Hangman Documentation

Release 2.0.6

Manu Phatak

December 07, 2015

Contents

CHAPTER 1

Contents:

1.1 Hangman

A Python TDD Experiment

A python version agnostic, tox tested, travis-backed program! Documented and distributed.

Has **very high** unit test coverage, with passing tests on every relevant version of python including PyPy.

1.1.1 Features

- Hangman!
- Idiomatic code.
- Thoroughly tested with very high coverage.
- Python version agnostic.
- Demonstrates MVC design out of the scope of web development.
- Documentation.

1.1.2 Compatibility

- Python 2.6
- Python 2.7
- Python 3.3
- Python 3.4
- Python 3.5
- PyPy

1.1.3 Getting Started

At the command line either via easy_install or pip:

```
$ mkvirtualenv hangman  # optional for venv users
$ pip install python_hangman

$ hangman
```

Uninstall

```
$ pip uninstall python_hangman
```

1.1.4 Goal

2.0.0

MVC pattern. The goal was to explicitely demonstrate an MVC pattern out of the scope of web development.

Idiomatic code. In this overhaul there's a big emphasis on idiomatic code. The code should be describing its' own intention with the clarity your grandmother could read.

1.0.0

Learning! This was a Test Driven Development(TDD) exercise.

Also, explored:

- Tox, test automation
- Travis CI
- Python version agnostic programming
- Setuptools
- Publishing on pip
- Coverage via coveralls
- Documentation with sphinx and ReadTheDocs
- Cookiecutter development

1.1.5 Design

MVC Intro

This game roughly follows the **Model-View-Controller(MVC)** pattern. In the latest overhaul, these roles have been explicitely named: *hangman.model*, *hangman.view*, *hangman.controller*.

Traditionally in MVC the `controller` is the focal point. It tells the `view` what information to collect from the user and what to show. It uses that information to communicate with the `model`–also, the data persistence later–and determine the next step. This Hangman MVC adheres to these principals

Model

The model is very simply the hangman game instance–*hangman.model.Hangman*. It's a class. Every class should have "state" and the methods of that class should manage that state. In this case, the "state" is the current "state of the game". The public API are for manageing that state.

The entirety of the game logic is contained in *hangman.model.Hangman*. You could technically play the game in the python console by instantiating the class, submitting guesses with the method *hangman.model.Hangman.guess()* and printing the game state.

For example:

```
>>> from hangman.hangman import Hangman
>>> game = Hangman(answer='hangman')
>>> game.guess('a')
hangman(status='_A___A_', misses=[], remaining_turns=10)

>>> game.guess('n').guess('z').guess('e')
hangman(status='_AN__AN', misses=['E', 'Z'], remaining_turns=8)

>>> game.status
'_AN__AN'

>>> game.misses
['E', 'Z']

>>> game.remaining_turns
8
```

View

hangman.view is a collection of stateless functions that represent the presentation layer. When called these functions handles printing the art to the console, and collecting input from the user.

Controller

In this program, the `controller` is actually the "game_loop"–*hangman.controller.game_loop()*. I still think of it as a `controller` because the role it plays–communicating I/O from the view with the model-persistence layer.

The controller tells the view later what to print and what data to collect. It uses that information update the state of the game (model) and handle game events.

1.1.6 Call Diagram

1.1.7 Credits

Tools used in rendering this package:

- Cookiecutter
- bionikspoon/cookiecutter-pypackage forked from audreyr/cookiecutter-pypackage

1.2 Installation

At the command line either via easy_install or pip:

```
$ pip install python_hangman
```

```
$ easy_install python_hangman
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv hangman
$ pip install python_hangman
```

Uninstall:

```
$ pip uninstall python_hangman
```

1.3 Usage

To use Hangman in a project:

```
import hangman
```

1.4 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:

1.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/bionikspoon/Hangman/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bionikspoon/Hangman/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 :)

1.4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/Hangman.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 hangman
$ cd hangman/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b feature/name-of-your-feature
$ git checkout -b fix/name-of-your-bugfix
```

   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 hangman 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.

1.4.3 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, 3.4, 3.5 and for PyPy. Check https://travis-ci.org/bionikspoon/Hangman/pull_requests and make sure that the tests pass for all supported Python versions.

1.4.4 Tips

To run a subset of tests:

```
$ py.test tests/test_hangman.py
```

1.5 Credits

1.5.1 Development Lead

- Manu Phatak <bionikspoon@gmail.com>

1.5.2 Contributors

None yet. Why not be the first?

1.6 History

1.6.1 Next Release

- Updated docs.

1.6.2 2.0.0 (2015-12-05)

- Establishing a changelog.
- Massive refactoring, explicit MVC structure.
- Code is even more idiomatic!
- Created a *FlashMessage* utility.
- Removed poorly implemented classes in favor of stateless functions.
- Add, Remove support for py35, py32.
- 100% code coverage. (2 untestable, inconsequential lines ignored)

1.7 hangman package

1.7.1 Hangman

A well tested, cli, python version-agnostic, multi-platform hangman game. It's built following TDD principles and each component services a sensibly distinct logical purpose.

1.7.2 Submodules

hangman.__main__

Entry point for `hangman` command.

hangman.controller

This module is responsible for guiding the user through the game.

`hangman.controller.`**`game_loop`**(*game=hangman(status='_____', misses=[], remaining_turns=10), flash=<hangman.utils.FlashMessage object>*)

Main game loop.

Parameters

- **`game`** (hangman.model.Hangman) – Hangman game instance.
- **`flash`** (hangman.utils.FlashMessage) – FlashMessage utility

Returns

hangman.model

This module contains all of the game logic.

class `hangman.model.`**`Hangman`**(*answer=None*)

Bases: `object`

The hangman game object contains the logic for managing the status of the game and raising key game related events.

```
>>> from hangman.model import Hangman
>>> game = Hangman(answer='hangman')
>>> game.guess('a')
hangman(status='_A___A_', misses=[], remaining_turns=10)
```

```
>>> game.guess('n').guess('z').guess('e')
hangman(status='_AN__AN', misses=['E', 'Z'], remaining_turns=8)
```

```
>>> game.status
'_AN__AN'
```

```
>>> game.misses
['E', 'Z']
```

```
>>> game.remaining_turns
8
```

`MAX_TURNS` = 10

`add_hit`(*value*)

Add a hit to the model. Check for game won.

Parameters **`value`** – A single letter.

Raises GameWon

`add_miss`(*value*)

Add a miss to the model. Check for game over.

Parameters **value** – A single letter.

Raises GameOver

guess(*letter*)
Check if guess is a hit or miss.

Parameters **letter** (*str*) – Letter to check

Returns self

Return type *Hangman*

Raises ValueError

hits
Get list of hits.

Return type [str]

is_valid_answer(*word*)
Validate answer. Letters only. Max:16

Parameters **word** (*str*) – Word to validate.

Returns

Return type bool

is_valid_guess(*letter*)
Validate guess. Letters only. Max:1

Parameters **letter** (*str*) – Letter to validate

Returns

Return type bool

misses
Get list of misses.

Return type [str]

remaining_turns
Calculate number of turns remaining.

Returns Number of turns remaining.

Return type int

status
Build a string representation of status with letters for hits and _ for unknowns.

Returns game status as string

Return type str

hangman.utils

App utilities.

class hangman.utils.**WordBank**
Bases: object

Default collection of words to choose from

WORDS = ['ATTEMPT', 'DOLL', 'ELLEN', 'FLOATING', 'PRIDE', 'HEADING', 'FILM', 'KIDS', 'MONKEY', 'LUNG

classmethod get()
Get a random word from word list.

Return str Random word.

classmethod set(**values*)
Set *WordBank* word list.

Parameters **values** (*tuple*) –

class hangman.utils.**FlashMessage**
Bases: object

Basic "flash message" implementation.

game_answer = ''

game_over = False

game_won = False

message = ''

exception hangman.utils.**GameOver**
Bases: exceptions.Exception

Raised when out of turns.

exception hangman.utils.**GameWon**
Bases: exceptions.Exception

Raised when answer has been guessed.

exception hangman.utils.**GameFinished**
Bases: exceptions.Exception

Raised when controller should break game loop.

hangman.view

This module handles user interaction. Printing and prompting.

hangman.view.**build_partial_picture**(*remaining_turns*)
Generator. Draw the iconic hangman game status.

Parameters **remaining_turns** (*int*) – Number of turns remaining.

Returns Line of picture.

hangman.view.**build_partial_status**(*misses_block*)
Generator. Draw game status.

Returns Line of status.

hangman.view.**draw_board**(*game*, *message=<hangman.utils.FlashMessage object>*)
Present the game status with pictures.

Clears the screen. Flashes any messages. Zip the two halves of the picture together.

Parameters

- **game** (*hangman.Hangman*) – game instance
- **message** (hangman.utils.FlashMessage) – flash message

Raises hangman.utils.GameFinished

Returns self

hangman.view.**print_partial_body**(*picture*, *status*)

hangman.view.**print_partial_footer**(*game_status*)

hangman.view.**print_partial_header**()

hangman.view.**print_partial_message**(*flash*)

hangman.view.**print_spacer**()

hangman.view.**prompt_guess**()
Prompt user for a single keystroke.

Returns a single letter

Raises KeyboardInterrupt

hangman.view.**prompt_play_again**()
Prompt user to play again.

Return type bool

Returns bool response

hangman.view.**say_goodbye**()
Write a goodbye message.

CHAPTER 2

Feedback

If you have any suggestions or questions about **Hangman** feel free to email me at bionikspoon@gmail.com.

If you encounter any errors or problems with **Hangman**, please let me know! Open an Issue at the GitHub https://github.com/bionikspoon/Hangman main repository.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

h