
turkleton Documentation

Release 1.2.1

Eric Scrivner

June 15, 2015

1	turkleton	3
1.1	Installation	3
1.2	Features	3
1.3	Examples	3
2	Installation	7
3	Usage	9
4	Contributing	11
4.1	Types of Contributions	11
4.2	Get Started!	12
4.3	Pull Request Guidelines	12
4.4	Tips	13
5	Credits	15
5.1	Development Lead	15
5.2	Contributors	15
6	History	17
7	1.2.1 (2015-06-15)	19
8	1.2.0 (2015-06-11)	21
9	1.1.0 (2015-06-06)	23
10	1.0.0 (2015-06-05)	25
11	0.1.0 (2015-01-11)	27
12	Indices and tables	29

Contents:

Dead simple Python interfaces for Amazon Mechanical Turk.

- Free software: BSD license
- Documentation: <https://turkleton.readthedocs.org>.

1.1 Installation

Simply use pip to download the package from PyPI

```
$ pip install turkleton
```

1.2 Features

The existing Python APIs for Mechanical Turk are thin wrappers at best - we can do better.

Turkleton aims to leverage the expressive powers of Python to improve the whole situation. While still under active development, the main features are:

- Simple interface for defining tasks from pre-built layouts.
- Simple interface for defining schema of assignment results.
- Easily upload tasks in batches.
- Easily download and validate assignments.

1.3 Examples

In turkleton there are several objects to be aware of: Tasks, HITs, and Assignments. A Task is a template from which HITs are created. A HIT corresponds to HIT in the Amazon Mechanical Turk API and represents an uploaded Task. Assignments are contained within HITs. An individual Assignment represents the set of answers submitted by a single worker. A HIT can have many Assignments.

1.3.1 Setting Up Your Connection

Turkleton uses a per-process global connection. It should be initialized before you attempt to upload or download anything. You can initialize it like so:

```
from turkleton import connection
connection.setup(AWS_ACCESS_KEY, AWS_SECRET_ACCESS_KEY)
```

That's it!

1.3.2 Creating A Task And Uploading It

To define a HIT you create a Task representing the template of the assignment you want a worker to complete. For example:

```
import datetime

from turkleton.assignment import task

class MyTask(task.BaseTask):
    __layout_id__ = 'MY LAYOUT ID'
    __reward__ = 0.25
    __title__ = 'Guess How Old From Picture'
    __description__ = 'Look at a picture and guess how old the person is.'
    __keywords__ = ['image', 'categorization']
    __time_per_assignment__ = datetime.timedelta(minutes=5)
```

Here we've created a Task from an existing layout. Now that we've defined our task we can easily upload HITs by filling out the layout parameters:

```
task = MyTask(image_url='http://test.com/img.png', first_guess='29')
hit = task.upload(batch_id='1234')
```

This will create a new assignment from the task template and upload it to Mechanical Turk. The optional `batch_id` parameter allows you to set the annotation for the task to an arbitrary string that you can use to retrieve tasks later in batches.

You can upload many tasks in a loop easily as follows:

```
for image_url in all_image_urls:
    MyTask.create_and_upload(
        image_url=image_url, first_guess='29', batch_id='1234'
    )
```

If you'd like to leave off the batch id you can also use the context manager:

```
with task.batched_upload(batch_id='1234'):
    for image_url in all_image_urls:
        MyTask.create_and_upload(image_url=image_url, first_guess='29')
```

1.3.3 Downloading The Results

To download results for a HIT you first need to define an assignment. The assignment defines what values are expected and their types. These are used to automatically parse answers to the various questions:


```
from turkleton.assignment import assignment
from turkleton.assignment import answers

class MyAssignment (assignment.BaseAssignment):
    categories = answers.MultiChoiceAnswer(question_name='Categories')
    notes = answers.TextAnswer(question_name='AdditionalNotes', default='')
    does_not_match_any = answers.BooleanAnswer(
        question_name='DoesNotMatchAnyCategories', default=False
    )
```

You can then download all of the HITs in a given batch as follows:

```
from turkleton.assignment import hit
reviewable_hits = hit.get_reviewable_by_batch_id('1234')
```

Each HIT may then have multiple assignments associated with it. You can download the assignments, review them, and then dispose of the HIT as follows:

```
for each in MyAssignment.get_by_hit_id(hit.hit_id):
    print('{} - {} - {}'.format(each.categories, each.notes, each.does_not_match_any))
    if is_valid_assignment(each):
        each.accept('Good job!')
    else:
        each.reject('Assignment does not follow instructions.')
hit.dispose()
```

Installation

At the command line:

```
$ easy_install turkleton
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv turkleton  
$ pip install turkleton
```

Usage

To use turkleton in a project:

```
import turkleton
```

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at <https://github.com/etscrivner/turkleton/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.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/etscrivner/turkleton/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/turkleton.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 turkleton
$ cd turkleton/
$ 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 turkleton tests
$ python setup.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.

4.3 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 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/etscrivner/turkleton/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_turkleton
```

Credits

5.1 Development Lead

- Eric Scrivner <eric.t.scrivner@gmail.com>

5.2 Contributors

None yet. Why not be the first?

History

1.2.1 (2015-06-15)

- Bugfix, error when retrieving hits by batch id

1.2.0 (2015-06-11)

- More answer types
- Bugfix where answers retained single value

1.1.0 (2015-06-06)

- Improvements to connection management
- More convenient syntax for uploading batches

1.0.0 (2015-06-05)

- Major version revisions and updates

0.1.0 (2015-01-11)

- First release on PyPI.

Indices and tables

- `genindex`
- `modindex`
- `search`