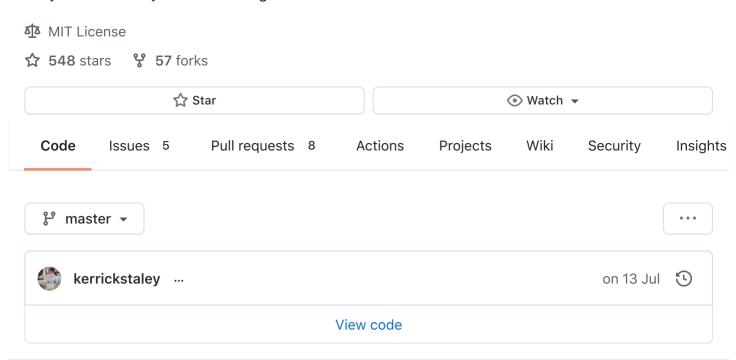
Registraley / genanki

A Python 3 Library for Generating Anki Decks



README.md

genanki: A Library for Generating Anki Decks

genanki allows you to programatically generate decks in Python 3 for Anki, a popular spaced-repetition flashcard program. Please see below for concepts and usage.

This library and its author(s) are not affiliated/associated with the main Anki project in any way.

build passing

Notes

The basic unit in Anki is the Note, which contains a fact to memorize. Note s correspond to one or more Card s.

Here's how you create a Note:

```
my_note = genanki.Note(
  model=my_model,
  fields=['Capital of Argentina', 'Buenos Aires'])
```

You pass in a Model, discussed below, and a set of fields.

Models

A Model defines the fields and cards for a type of Note. For example:

This note-type has two fields and one card. The card displays the Question field on the front and the Question and Answer fields on the back, separated by a <hr> . You can also pass a css argument to Model() to supply custom CSS.

You need to pass a model_id so that Anki can keep track of your model. It's important that you use a unique model_id for each Model you define. Use random.randrange(1 << 30, 1 << 31) to generate a suitable model_id, and hardcode it into your Model definition.

Generating a Deck/Package

To import your notes into Anki, you need to add them to a Deck:

```
my_deck = genanki.Deck(
  2059400110,
  'Country Capitals')
my_deck.add_note(my_note)
```

Once again, you need a unique deck_id that you should generate once and then hardcode into your .py file.

Then, create a Package for your Deck and write it to a file:

```
genanki.Package(my_deck).write_to_file('output.apkg')
```

You can then load output.apkg into Anki using File -> Import...

Media Files

To add sounds or images, set the media_files attribute on your Package:

```
my_package = genanki.Package(my_deck)
my_package.media_files = ['sound.mp3', 'images/image.jpg']
```

media_files should have the path (relative or absolute) to each file. To use them in notes, first add a field to your model, and reference that field in your template:

```
my model = genanki.Model(
  1091735104,
  'Simple Model with Media',
  fields=[
    {'name': 'Question'},
    {'name': 'Answer'},
    {'name': 'MyMedia'},
                                                            # ADD THIS
  ],
  templates=[
    {
      'name': 'Card 1',
      'qfmt': '{{Question}}<br>{{MyMedia}}',
                                                            # AND THIS
      'afmt': '{{FrontSide}}<hr id="answer">{{Answer}}',
    },
  1)
```

Then, set the MyMedia field on your card to [sound:sound.mp3] for audio and for images.

You cannot put in the template and image.jpg in the field. See these sections in the Anki manual for more information: Importing Media and Media & LaTeX References.

You should only put the filename (aka basename) and not the full path in the field; will not work. Media files should have unique filenames.

Note GUIDs

Note s have a guid property that uniquely identifies the note. If you import a new note that has the same GUID as an existing note, the new note will overwrite the old one (as long as their models have the same fields).

This is an important feature if you want to be able to tweak the design/content of your notes, regenerate your deck, and import the updated version into Anki. Your notes need to have stable GUIDs in order for the new note to replace the existing one.

By default, the GUID is a hash of all the field values. This may not be desirable if, for example, you add a new field with additional info that doesn't change the identity of the note. You can create a custom GUID implementation to hash only the fields that identify the note:

```
class MyNote(genanki.Note):
    @property
    def guid(self):
        return genanki.guid_for(self.fields[0], self.fields[1])
```

sort_field

Anki has a value for each Note called the <code>sort_field</code>. Anki uses this value to sort the cards in the Browse interface. Anki also is happier if you avoid having two notes with the same <code>sort_field</code>, although this isn't strictly necessary. By default, the <code>sort_field</code> is the first field, but you can change it by passing <code>sort_field= to Note()</code> or implementing <code>sort_field</code> as a property in a subclass (similar to <code>guid</code>).

YAML for Templates (and Fields)

You can create your template definitions in the YAML format and pass them as a str to Model(). You can also do this for fields.

Using genanki inside an Anki addon

genanki supports adding generated notes to the local collection when running inside an Anki 2.1 addon (Anki 2.0 may work but has not been tested). See the

```
.write_to_collection_from_addon() method.
```

Publishing to PyPl

If your name is Kerrick, you can publish the genanki package to PyPI by running these commands from the root of the genanki repo:

```
rm -rf dist/*
python3 setup.py sdist bdist_wheel
python3 -m twine upload dist/*
```

Note that this directly uploads to prod PyPI and skips uploading to test PyPI.

D۵	leas	
κ	1626	

🗘 1 tags

Packages

No packages published

Contributors 4



kerrickstaley Kerrick Staley



sciencemanx Adam Van Prooyen



holocronweaver Jesse Johnson



dvklopfenstein DV Klopfenstein

Languages

• Python 96.8%

Shell 2.4%

• Makefile 0.8%