
Cardsource Documentation

Release 0.0.1

David Fischer

Aug 24, 2018

Contents

1	Introduction	3
1.1	Philosophy	3
1.2	License	3
2	Installation	5
3	Basic usage	7
3.1	Examples	7
4	Advanced usage	9
5	Performance	11
5.1	Raw data	11
6	API	13
6.1	Basic types	13
6.2	Exceptions	14
7	Contributing	15
7.1	Semantic versioning	15
7.2	Unit testing	15
7.3	Code quality	15
7.4	Documentation	16
8	Changelog	17
9	Indices and tables	19
	Python Module Index	21

Cardsource is a well-tested library that can be used to build your own playing card game or to create Monte Carlo simulations of existing card games.

Contents:

1.1 Philosophy

Cardsource was designed with rapid prototyping and easy simulation in mind. While performance is important (and it is measured constantly), ease of use, elegant design and more obvious correctness should always be emphasized even at the expense of performance.

1.2 License

Cardsource is licensed under the BSD license.

Copyright (c) 2013, the authors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY David Fischer "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL David Fischer BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 2

Installation

Cardsource has no dependencies and can be installed like any Python package.

```
pip install cardsource
```

The source is available on [GitHub](#) and packaged releases are on [PyPI](#)

3.1 Examples

3.1.1 WAR

The game of **War** is entirely determined by starting position. It is also possible that the initial state results in a neverending game.

Creating the game of War using `cardsource` is fairly straight-forward. Take a look at the comments to see the `cardsource` library in action.

```
from cardsource import Deck

deck = Deck()
deck.shuffle()

# Split the deck into two halves
player1 = deck[:26]
player2 = deck[26:]

# Game loop which is potentially infinite
while len(player1) > 0 and len(player2) > 0:
    card1 = player1.pop()
    card2 = player2.pop()
    stakes = [card1, card2]
    winner = None

    # Card gt/lt operations are based on rank alone
    # Suit is not considered
    if card1 > card2:
        winner = player1
    elif card1 < card2:
        winner = player2
```

(continues on next page)

(continued from previous page)

```
else:
    # handle WAR
    while winner is None:
        # Verify the players have enough cards for WAR
        # If either player does not have enough, they
        # automatically lose the WAR and the game.
        if len(player1) < 2:
            winner = player2
            while len(player1) > 0:
                stakes.append(player1.pop())
        elif len(player2) < 2:
            winner = player1
            while len(player2) > 0:
                stakes.append(player2.pop())
        else:
            # append additional stakes for the war
            stakes.append(player1.pop())
            stakes.append(player2.pop())
            card1 = player1.pop()
            card2 = player2.pop()
            stakes.append(card1)
            stakes.append(card2)
            if card1 > card2:
                winner = player1
            elif card2 > card1:
                winner = player2
    for card in stakes:
        winner.appendleft(card)

if len(player1) > 0:
    print("Player 1 wins!")
else:
    print("Player 2 wins!")
```

CHAPTER 4

Advanced usage

There is a small performance suite which can be run with the following:

```
% python setup.py performance
```

5.1 Raw data

	v0.0.1
Python2.6	20.77s
Python2.7	21.13s
Python3.3	27.32s
Pypy	1.333s

The moral of the performance story is that Pypy should be used if performance is important for your use case. However, the current performance tests probably overstate Pypy's performance.

6.1 Basic types

class `cardsource.cards.Card` (*value*)

Represents a single playing card

A `Card` object supports a number of operations.

When compared to another card, a card is greater than another card if the rank (A, 7, 2) is higher than the other card's rank. Suits are not considered. Jokers are higher than any card. For equivalence, both suit and rank must be equal for the objects to be equal.

class `cardsource.hand.Hand`

Represents a playing card game hand containing instances of `cardsource.cards.Card`

A `Hand` is an iterable Python object that supports being added to other hands as well as other common iterable operations. Hands are not directly comparable but this is common in subclasses of `Hand`.

append (*card*)

Adds a `Card` to the hand

This class can be overridden in subclasses to ensure that the correct type of cards are added to the hand. Hands should not contain both instances of `Card` and subclasses of `Card`.

Parameters `card` (`cardsource.cards.Card`) – the card to add

clear ()

Removes all cards from the hand

count (*card*)

Returns the number of instances of the specified card

Parameters `card` (`cardsource.cards.Card`) – the card to search for

Return type `int`

Returns the number of instances of the specified card

extend (*otherhand*)

Extends hand by appending cards from another hand

Parameters **card** (*cardsource.hand.Hand*) – the hand to append to this hand

class *cardsource.deck.Deck* (*numjokers=0*)

Represents a playing card deck optionally with jokers. Each member is an instance of *cardsource.cards.Card*.

A *Deck* is an iterable object that supports a number of standard Python operations like indexing, iteration and slicing.

append (*card*)

Put a card on the top of the deck

Parameters **card** (*cardsource.cards.Card*) – the card to add

appendleft (*card*)

Put a card on the bottom of the deck

Parameters **card** (*cardsource.cards.Card*) – the card to add

clear ()

Remove all cards in the deck

pop ()

Removes and returns the top card in the deck

Raises *IndexError* if the deck is empty

Returns the top card in the deck

Return type *cardsource.cards.Card*

shuffle ()

Shuffle the deck

Uses *random.shuffle*

6.2 Exceptions

exception *cardsource.errors.CardSourceError*

All *cardsource* errors are instances of or derive from this exception

Cardsource should work and is tested on Python 2.6+, 3.3+ and pypy. Pull requests are welcomed on [GitHub](#), but major changes should probably be discussed before simply sending a huge patch.

7.1 Semantic versioning

Cardsource uses [semantic versioning](#) which means that it declares a public API and acts reasonably with respect to version numbers. However, since cardsource has a major version of zero (0.y.x) the public API should not be considered entirely stable. Backward incompatible changes are not introduced lightly and will be documented in depth in the changelog.

7.2 Unit testing

All patches that add features or fix bugs should come with unit tests. Unit tests are run automatically as part of continuous integration and can be run manually with:

```
% python setup.py test
```

If `tox` is installed, unit tests for all supported Python versions can be run with the following command. See `.tox.ini` for details.

```
% tox
```

7.3 Code quality

All code under pokersource is run through [flake8](#) as part of continuous integration. See `.travis.yml` for details.

7.4 Documentation

This documentation is generated using the [sphinx](#) package. All patches that change or add features should include associated documentation changes.

Generating the documentation is done with:

```
% cd docs && make html
```

CHAPTER 8

Changelog

Version 0.0.1 (June 25, 2013)

- First release of cardsource

CHAPTER 9

Indices and tables

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

C

`cardsource.cards`, [13](#)
`cardsource.deck`, [14](#)
`cardsource.errors`, [14](#)
`cardsource.hand`, [13](#)

A

`append()` (`cardsource.deck.Deck` method), 14
`append()` (`cardsource.hand.Hand` method), 13
`appendleft()` (`cardsource.deck.Deck` method), 14

C

`Card` (class in `cardsource.cards`), 13
`cardsource.cards` (module), 13
`cardsource.deck` (module), 14
`cardsource.errors` (module), 14
`cardsource.hand` (module), 13
`CardSourceError`, 14
`clear()` (`cardsource.deck.Deck` method), 14
`clear()` (`cardsource.hand.Hand` method), 13
`count()` (`cardsource.hand.Hand` method), 13

D

`Deck` (class in `cardsource.deck`), 14

E

`extend()` (`cardsource.hand.Hand` method), 13

H

`Hand` (class in `cardsource.hand`), 13

P

`pop()` (`cardsource.deck.Deck` method), 14

S

`shuffle()` (`cardsource.deck.Deck` method), 14