# Quickstart - how to run:

\*\*\* Current formatting of all 4 spreadsheets in google drive "examgeneration" folder is correct and functional

1. Open questions spreadsheet in google drive; use file > download > tsv
2. Open signups spreadsheet in google drive; use file > download > tsv
3. Download images/ folder
4. Download generateexams.py and Exam.py
5. Download/customize config file as per instructions in next section.
6. Run generateexams.py (it looks for a command-line argument with the name of the config file, but if it doesn't find one will ask you for input). This creates .tex files.
7. Open generated .tex sources compile into pdfs using your favourite LaTeX editor (or you could uncomment the pdf generation lines in the python script, but be aware that these don't handle compilation errors at all gracefully and you could end up with both latex and python processes hanging).

## Config file formatting:

* Can be named whatever you like.
* Properties can go in any order (ie, entire lines can switch spots).
* Required properties:
  + questions
  + signups
* Optional properties:
  + exam type (default = final)
  + student groups (default = none / separate ids with comma / separate groups with semicolon / spaces are ok)
  + random seed (default = wugz)
* Property keys and values must be typed as shown in this example:

questions: pathtoquestionsfile\_dontusespaces.tsv

signups: pathtosignupsfile/couldbenested.tsv

exam type: midterm

student groups: 1234,3456;8846,2345,1220

random seed: whateveryoulike could have spaces

## Current midterm exam characteristics:

* 5 questions total: one each from topics transcription, articulatory phonetics, skewed distributions, phonological relationships, and "wild" (wildcard question randomly selected from transcription, articulatory phonetics, phonological relationships, or other).
* Difficulty distribution is either (1x very hard, 5x medium), or (1x very hard, 1x hard, 2x medium, 1x easy).
* Because of the fact that there is a not a full crossing of difficulty & topic, the choosing of questions for midterms is done in an annoyingly hard-coded way. If the question coverage were to improve, we could use the more elegant (and more random) algorithm that's used for the final exams.

## Current final exam characteristics:

* 6 questions total: one each from topics acoustics, alternations, phonological features, syllables, tone, and dataset.
* Difficulty distribution is (2, easy, 2x medium, 1x hard, and 1x very hard). Very hard always comes from the dataset questions.
* The non-dataset topics fully cross the difficulty levels easy/medium/hard, so there's lots of opportunity for randomness in terms of which topics get which difficulties, etc.

## Shared characteristics:

* Any students who are grouped together in the config file (eg, who worked together on assignments) will not have any overlap in exam questions (TODO though could have overlap in sources, just with different actual data).
* No individual student's exam will have more than one question with the same question type (eg morphology, signlanguage, etc), based on optional entries in the "QuestionType" column of the questions spreadsheet.

## LaTeX compiling:

* In order to make it easy to use verbatim input of ipa characters, I am using font packages that require compilation with xelatex.
* pdflatex will not work.

## Student groups:

* Any students listed in a group together (see config) will \*not\* have overlap in their exam questions, but could have overlap in their question sources. For example, the following is currently possible within a group:
  + From source "Day 2 Handout, Question 6" student A gets "Provide the IPA transcription for the word 'grilled'."
  + From source "Day 2 Handout, Question 6" student B gets "Provide the IPA transcription for the word 'cheese'."

## Question subtypes:

* There is a column in the questions spreadsheet called QuestionType. If a value is entered in this column (e.g. "UR" or "signlanguage"), then no student will get more than one of that question type.
* \*\*\* Note! I have combined the signlanguage column with the subtype column. Hopefully that doesn't cause any problems.

## Image sizes:

* If images are to be placed horizontally next to each other (as per ImageArrangement column in spreadsheet), they will be auto-scaled so that each takes up half the page.
* If images are placed vertically (default), they are not scaled at all. This means that you should aim for max widths of about 900px.

## Formatting data for LaTeX:

* Any of the spreadsheet values that will be displayed as text in the exam document must be LaTeX-aware.
  + Relevant columns: source, instructions, data1&2, image captions, instructor comments.
* What do I mean by LaTeX-aware? ...
  + Lists should be created using \begin{itemize} \item text 1 \item text 2 \end{itemize} (or enumerate instead of itemize)
  + If you want to see curly braces { } appear in the doc, they need to be escaped \{ \}
  + The font packages I'm using don't permit italics, so use \ul{text to underline} to emphasize text.
  + If you want to use quotes, make sure to use `` '' or ` '
  + Non-IPA subscripts must be entered using $\_{text}$; superscripts get $^{text}$
  + Ampersands and underscores must be escaped: \& \\_
  + Exception: square brackets are assumed to be around transcriptions; they will be auto-escaped by the python script. This means you can't use them to pass any arguments to commands (eg \item[a.] )
  + Use \\ for line breaks.
  + I hope I got them all...
* On the bright side: you get to copy/paste IPA symbols instead of having to code them. Yay!