wanikani$_{api}$Documentation

Release __version__ = '0.5.1'

Gary Grant Graham

Aug 18, 2020

Contents:

CHAPTER 1

wanikani_api

An API wrapper for Wanikani (V2)

- Free software: BSD license
- Documentation: https://wanikani-api.readthedocs.io.

1.1 Features

- Easy access to Wanikani resources associated to your account.
- Automatic handling of pagination.
- Automatic fetching of related Subjects

1.2 Quickstart

```
>>> from wanikani_api.client import Client
>>> v2_api_key = "drop_your_v2_api_key_in_here" # You can get it here: https://www.
↪wanikani.com/settings/account
>>> client = Client(v2_api_key)
>>> user_information = client.user_information()
>>> print(user_information)
UserInformation{ username:Tadgh11, level:8, max_level_granted_by_subscription:60,␣
↪profile_url:https://www.wanikani.com/users/Tadgh11 started_at:2013-07-09 12:02:54.
↪952786+00:00, subscribed:True, current_vacation_started_at:None }
>>> all_vocabulary = client.subjects(types="vocabulary")
```

(continues on next page)

(continued from previous page)

```
>>> for vocab in all_vocabulary:
>>>     print(vocab.meanings[0].meaning) #Vocabulary may have multiple meanings, we␣
↪just grab the first in the list.
One
One Thing
Seven
Seven Things
Nine
Nine Things
Two
...
```

1.3 TODO

- Make use of ETags for caching
- simplify API
- Improve automatic prefetching of subjects when relevant.

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install wanikani_api, run this command in your terminal:

```
$ pip install wanikani_api
```

This is the preferred method to install wanikani_api, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for wanikani_api can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/Kaniwani/wanikani_api
```

Or download the tarball:

```
$ curl  -OL https://github.com/Kaniwani/wanikani_api/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

3.1 Getting a client

```
>>> import wanikani_api.client as client
>>> wk_api = client.Client("enter your V2 API key here")
```

3.2 User Information

```
>>> import wanikani_api.client as client
>>> wk_api = client.Client("enter your V2 API key here")
>>> user_info = wk_api.user_information()
>>> user.username
"Tadgh"
```

3.3 Subjects

This is how to retrieve all Subjects in Wanikani. Subjects are either *models.Kanji*, *models.Radical*, or *models.Vocabulary*.

```
>>> vocabulary = wk_api.subjects(types="vocabulary")
>>> for vocab in vocabulary:
>>>    print(vocab.readings[0].reading)
""
""
""
""
""
""
```

(continues on next page)

(continued from previous page)

```
...
>>> print(len(vocabulary))
1000
```

Note that by default the client will only retrieve the first Page of results. This can be changed by passing `fetch_all=True` to any client function which returns multiple results. Like so:

```
>>> vocabulary = wk_api.subjects(types="vocabulary", fetch_all=True)
>>> print(len(vocabulary))
6301
```

Alternatively, if you decide afterwards you'd like to fill in the missing data, you can do this:

```
>>> vocabulary = wk_api.subjects(types="vocabulary")
>>> print(len(vocabulary))
1000
>>> vocabulary.fetch_all_pages()
>>> print(len(vocabulary))
6301
```

You are also free to fetch one page at a time. Note also that you can access indiviual `models.Page` objects if you like.

```
>>> vocabulary = wk_api.subjects(types="vocabulary")
>>> print(len(vocabulary))
1000
>>> vocabulary.fetch_next_page()
>>> print(len(vocabulary))
2000
>>> print(len(vocabulary.pages))
2
# Iterate only over elements in the second page:
>>> for vocab in vocabulary.pages[1]:
>>>     print(vocab.parts_of_speech)
['noun', 'suru_verb']
['noun']
['intransitive_verb', 'godan_verb']
```

This works for any client function that is *plural*, e.g. assignments(), subjects(), reviews(), etc.

By default, the Wanikani API returns only subject IDs when referring to a subject. Therefore, for any resource which contains a field *subject_id* or *subject_ids* can make use of convenient properties *subject* and *subjects*, respectively. This allows you to quickly grab related subjects without making a separate explicit call to the subjects endpoint. See below.

3.4 Assignments

```
>>> assignments = wk_api.assignments(subject_types="vocabulary")
>>> for assignment in assignments:
>>>     print(assignment.srs_stage)
>>>     print(assignment.subject.meaning) # The client will automatically go and fetch␣
↪this subject for you.
9
"One"
```

(continues on next page)

(continued from previous page)

```
9
"One Thing"
```

Note that the above will make a new API call every time you call `subject` on a new assignment.

3.5 Review Statistics

Here's how to get your review statistics for your level 30 vocabulary and kanji (but not radicals), that you have gotten correct at most 50%

```
>>> subjects = wk_api.subjects(types=["vocabulary", "kanji"], level=30)
>>> stats = wk_api.review_statistics(subject_ids=[subject.id for subject in subjects],
↪ percentages_less_than=50)
>>> for stat in stats:
>>>     print(stat.percentage_correct)
44
42
49
31
```

3.6 Study Materials

Here's how to get all study materials for any vocabulary that have the slug. The *slug* is a simple identifier on the wanikani site (like this: https://www.wanikani.com/vocabulary/)

```
>>> subjects = wk_api.subjects(slugs="", types="vocabulary")
>>> study_mats = wk_api.study_materials(subject_ids=[subject.id for subject in␣
↪subjects])
>>> for study_material in study_mats:
>>> print (", ".join(study_material.meaning_synonyms)
"wool,yarn"
```

CHAPTER 4

Client Module

class wanikani_api.client.**Client**(*v2_api_key*, *subject_cache_enabled=False*)

This is the only object you can instantiate. It provides access to each relevant API endpoint on Wanikani.

assignment(*assignment_id*)

Get a single *models.Assignment* by its known id

Parameters **assignment_id** – the id of the assignment

Returns a single *models.Assignment*

assignments(*ids=None*, *created_at=None*, *subject_ids=None*, *subject_types=None*, *levels=None*, *available_before=None*, *available_after=None*, *srs_stages=None*, *unlocked=None*, *started=None*, *passed=None*, *burned=None*, *resurrected=None*, *hidden=None*, *updated_after=None*, *fetch_all=False*)

Assignments are the association between a user, and a subject. This means that every time something is added to your lessons, a new *models.Assignment* is created.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **ids** (*int[]*) – Return only results with the given IDs
- **created_at** – Timestamp when resource was created
- **subject_ids** (*int[]*) – Return only :class:‘.models.Assignment‘s which are tied to the given subject_ids
- **subject_types** (*str[]*) – The specific *models.Subject* types you wish to retrieve. Possible values are: `["kanji", "vocabulary", "radicals"]`
- **levels** (*int[]*) – Include only *models.Assignment* where the subjects are from the specified levels.
- **available_before** (*datetime*) – Return assignment reviews available before timestamp

- **available_after** (*datetime*) – Return assignment reviews available after timestamp
- **srs_stages** (*int*) – Return assignments of specified srs stages. Note, 0 is lessons, 9 is the burned state
- **unlocked** (*bool*) – Return assignments which have unlocked (made available to lessons)
- **started** (*bool*) – Return assignments which move from lessons to reviews
- **passed** (*bool*) – Return assignments which have reach Guru (aka srs_stage 5) at some point (true) or which have never been Guru'd (false)
- **burned** (*bool*) – Return assignments which have been burned at some point (true) or never have been burned (false)
- **resurrected** (*bool*) – Return assignments which either have been resurrect (true) or not (false)
- **hidden** (*bool*) – Return assignments which are or are not hidden from the user-facing application
- **updated_after** (*datetime*) – Return results which have been updated after the timestamp

Returns An iterator over a set of `models.Page` where the data contained is all *models.Assignment*

level_progression(*level_progression_id*)
Get a single `models.LevelProgression` by its known id

Parameters **level_progression_id** – the id of the level_progression

Returns a single `models.LevelProgression`

level_progressions(*ids=None*, *updated_after=None*, *fetch_all=False*)
Retrieve all `models.LevelProgression` for a given user.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **ids** (*int[]*) – Return only results with the given IDs
- **updated_after** (*datetime*) – Return results which have been updated after the timestamp

Returns An iterator over all `models.LevelProgression` for a given user.

reset(*reset_id*)
Get a single *models.Reset* by its known id

Parameters **reset_id** – the id of the reset

Returns a single *models.Reset*

resets(*ids=None*, *updated_after=None*, *fetch_all=False*)
Retrieve information for all resets the user has performed on Wanikani.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.

- **ids** (*int[]*) – Return only results with the given IDs
- **updated_after** (*datetime*) – Return results which have been updated after the timestamp

Returns An iterator over all *models.Reset* for a given user.

review(*review_id*)
Get a single *models.Review* by its known id

Parameters **review_id** – the id of the review

Returns a single *models.Review*

review_statistic(*review_statistic_id*)
Get a single *models.ReviewStatistic* by its known id

Parameters **review_statistic_id** – the id of the review_statistic

Returns a single *models.ReviewStatistic*

review_statistics(*ids=None*, *subject_ids=None*, *subject_types=None*, *updated_after=None*, *percentages_greater_than=None*, *percentages_less_than=None*, *hidden=None*, *fetch_all=False*)
Retrieve all Review Statistics from Wanikani. A Review Statistic is related to a single subject which the user has studied.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **ids** (*int[]*) – Return only results with the given IDs
- **subject_ids** (*int[]*) – Return only :class:‘.models.Assignment‘s which are tied to the given subject_ids
- **subject_types** (*str[]*) – The specific *models.Subject* types you wish to retrieve. Possible values are: `["kanji", "vocabulary", "radicals"]`
- **updated_after** (*datetime*) – Return results which have been updated after the timestamp
- **percentages_greater_than** (*int*) – Return results where the percentage_correct is greater than the value. [0-100]
- **percentages_less_than** (*int*) – Return results where the percentage_correct is less than the value. [0-100]
- **hidden** (*bool*) – Return only results where the related subject has been hidden.

Returns An iterator which contains all Review Statistics

reviews(*ids=None*, *subject_ids=None*, *updated_after=None*, *fetch_all=False*)
Retrieve all reviews for a given user. A *models.Review* is a single instance of this user getting a single review correctly submitted.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **ids** (*int[]*) – Return only results with the given IDs
- **subject_ids** (*int[]*) – Return only :class:‘.models.Assignment‘s which are tied to the given subject_ids

- **updated_after** (*datetime*) – Return results which have been updated after the timestamp

Returns An iterator over all *models.Review* for a given user.

study_material (*study_material_id*)
Get a single *models.StudyMaterial* by its known id

Parameters **study_material_id** – the id of the study material

Returns a single *models.StudyMaterial*

study_materials (*ids=None*, *subject_ids=None*, *subject_types=None*, *hidden=None*, *updated_after=None*, *fetch_all=False*)
Retrieve all Study Materials. These are primarily meaning notes, reading notes, and meaning synonyms.

Parameters

- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **ids** (*int[]*) – Return only results with the given IDs
- **subject_ids** (*int[]*) – Return only :class:‘.models.Assignment‘s which are tied to the given subject_ids
- **subject_types** (*str[]*) – The specific *models.Subject* types you wish to retrieve. Possible values are: `["kanji", "vocabulary", "radicals"]`
- **hidden** (*bool*) – Return only results where the related subject has been hidden.
- **updated_after** (*datetime*) – Return results which have been updated after the timestamp

Returns An iterator over all Study Materials

subject (*subject_id*)
Get a single subject by its known id

Parameters **subject_id** – the id of the subject

Returns a single *models.Subject*. This might be either: * *models.Radical* * *models.Kanji* * *models.Vocabulary*

subjects (*ids=None*, *types=None*, *slugs=None*, *levels=None*, *hidden=None*, *updated_after=None*, *fetch_all=False*)
Retrieves Subjects

Wanikani refers to Radicals, Kanji, and Vocabulary as Subjects. This function allows you to fetch all of the subjects, regardless of the current level of the account that the API key is associated to. All parameters to this function are optional, and are for filtering the results. are ignored, and the subject with that ID in question is fetched.

Parameters

- **ids** (*int[]*) – Filters based on a list of IDs. Does not cause other parameters to be ignored.
- **types** (*str[]*) – The specific *models.Subject* types you wish to retrieve. Possible values are: `["kanji", "vocabulary", "radicals"]`
- **slugs** (*str[]*) – The wanikani slug
- **levels** (*int[]*) – Include only *models.Subject* from the specified levels.

- **hidden** (*bool*) – Return *models.Subject* which are or are not hidden from the user-facing application
- **fetch_all** (*bool*) – if set to True, instead of fetching only first page of results, will fetch them all.
- **updated_after** (`datetime.datetime`) – Return results which have been updated after the timestamp

Returns

An iterator over multiple `models.Page` , in which the `data` field contains a list anything that is a *models.Subject*, e.g.:

- *models.Radical*
- *models.Kanji*
- *models.Vocabulary*

summary()

Returns

user_information()

Gets all relevant information about the user.

Raises `exceptions.InvalidWanikaniApiKeyException`

Return type *models.UserInformation*

CHAPTER 5

Datatypes

class wanikani_api.models.**Assignment**(*json_data*, **args*, ***kwargs*)
Simple class holding information about Assignmetns.

class wanikani_api.models.**AuxiliaryMeaning**(*auxiliary_meaning_json*)
Simple data class for handling auxiliary meanings

class wanikani_api.models.**Kanji**(*json_data*, **args*, ***kwargs*)
A model for the Kanji Resource

amalgamation_subject_ids = None
A list of IDs for the related *models.Vocabulary* which this Kanji is a component in.

component_subject_ids = None
A list of IDs for the related *models.Radical* which combine to make this kanji

readings = None
A list of *models.Reading* related to this Vocabulary.

class wanikani_api.models.**Meaning**(*meaning_json*)
Simple class holding information about a given meaning of a vocabulary/Kanji

accepted_answer = None
Whether or not this answer is accepted during reviews in Wanikani.

meaning = None
The english meaning of a Subject.

primary = None
Wether or not the meaning is considered to be the main one.

class wanikani_api.models.**Radical**(*json_data*, **args*, ***kwargs*)
A model for the Radical object.

amalgamation_subject_ids = None
IDs for various other *models.Subject* for which this radical is a component.

character_images = None
A list of dictionaries, each containing a bunch of information related to a single character image.

class wanikani_api.models.**Reading**(*meaning_json*)
Simple class holding information about a given reading of a vocabulary/kanji

accepted_answer = None
Whether this answer is accepted as correct by Wanikani during review.

primary = None
Whether this is the primary reading.

reading = None
the actual for the reading.

class wanikani_api.models.**Reset**(*json_data*, **args*, ***kwargs*)
Simple model holding resource information

class wanikani_api.models.**Review**(*json_data*, **args*, ***kwargs*)

class wanikani_api.models.**ReviewStatistic**(*json_data*, **args*, ***kwargs*)
Simple model holding ReviewStatistic Information

class wanikani_api.models.**StudyMaterial**(*json_data*, **args*, ***kwargs*)
Simple model holding information about Study Materials

class wanikani_api.models.**Subject**(*json_data*, **args*, ***kwargs*)
This is the base Subject for Wanikani. This contains information common to Kanji, Vocabulary, and Radicals.

characters = None
The actual japanese kanji/radical symbol such as

created_at = None
The date at which the Subject was created originally on Wanikani.

document_url = None
The direct URL where the subject can be found on Wanikani

hidden_at = None
When Wanikani removes a subject, they seem to instead set it to hidden, for backwards compatibilty with clients.

level = None
The level of the subject.

meanings = None
A list of *models.Meaning* for this subject.

class wanikani_api.models.**Subjectable**(**args*, ***kwargs*)
A Mixin allowing a model to quickly fetch related subjects.

Any resource which inherits Subjectable must have either *subject_id* or *subject_ids* as an attribute.

class wanikani_api.models.**UpcomingReview**(*json_data*, **args*, ***kwargs*)

class wanikani_api.models.**UserInformation**(*json_data*, **args*, ***kwargs*)
This is a simple container for information returned from the `/user/` endpoint. This is all information related to the user.

current_vacation_started_at = None
datetime at which vacation was enabled on wanikani.

level = None
current wanikani level

profile_url = None
Link to user's profile.

started_at = None
datetime at which the user signed up.

subscription = None
maximum level granted by subscription.

username = None
username

class wanikani_api.models.**Vocabulary**(*json_data*, **args*, ***kwargs*)
A model for the Vocabulary Resource

component_subject_ids = None
List of IDs for :class".*models.Kanji* which make up this vocabulary.

parts_of_speech = None
A list of strings, each of which is a part of speech.

readings = None
A list of *models.Reading* related to this Vocabulary.

CHAPTER 6

Exceptions

members

CHAPTER 7

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/kaniwani/wanikani_api/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

7.1.4 Write Documentation

wanikani_api could always use more documentation, whether as part of the official wanikani_api docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kaniwani/wanikani_api/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up *wanikani_api* for local development.

1. Fork the *wanikani_api* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/wanikani_api.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv wanikani_api
$ cd wanikani_api/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 wanikani_api tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/Kaniwani/wanikani_api/pull_requests and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

```
$ py.test tests.test_wanikani_api
```

7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 8

Credits

8.1 Development Lead

- Gary Grant Graham <gary@kaniwani.com>

8.2 Contributors

None yet. Why not be the first?

CHAPTER 9

History

9.1 0.5.0 (2020-08-18)

- Remove mock responses that included stage names.

9.2 0.5.0 (2020-08-18)

- Remove *passed* from resource.
- Remove *srs_stage_name* from assignments and reviews

9.3 0.4.0 (2020-04-30)

- Add Preferences to User Information
- Add Subscription to User Information

9.4 0.3.0 (2019-11-09)

- Add *auxiliary_meanings* to Subject

9.5 0.2.1 (2019-11-05)

- Fix crash caused by WK removing a field from their API.

9.6 0.1.1 (2018-06-26)

- Change Assignment endpoint to reflect the newly dropped fields from the api (*level* specifically).
- Add some proper String representation
- Work on the Etag cache, bringing it closer to completion.

9.7 0.1.0 (2018-06-26)

- First release on PyPI.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

S

U

V

W