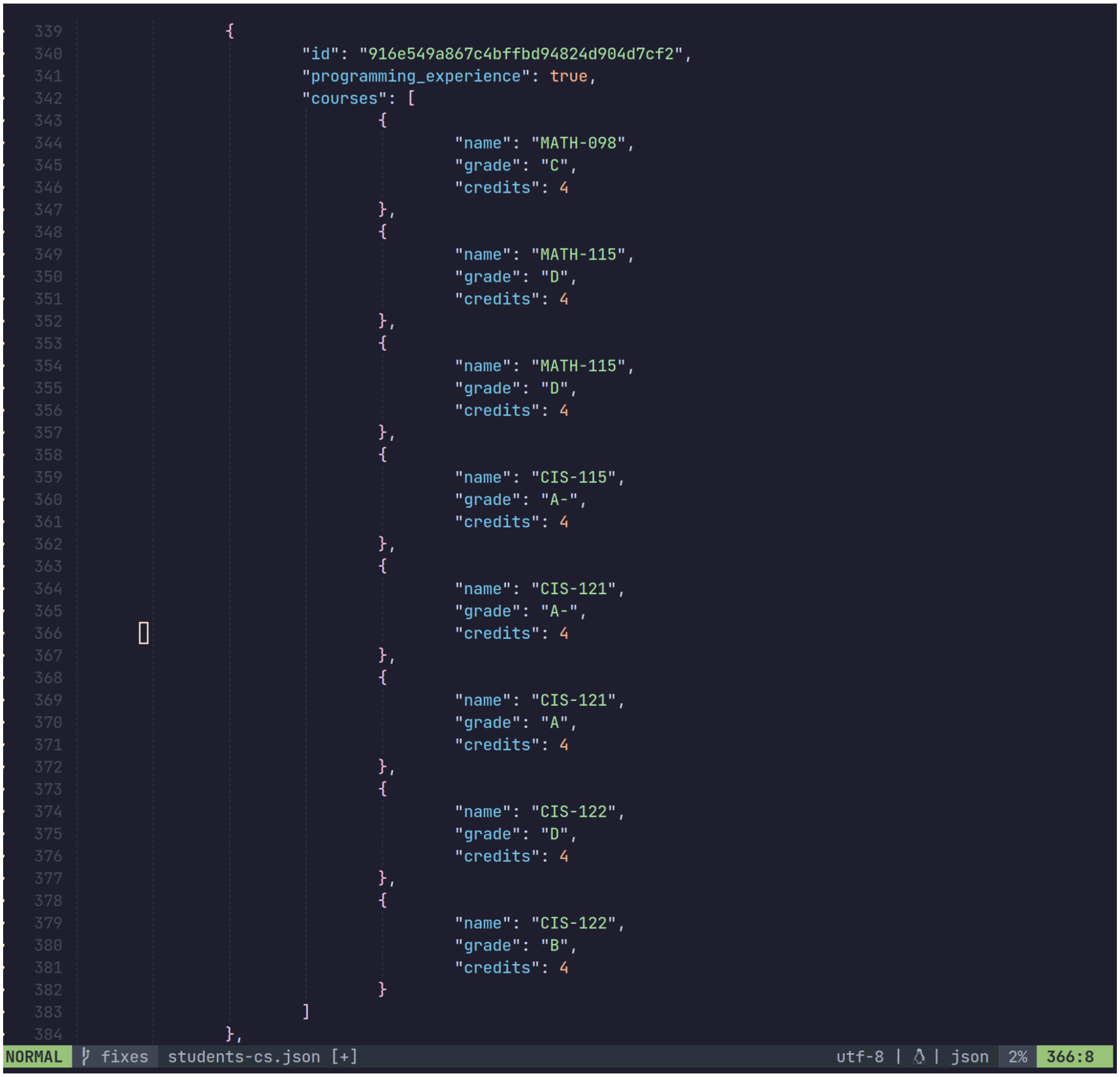
```python
grades = ["A", "A-", "B+", "B", "B-",
          "C+", "C", "C-", "D+", "D", "D-", "F"]

math_courses = [("MATH-098", 4), ("MATH-115", 4), ("MATH-121", 4),
                ("MATH-122", 4), ("MATH-247", 4), ("MATH-280", 4)]
cis_courses = [("CIS-115", 4), ("CIS-121", 4), ("CIS-122", 4),
               ("CIS-223", 4), ("CIS-224", 4)]

"""
Generator variables
"""
retake_probability = 0.05
prob_programming_exp = 0.20


def main():

    # The number of existing and incoming Computer Science majors
    existing_cs, incoming_cs = 300, 100
    # The number of existing and incoming Management Information Science majors
    existing_mis, incoming_mis = 300, 100
    # The number of existing and incoming Computer Information Technology majors
    existing_cit, incoming_cit = 300, 100
    # The number of existing and incoming Health Informatic majors
    existing_hi, incoming_hi = 300, 100

    generate_cs_students("students-cs.json", existing_cs, incoming_cs)
    generate_mis_students("students-mis.json", existing_mis, incoming_mis)
    generate_cit_students("students-cit.json", existing_cit, incoming_cit)
    generate_hi_students("students-hi.json", existing_hi, incoming_hi)

```

This random data incorporates retakes into its generation, and if they get a low enough score on their first go at the class, it'll make the generated student retake the course as well. This gives us just about as close to a real-world data set as we can get currently with our implementation. There may be a few parameters that can be tweaked and added for more like-like data (like giving the chance to retake n times, withdraws, etc..), but with this given data, we are able to produce a good estimate of class sizes/section counts.

Example generated data

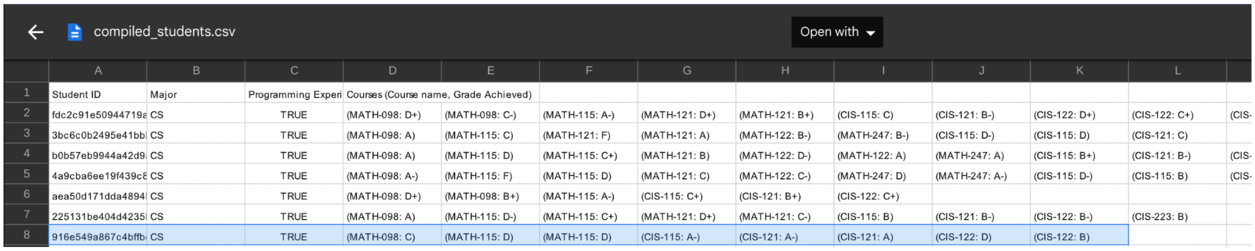
An example student in JSON and CSV format:

JSON



```
                {
                        "id": "916e549a867c4bffbd94824d904d7cf2",
                        "programming_experience": true,
                        "courses": [
                                {
                                        "name": "MATH-098",
                                        "grade": "C",
                                        "credits": 4
                                },
                                {
                                        "name": "MATH-115",
                                        "grade": "D",
                                        "credits": 4
                                },
                                {
                                        "name": "MATH-115",
                                        "grade": "D",
                                        "credits": 4
                                },
                                {
                                        "name": "CIS-115",
                                        "grade": "A-",
                                        "credits": 4
                                },
                                {
                                        "name": "CIS-121",
                                        "grade": "A-",
                                        "credits": 4
                                },
                                {
                                        "name": "CIS-121",
                                        "grade": "A",
                                        "credits": 4
                                },
                                {
                                        "name": "CIS-122",
                                        "grade": "D",
                                        "credits": 4
                                },
                                {
                                        "name": "CIS-122",
                                        "grade": "B",
                                        "credits": 4
                                }
                        ]
                },
NORMAL  fixes  students-cs.json [+]                          utf-8 |  | json  2%  366:8
```

CSV



	A	B	C	D	E	F	G	H	I	J	K	L	
1	Student ID	Major	Programming Experi	Courses (Course name, Grade Achieved)									
2	fdc2c91e50944719a	CS	TRUE	(MATH-098: D+)	(MATH-098: C-)	(MATH-115: A-)	(MATH-121: D+)	(MATH-121: B+)	(CIS-115: C)	(CIS-121: B-)	(CIS-122: D+)	(CIS-122: C+)	(CIS-
3	3bc6c0b2495e41bbl	CS	TRUE	(MATH-098: A)	(MATH-115: C)	(MATH-121: F)	(MATH-121: A)	(MATH-122: B-)	(MATH-247: B-)	(CIS-115: D-)	(CIS-115: D)	(CIS-121: C)	
4	b0b57eb9944a42d9	CS	TRUE	(MATH-098: A)	(MATH-115: D)	(MATH-115: C+)	(MATH-121: B)	(MATH-122: D-)	(MATH-122: A)	(MATH-247: A)	(CIS-115: B+)	(CIS-121: B-)	(CIS-
5	4a9cba6ee19f439c8	CS	TRUE	(MATH-098: A-)	(MATH-115: F)	(MATH-115: D)	(MATH-121: C)	(MATH-122: C-)	(MATH-247: D)	(MATH-247: A-)	(CIS-115: D-)	(CIS-115: B)	(CIS-
6	aea50d171dda4894l	CS	TRUE	(MATH-098: D+)	(MATH-098: B+)	(MATH-115: A-)	(CIS-115: C+)	(CIS-121: B+)	(CIS-122: C+)				
7	225131be404d4235l	CS	TRUE	(MATH-098: A)	(MATH-115: D-)	(MATH-115: C+)	(MATH-121: D+)	(MATH-121: C-)	(CIS-115: B)	(CIS-121: B-)	(CIS-122: B-)	(CIS-223: B)	
8	916e549a867c4bffb	CS	TRUE	(MATH-098: C)	(MATH-115: D)	(MATH-115: D)	(CIS-115: A-)	(CIS-121: A-)	(CIS-121: A)	(CIS-122: D)	(CIS-122: B)		

You can notice they are given a random student ID (Currently the hex value of a UUID (v4), but can be a StarID or TechID. It is then followed by their major to denote differences between the 4 generated majors (CS, CIT, MIS, and HI). Then it contains whether the student has programming experience and their classes in a variable length CSV.

Note: This CSV is not used as an input anywhere. It is purely for output purposes, and the JSON is what is currently used for the input to the projections. The function that translates the JSON to student objects is in **utils.py**, so tweaking that code should allow for taking in CSV as input as well.

Why is this useful?

This proof of concept (PoC) demonstrates that given a subset of student information, and some configurable parameters, you can “simulate” how many sections of a given class is needed for the following semester(s). This code was created with tweaking in mind and should be adaptable to other majors and courses. There is also room for even further specificity, however, during this PoC, we stuck to 4 majors in the CIS department.

Next steps

More in-depth analysis on rates of course size changes through its respective majors’ course sequence.

Currently our courses and math rely on a course having one prerequisite, this can be changed to either a graph structure to better handle the capacity projections through the graph.

Things to consider:

- Population edge cases.
- Semester one to semester two vs semester two to semester one, course size changes; S1 -> S2, S2 -> S1
- Analysis of prior populations to find reputable parameters for data.
    - For example, utilizing student DFW rate in fail rate for a course
- Find out room for error using inconsistencies in data, something like a high vs low projection