# random-student-cli README

The random-student-cli repository contains MATLAB functions that allow me to call on, and keep track of, random students in my class. It allows me to track participation and encourages student engagement. It also removes instructor bias and forces me to call on shy, quiet, or unengaged students.

Furthermore, it is open source so the students can look at the source code and can not accuse me of bias! My intention is for my students to peruse this code. Although there are more powerful and arguably more useful ways to implement these same features, I am teaching courses that emphasize MATLAB and I want to show students that MATLAB can be used for a host of different things. Hopefully this will motivate some of them to use MATLAB, git, and other open source tools in the future.

## Intended use

As the repository name implies, I intended to call these functions from a terminal running `matlab -nodesktop`. This way I can have MATLAB open and call on a student at the same time while preserving my workspace variables for whatever I am currently working on in class.

### Example Usage

1. Open Terminal
2. run `matlab -nodesktop`
3. run `callStudent("mech105")` to call on student
4. depending on answer, type 0 or 1 to grade
5. thats it! To check a students progress you need to figure out thier "number"

## How the set of functions work

The randomStudent collection of functions operates in a very simple manner. The user selects a `courseIdent` when running the `initializeRandomStudent()`. That function creates a .mat file containing 4 vectors. Each student's performance cooresponds to a row in that vector.

The vectors are:

- `blacklist` - initially empty, stores dropped student numbers
- `calls` - a vector that stores how many time a student has been called on
- `names` - a vector that contains the names of all the students in the class

- `numCorrect` - a vector that stores how many times a student that was called on answered "correctly". In my class, any answer is ok as long as it is clear that the student was paying attention and is engaged in learning.

## Initial Setup

If you clone / download the repository, it is reccomended to add the directory you store the functions in to your MATLAB path.

That way it is super easy to call the functions from a `matlab -nodesktop` terminal.

## Semester Setup (Run at beginning of every semester)

These functions will work with any school system but since I am at Colorado State University, it is designed to work fairly easily with AiresWeb.

The following will help you get ready to start the semester using the random-student-cli.

**It is reccomended to wait until the day before (or even the day of) the start of classes to perform these steps.** Students tend to add/drop before classes start. There are ways to mitigate students adding and dropping the class (see `dropStudent` and `addStudent` below).

To setup the program for the semester perform the following steps:

- Open a web browser (preferrably Firefox) go to https://ariesweb.colostate.edu and login using your CSU credentials. **You must be logged into CSU wi-fi or VPN for this link to work**.
- Under "Instructor Tools" click the "Class Lists" link.
- In the window that opens, click the "Class List" next to the class you want to setup.
- You should now see a list of all the students in your class, scroll to the bottom and click the "Save Test (Comma Delimited)" button and save the file somewhere on your computer
- Open the file you just downloaded with your favorite spreadsheet app (for a free open-source alternative to Excel, try LibreOffice!). Copy the student names column (there must be a header "Names") and paste them in a new spreadsheet. Save this spreadsheet as an excel XLS file.
- Finally, you can open MATLAB. If you added the repository to your path, you should be able to type run the `initializeRandomStudent()` function to get started.

## Function Definitions

All that is necessary to run the program is to call the following functions when necessary. Each student is assigned a "student number" which is really just the vector row number that corresponds to their statistics.

### `initializeRandomStudent(csvFileName,courseIdent)`

This function creates the .mat file that contains the workspace variables necessary to run the program. The .mat file will be saved as `courseIdent.mat` in the current working directory so it is reccomended to move to the cloned directory before running.

Inputs:

- `csvFileName` - this is a string that points to a semicolon delimited CSV file with the students names
- `courseIdent` - this is a string that the user provides to specify the course in the future. That way you can have multiple sections of the same course. i.e.) "mech105_2" for MECH105 section 2

There are no Outputs available for this function.

### `callStudent(courseIdent)`

This function allows the instructor to call on a random student. Currently the function will not call on a student again until all students have been called on. Future implementations will change this (don't want students thinking they can check out in big classess).

The function also checks the student number against the blacklist (the list of students that have dropped)

Inputs:

- `courseIdent` - this is the name of the .mat file that the user specified when running `initializeRandomStudent()` for the first time.

There are no Outputs available for this function.

**NOTE: The user can type any number other than 0 or 1 after calling on a student to "skip" them**

### `checkStudent.m`

This function allows the instrcutor to check a students progress, how many times they have been called, and how many of thier answers are correct.

Inputs:

- `courseIdent` - same as before
- `studentNum` - the unique identifier for each student

Outputs (optional):

- `numCalls` - output how many times the student has been called on
- `numCorrect` - output how many times the student has been correct
- `percentage` - `numCorrect / numCalls * 100`

**`dropStudent.m`**

This function allows the instructor to add a student number to the blacklist, so they won't be called on again. Intended to be used for students that drop the course. Removing them from the array would shift everyone's numbers around. Furthermore, it might be useful to "save" the dropped students performance.

Inputs:

- `courseIdent`- a string that cooresponds to the .mat file with data
- `studentNum` - the number that I assign to the student

Outputs (optional):

- `confirmation` - logical 0 or 1 if student was dropped

**`addStudent.m`**

This function allows the instructor to add a new student to the course if necessary. They will always be put at the end of the list of names (will mess up alphabetical order).

Inputs:

- `courseIdent` - same old
- `studentName` - the name of the student being added. Reccomended format *Last Name, First Name.*

Outputs (optional):

- `studentNumber` - the unique identifier for the student in that course.