PyDealer Documentation

Release 1.4.0

Alex Crawford

January 12, 2015

Contents

PyDealer is a simple to use Python package for "simulating" decks of standard playing cards (also known as a French deck). PyDealer let's you easily create `Deck` instances, each containing a full 52 card deck of playing cards. Each card is a separate `Card` instance, with a name, value, suit, and abbreviation. There is also the `Stack` class, which is useful for creating hands, or discard piles, etc. It is the backbone of the PyDealer package, and actually the `Deck` class is just a subclass of the `Stack` class.

PyDealer could possibly be used as part of a CLI (command line interface) card-based game, or even a graphical game as well, I suppose. It may also be of interest to beginner Python programmers, since it's a relatively simple package, which I created as a way to learn Python, packaging, testing, documentation (Sphinx), etc. I even ended up learning how to use Git a bit, which I must say was slightly frustrating at first. This package has taught me a lot, and maybe someone else can benefit from it as well. Or maybe not. Either way, here it is.

CHAPTER 1

Quick Usage Example

Here is a quick example, using IDLE, demonstrating how to construct a new `Deck` instance, representing a full French deck of cards, as well as how to shuffle the deck, and deal some cards (7 of them) from it, to a hand. We'll then sort the hand, and print a listing of it's contents, in a human readable way, with a simple print statement.

```
>>> import pydealer
>>> deck = pydealer.Deck()
>>> deck.shuffle()
>>> hand = deck.deal(7)
>>> hand.sort()
>>> print hand
2 of Diamonds
5 of Hearts
9 of Hearts
9 of Spades
Jack of Spades
King of Clubs
Ace of Clubs
```

CHAPTER 2

Table of Contents

2.1 Getting Started

This is an overview of all of the methods/functions that users will likely use most.

Contents

2.1.1 Install/Uninstall with pip

I recommend downloading and installing pip, if you haven't already, and using that to install PyDealer, from the Python Package Index.

Enter one of the following commands into your *nix Bash console, Windows Command Prompt, etc. (after installing pip).

Install

```
$ pip install pydealer
```

Update

```
$ pip install pydealer -U
```

Uninstall

```
$ pip uninstall pydealer
```

2.1.2 Import PyDealer

I'm sure most of you know how this is done already, but for those that don't, here is how you import pydealer.

```
# Import PyDealer
import pydealer

# I, personally, prefer to import PyDealer with a shorter name:
# import pydealer as pd

# I also like to alias the utility functions:
# import pydealer.utils as utils
```

Import Specific Classes/Functions

You can, of course also just import the specific classes/functions that you need.

```
# Import the base classes:
from pydealer import (
    Card,
    Deck,
    Stack
)

# Import specific utility functions:
from pydealer.utils import (
    build_cards,
    compare_stacks,
    check_sorted
    # And/or any other functions you wish to import
)
```

2.1.3 Stack/Deck Manipulation

Construct a Deck

Constructing a new, full deck of cards is about as simple as it gets, but let's just get it out of the way, so I don't have to explain it in every subsequent example.

```
import pydealer

# Construct a Deck instance, with 52 cards.
deck = pydealer.Deck()
```

Set Rank Dict to Reference for Sorting, Etc.

You may supply a new `Deck`/`Stack` instance with a rank dict, which it will refer to when sorting, etc. If none is provided, it defaults to `pydealer.const.DEFAULT_RANKS`.

```
import pydealer
from pydealer.const import POKER_RANKS

# Set the default rank dict to reference.
deck = pydealer.Deck(ranks=POKER_RANKS)
```

You can, of course always change the rank dict after instantiation as well.

```
deck.ranks = POKER_RANKS
```

Construct a Deck that Rebuilds when Empty

You can construct a deck that will rebuild itself when you have dealt all of the cards from it, and try to deal any more, with the `rebuild` argument.

```
import pydealer

# Construct a Deck instance, with 52 cards.
deck = pydealer.Deck(rebuild=True)

# If you want it shuffle when rebuilding:
deck = pydealer.Deck(rebuild=True, re_shuffle=True)
```

Construct an Empty Stack

Constructing a new, empty stack, for use as a hand, discard pile, etc., is as simple as constructing a deck.

```
import pydealer

# Construct a Stack instance, for use as a hand in this case.
hand = pydealer.Stack()
```

Shuffle a Stack/Deck

Shuffling is also simple, and done probably exactly how you might expect. Pretty much everything with PyDealer is simple, because it's such a simple package.

```
import pydealer

deck = pydealer.Deck()

# Shuffle the deck, in place.
deck.shuffle()
```

Sort a Stack/Deck

Sorting is also done like you might expect.

```
import pydealer

deck = pydealer.Deck()

# Sort the deck, in place.
deck.sort()
```

Deal Cards from a Stack/Deck

In this example we will create a `Deck` instance, and then deal 7 cards from it.

```
import pydealer

deck = pydealer.Deck()

# Deal some cards from the deck.
dealt_cards = deck.deal(7)
```

Add Cards to a Stack/Deck

Add to the Top

In this example we will create a `Deck` instance, representing a deck of cards, and a `Stack` instance, which will represent a hand. We will then deal 7 cards from the deck, and add them to the exisiting hand.

```
import pydealer

deck = pydealer.Deck()
hand = pydealer.Stack()

dealt_cards = deck.deal(7)

# Add the cards to the top of the hand (Stack).
hand.add(dealt_cards)
```

If you don't care where the dealt cards are placed in the `Stack`, or are just adding them to the top, you can just use the += operand to add cards to the top of a `Stack`.

```
hand += deck.deal(7)
```

Add to the Bottom

You can also add cards to the bottom of a `Stack`/`Deck` as well, if that is preferred.

```
from pydealer.const import BOTTOM

# Note that the constant ``BOTTOM`` is just the string ```"bottom"```.
# The constant ``TOP`` is the string ```"top"```. This is the default value.
hand.add(dealt_cards, end=BOTTOM)
```

Insert Card Into Position of a Stack/Deck

You can also insert a card into any given position (indice) of the `Stack`/`Deck`.

```
# ``deck`` is a Deck instance, and ``card`` is a Card instance. ``20`` is
# the position (or indice) the card is inserted to.
deck.insert(card, 20)
```

Insert List of Cards Into Position of a Stack/Deck

You can also insert a card into any given position (indice) of the `Stack`/`Deck`.

```
# ``stack`` is a Stack instance, and ``cards`` is a list of Card instances,
# or a Stack/Deck instance. ``20`` is the  position (or indice) the card is
# inserted into.
stack.insert_list(cards, 20)
```

Retrieve a Card at a Given Stack/Deck Indice

In this example we will retrieve (but not remove) the card at a given `Deck` indice (or position, if you prefer). You can access the cards in a PyDealer `Stack` or `Deck` instance just like you would access the items in a list or other sequence in Python.

```
import pydealer

deck = pydealer.Deck()

# Access the indice of the ``Deck`` instance.
card = deck[25]
```

Find Specific Card Locations in a Stack/Deck

Single Card

In this example we will search for a given card in the deck. Users can search using full card names, abbreviations, suits, or values. Just remember that `Deck.find` (and `Stack.find`) return the *indices* of the cards, not the cards themselves, and they always return a list, even if there is only one item in it.

```
import pydealer

deck = pydealer.Deck()

# Find the indice(s) of the Ace of Spades.
indices = deck.find("Ace of Spades")
```

List of Cards

In this example we will search for a given list of cards in the deck. Users can search using full card names, abbreviations, suits, or values, or a mixture of any/all of those. Just remember that `Deck.find_list` (and `Stack.find_list`) return the *indices* of the cards, not the cards themselves, always return a list, even if there is only one item in it.

```
import pydealer

deck = pydealer.Deck()

# Construct a list of terms to search for.
terms = ["Ace of Spades", "QH", "2", "Clubs"]

# Find the indices of the cards matching the terms in the given list.
indices = deck.find_list(terms)
```

Get & Remove Specific Cards from a Stack/Deck

You can get & remove specific cards from a `Stack` or `Deck` instance with a given full card name, abbreviation, suit, value, or indice.

Note that the `Stack` and `Deck` "get methods" always return a list, even if there is only one item in it. And also remember that unlike `Stack` and `Deck` "find methods", which return indices, the `Stack` and `Deck` "get methods" return the card instances themselves.

Single Card

In this example we will retrieve and remove a given card from the deck. If there were more than one "Ace of Spades" in the deck, it would retrieve them all.

```
import pydealer

deck = pydealer.Deck()

# Get the card with the given name from the deck.
cards = deck.get("Ace of Spades")
```

List of Cards

In this example we will retrieve and remove a given list of terms from the deck. For demonstration purposes, I am going to construct a mixed list of terms, including a full card name, abbreviation, face, suit, and indice, just to show that you can do that, if you really want to.

```
import pydealer

deck = Deck()

# Construct a list of terms to search for.
terms = ["Queen of Hearts", "KD", "2", "Clubs", 25]

# Get the cards matching the terms and indices in the given list.
cards = deck.get_list(terms)
```

Empty a Stack/Deck

If, for some reason, you want to empty a Stack/Deck of it's cards, you can use the `Stack.empty` method. This will remove all of the cards from the Stack/Deck and will also return them in a list.

```
import pydealer

deck = pydealer.Deck()

deck.empty()
# Or if you would like to keep the emptied cards elsewhere:
cards = deck.empty()
```

2.1.4 Comparisons/Checks

Get the Size of a Stack/Deck

To get the number of cards in a Stack/Deck, simply access the `size` property. It's the same as doing `len(deck)`, which you could also do.

```
import pydealer

deck = pydealer.Deck()

deck_size = deck.size
```

Compare Two Stacks/Decks

Using the `compare_stacks()` function, you can compare two given `Stack` or `Deck` instances, checking whether or not they contain all of the same cards, based on card faces and suits, *not* card instance. This function *does not* take into account the ordering of either Stack/Deck.

```
import pydealer
from pydealer.utils import compare_stacks

deck_x = pydealer.Deck()
deck_y = pydealer.Deck()

result = compare_stacks(deck_x, deck_y)
```

If you care about the ordering of the Stack/Deck instances as well, you can simply use the == (or `!=`) operand. This is the same as using the `compare_stacks()` function, except it also takes into account the order of each Stack/Deck.

```
import pydealer
from pydealer.utils import compare_stacks

deck_x = pydealer.Deck()
deck_y = pydealer.Deck()

result = deck_x == deck_y
```

You can also, obviously, check whether two Stack/Deck are the same object, using `is`.

```
import pydealer
from pydealer.utils import compare_stacks

deck_x = pydealer.Deck()
deck_y = pydealer.Deck()

result = deck_x is deck_y
```

Compare Two Cards

You can compare two cards just as you would compare a couple of integers, using the standard operands (==, `!=`, >, >=, <, <=). By default, it will compare based on `DEFAULT_RANKS`.

```
import pydealer

deck = pydealer.Deck()

card_x = deck.deal()
card_y = deck.deal()

result = card_x == card_y
result = card_x != card_y
result = card_x > card_y
result = card_x >= card_y
result = card_x < card_y
result = card_x <= card_y
```

If you would prefer to compare using a different rank dictionary, you can use the comparison methods built into the card, and supply the dictionary.

```
import pydealer
from pydealer.const import POKER_RANKS

deck = pydealer.Deck()
```

```
card_x = deck.deal()
card_y = deck.deal()

result = card_x.eq(card_y, POKER_RANKS)  # ==
result = card_x.ne(card_y, POKER_RANKS)  # !=
result = card_x.gt(card_y, POKER_RANKS)  # >
result = card_x.ge(card_y, POKER_RANKS)  # >=
result = card_x.lt(card_y, POKER_RANKS)  # <
result = card_x.le(card_y, POKER_RANKS)  # <=
```

Check if a Stack/Deck is Sorted

Using the `check_sorted()` function, you can check to see if the cards in a given Stack/Deck or list are sorted.

```
import pydealer
from pydealer.utils import check_sorted

deck = pydealer.Deck()

result = check_sorted(deck)
```

2.1.5 Defining New Rank Dictionaries

Defining your own rank dictionaries, for use with sorting functions/methods, etc., is straight forward.

Rank dictionaries are just nested dictionaries containing a `"values"` dict, which itself contains all of the card values, and/or a `"suits"` dict, which itself contains all of the card suits, and their associated values.

```
# Define a new rank dict, ``new_ranks``, with ranks for card faces only.
new_ranks = {
    "values": {
        "Ace": 13,
        "King": 12,
        "Queen": 11,
        "Jack": 10,
        "10": 9,
        "9": 8,
        "8": 7,
        "7": 6,
        "6": 5,
        "5": 4,
        "4": 3,
        "3": 2,
        "2": 1
    }
}

# Define a new rank dict, with ranks for card suits only.
new_ranks = {
    "suits": {
        "Spades": 4,
        "Hearts": 3,
```

```
        "Clubs": 2,
        "Diamonds": 1
    }
}

# Define a new rank dict, with both faces & suits.
new_ranks = {
    "values": {
        "Ace": 13,
        "King": 12,
        "Queen": 11,
        "Jack": 10,
        "10": 9,
        "9": 8,
        "8": 7,
        "7": 6,
        "6": 5,
        "5": 4,
        "4": 3,
        "3": 2,
        "2": 1
    },
    "suits": {
        "Spades": 4,
        "Hearts": 3,
        "Clubs": 2,
        "Diamonds": 1
    }
}
```

2.2 API Documentation

This is the PyDealer API documentation. It contains the documentation extracted from the docstrings of the various classes, methods, and functions in the PyDealer package. If you want to know what a certain function/method does, this is the place to look.

Contents

2.2.1 `card` Module

Source This module contains the `Card` class. Each `Card` instance represents a single playing card, of a given value and suit.

class `pydealer.card.`**`Card`**(*value*, *suit*)

The Card class, each instance representing a single playing card.

Parameters

- **value** (*str*) – The card value.
- **suit** (*str*) – The card suit.

eq(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is equal to `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

ge(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is greater than or equal to `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

gt(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is greater than `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

le(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is less than or equal to `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

lt(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is less than `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

ne(*other*, *ranks=None*)
Compares the card against another card, `other`, and checks whether the card is not equal to `other`, based on the given rank dict.

Parameters

- **other** (*Card*) – The second Card to compare.
- **ranks** (*dict*) – The ranks to refer to for comparisons.

Returns `True` or `False`.

`pydealer.card.`**card_abbrev**(*value*, *suit*)
Constructs an abbreviation for the card, using the given value, and suit.

Parameters

- **value** (*str*) – The value to use.
- **suit** (*str*) – The suit to use.

Returns A newly constructed abbreviation, using the given value & suit

`pydealer.card.`**card_name**(*value*, *suit*)
Constructs a name for the card, using the given value, and suit.

Parameters

- **value** (*str*) – The value to use.
- **suit** (*str*) – The suit to use.

Returns A newly constructed name, using the given value & suit.

2.2.2 `stack` Module

Source This module contains the `Stack` class, which is the backbone of the PyDealer package. A `Stack` is essentially just a generic "card container", with all of the methods users may need to work with the cards they contain. A `Stack` can be used as a hand, or a discard pile, etc.

class `pydealer.stack.`**Stack**(***kwargs*)
The Stack class, representing a collection of cards. This is the main 'card container' class, with methods for manipulating it's contents.

Parameters

- **cards** (*list*) – A list of cards to be the initial contents of the Stack.
- **ranks** (*dict*) – If `sort=True`, The rank dict to reference for sorting. Defaults to `DEFAULT_RANKS`.
- **sort** (*bool*) – Whether or not to sort the stack upon instantiation.

add(*cards*, *end='top'*)
Adds the given list of `Card` instances to the top of the stack.

Parameters

- **cards** – The cards to add to the `Stack`. Can be a single `Card` instance, or a `list` of cards.
- **end** (*str*) – The end of the `Stack` to add the cards to. Can be `TOP` ("top") or `BOTTOM` ("bottom").

cards
The cards property.

Returns The cards in the Stack/Deck.

deal(*num=1*, *end='top'*)
Returns a list of cards, which are removed from the Stack.

Parameters

- **num** (*int*) – The number of cards to deal.
- **end** (*str*) – Which end to deal from. Can be `0` (top) or `1` (bottom).

Returns The given number of cards from the stack.

empty(*return_cards=False*)
Empties the stack, removing all cards from it, and returns them.

Parameters **return_cards** (*bool*) – Whether or not to return the cards.

Returns If `return_cards=True`, a list containing the cards removed from the Stack.

find(*term*, *limit=0*, *sort=False*, *ranks=None*)
Searches the stack for cards with a value, suit, name, or abbreviation matching the given argument, 'term'.

Parameters

- **term** (*str*) – The search term. Can be a card full name, value, suit, or abbreviation.
- **limit** (*int*) – The number of items to retrieve for each term. `0` equals no limit.
- **sort** (*bool*) – Whether or not to sort the results.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of stack indices for the cards matching the given terms, if found.

find_list(*terms*, *limit=0*, *sort=False*, *ranks=None*)
Searches the stack for cards with a value, suit, name, or abbreviation matching the given argument, 'terms'.

Parameters

- **terms** (*list*) – The search terms. Can be card full names, suits, values, or abbreviations.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of stack indices for the cards matching the given terms, if found.

get(*term*, *limit=0*, *sort=False*, *ranks=None*)
Get the specified card from the stack.

Parameters

- **term** – The search term. Can be a card full name, value, suit, abbreviation, or stack indice.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of the specified cards, if found.

get_list(*terms*, *limit=0*, *sort=False*, *ranks=None*)
Get the specified cards from the stack.

Parameters

- **term** – The search term. Can be a card full name, value, suit, abbreviation, or stack indice.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of the specified cards, if found.

insert(*card*, *indice=-1*)
Insert a given card into the stack at a given indice.

Parameters

- **card** (*Card*) – The card to insert into the stack.
- **indice** (*int*) – Where to insert the given card.

insert_list(*cards*, *indice=-1*)
Insert a list of given cards into the stack at a given indice.

Parameters

- **cards** (*list*) – The list of cards to insert into the stack.
- **indice** (*int*) – Where to insert the given cards.

is_sorted(*ranks=None*)
Checks whether the stack is sorted.

Parameters **ranks** (*dict*) – The rank dict to reference for checking. If `None`, it will default to `DEFAULT_RANKS`.

Returns Whether or not the cards are sorted.

open_cards(*filename=None*)
Open cards from a txt file.

Parameters **filename** (*str*) – The filename of the deck file to open. If no filename given, defaults to "cards-YYYYMMDD.txt", where "YYYYMMDD" is the year, month, and day. For example, "cards-20140711.txt".

random_card(*remove=False*)
Returns a random card from the Stack. If `remove=True`, it will also remove the card from the deck.

Parameters **remove** (*bool*) – Whether or not to remove the card from the deck.

Returns A random Card object, from the Stack.

reverse()
Reverse the order of the Stack in place.

save_cards(*filename=None*)
Save the current stack contents, in plain text, to a txt file.

Parameters **filename** (*str*) – The filename to use for the file. If no filename given, defaults to "cards-YYYYMMDD.txt", where "YYYYMMDD" is the year, month, and day. For example, "cards-20140711.txt".

set_cards(*cards*)
Change the Deck's current contents to the given cards.

Parameters **cards** (*list*) – The Cards to assign to the stack.

shuffle(*times=1*)
Shuffles the Stack.

Note: Shuffling large numbers of cards (100,000+) may take a while.

Parameters **times** (*int*) – The number of times to shuffle.

size
Counts the number of cards currently in the stack.

Returns The number of cards in the stack.

sort(*ranks=None*)
Sorts the stack, either by poker ranks, or big two ranks.

Parameters **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns The sorted cards.

split(*indice=None*)
Splits the Stack, either in half, or at the given indice, into two separate Stacks.

Parameters **indice** (*int*) – Optional. The indice to split the Stack at. Defaults to the middle of the `Stack`.

Returns The two parts of the Stack, as separate Stack instances.

pydealer.stack.**convert_to_stack**(*deck*)
Convert a `Deck` to a `Stack`.

Parameters **deck** (*Deck*) – The `Deck` to convert.

Returns A new `Stack` instance, containing the cards from the given `Deck` instance.

2.2.3 deck Module

Source This module contains the `Deck` class. Each `Deck` instance contains a full, 52 card French deck of playing cards upon instantiation. The `Deck` class is a subclass of the `Stack` class, with a few extra/differing methods.

class pydealer.deck.**Deck**(***kwargs*)
Bases: pydealer.stack.Stack

The Deck class, representing the deck that the cards will be in. It is a sublcass of Stack, sharing all of the same methods, in addition to a couple of others you would expect a deck class to have.

Warning: At the moment, adding Jokers may cause some (most) functions/methods to throw errors.

Parameters

- **cards** – A list of cards to be the initial contents of the Deck. If provided, the deck will not automatically build a new deck. Can be a `Stack`, `Deck`, or `list` instance.
- **jokers** (*bool*) – Whether or not to include jokers in the deck.
- **num_jokers** (*int*) – How many jokers to add to the deck.

- **build** (*bool*) – Whether or not to build the deck on instantiation.
- **rebuild** (*bool*) – Whether or not to rebuild the deck when it runs out of cards due to dealing.
- **re_shuffle** (*bool*) – Whether or not to shuffle the deck after rebuilding.
- **ranks** (*dict*) – The rank dict that will be referenced by the sorting methods etc. Defaults to `DEFAULT_RANKS`

build (*jokers=False*, *num_jokers=0*)
Builds a standard 52 card French deck of Card instances.

Parameters

- **jokers** (*bool*) – Whether or not to include jokers in the deck.
- **num_jokers** (*int*) – The number of jokers to include.

deal (*num=1*, *rebuild=False*, *shuffle=False*, *end='top'*)
Returns a list of cards, which are removed from the deck.

Parameters

- **num** (*int*) – The number of cards to deal.
- **rebuild** (*bool*) – Whether or not to rebuild the deck when cards run out.
- **shuffle** (*bool*) – Whether or not to shuffle on rebuild.
- **end** (*str*) – The end of the `Stack` to add the cards to. Can be `TOP` ("top") or `BOTTOM` ("bottom").

Returns A given number of cards from the deck.

pydealer.deck.**convert_to_deck** (*stack*)
Convert a `Stack` to a `Deck`.

Parameters **stack** (*Stack*) – The `Stack` instance to convert.

2.2.4 `tools` Module

Source The tools module contains functions for working with sequences of cards, some of which are used by the classes in the PyDealer package, such as the functions `build_cards`, `sort_cards`, and `check_term` for example.

pydealer.tools.**build_cards** (*jokers=False*, *num_jokers=0*)
Builds a list containing a full French deck of 52 Card instances. The cards are sorted according to `DEFAULT_RANKS`.

Parameters

- **jokers** (*bool*) – Whether or not to include jokers in the deck.
- **num_jokers** (*int*) – The number of jokers to include.

Returns A list containing a full French deck of 52 Card instances.

pydealer.tools.**check_sorted** (*cards*, *ranks=None*)
Checks whether the given cards are sorted by the given ranks.

Parameters

- **cards** – The cards to check. Can be a `Stack`, `Deck`, or `list` of `Card` isntances.
- **ranks** (*dict*) – The ranks to check against. Default is DEFAULT_RANKS.

Returns `True` or `False`.

`pydealer.tools.`**`check_term`**(*card*, *term*)
Checks a given search term against a given card's full name, suit, value, and abbreviation.

Parameters

- **card** (*Card*) – The card to check.
- **term** (*str*) – The search term to check for. Can be a card full name, suit, value, or abbreviation.

Returns `True` or `False`.

`pydealer.tools.`**`compare_stacks`**(*cards_x*, *cards_y*, *sorted=False*)
Checks whether two given `Stack`, `Deck`, or `list` instances, contain the same cards (based on value & suit, not instance). Does not take into account the ordering.

Parameters

- **cards_x** – The first stack to check. Can be a `Stack`, `Deck`, or `list` instance.
- **cards_y** – The second stack to check. Can be a `Stack`, `Deck`, or `list` instance.
- **sorted** (*bool*) – Whether or not the cards are already sorted. If `True`, then `compare_stacks` will skip the sorting process.

Returns `True` or `False`.

`pydealer.tools.`**`find_card`**(*cards*, *term*, *limit=0*, *sort=False*, *ranks=None*)
Searches the given cards for cards with a value, suit, name, or abbreviation matching the given argument, `term`.

Parameters

- **cards** – The cards to search. Can be a `Stack`, `Deck` or `list`.
- **term** (*str*) – The search term. Can be a card full name, value, suit, or abbreviation.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of indices for the cards matching the given terms, if found.

`pydealer.tools.`**`find_list`**(*cards*, *terms*, *limit=0*, *sort=False*, *ranks=None*)
Searches the given cards for cards with a value, suit, name, or abbreviation matching the given argument, `terms`.

Parameters

- **cards** – The cards to search. Can be a `Stack`, `Deck` or `list`.
- **terms** (*list*) – The search terms. Can be card full names, suits, values, or abbreviations.
- **limit** (*int*) – The number of items to retrieve for each term. 0 == no limit.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns A list of indices for the cards matching the given terms, if found.

`pydealer.tools.`**`get_card`**(*cards*, *term*, *limit=0*, *sort=False*, *ranks=None*)
Get the specified card from the stack.

Parameters

- **cards** – The cards to get from. Can be a `Stack`, `Deck` or `list`.
- **term** (*str*) – The card's full name, value, suit, abbreviation, or stack indice.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – If `sort=True`, the rank dict to refer to for sorting.

Returns A copy of the given cards, with the found cards removed, and a list of the specified cards, if found.

`pydealer.tools.`**`get_list`**(*cards*, *terms*, *limit=0*, *sort=False*, *ranks=None*)
Get the specified cards from the stack.

Parameters

- **cards** – The cards to get from. Can be a `Stack`, `Deck` or `list`.
- **terms** (*list*) – A list of card's full names, values, suits, abbreviations, or stack indices.
- **limit** (*int*) – The number of items to retrieve for each term.
- **sort** (*bool*) – Whether or not to sort the results, by poker ranks.
- **ranks** (*dict*) – If `sort=True`, the rank dict to refer to for sorting.

Returns A list of the specified cards, if found.

`pydealer.tools.`**`open_cards`**(*filename=None*)
Open cards from a txt file.

Parameters **filename** (*str*) – The filename of the deck file to open. If no filename given, defaults to "cards-YYYYMMDD.txt", where "YYYYMMDD" is the year, month, and day. For example, "cards-20140711.txt".

Returns The opened cards, as a list.

`pydealer.tools.`**`random_card`**(*cards*, *remove=False*)
Returns a random card from the Stack. If `remove=True`, it will also remove the card from the deck.

Parameters **remove** (*bool*) – Whether or not to remove the card from the deck.

Returns A random Card object, from the Stack.

`pydealer.tools.`**`save_cards`**(*cards*, *filename=None*)
Save the given cards, in plain text, to a txt file.

Parameters

- **cards** – The cards to save. Can be a `Stack`, `Deck`, or `list`.
- **filename** (*str*) – The filename to use for the cards file. If no filename given, defaults to "cards-YYYYMMDD.txt", where "YYYYMMDD" is the year, month, and day. For example, "cards-20140711.txt".

`pydealer.tools.`**`sort_card_indices`**(*cards*, *indices*, *ranks=None*)
Sorts the given Deck indices by the given ranks. Must also supply the `Stack`, `Deck`, or `list` that the indices are from.

Parameters

- **cards** – The cards the indices are from. Can be a `Stack`, `Deck`, or `list`
- **indices** (*list*) – The indices to sort.

- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns The sorted indices.

`pydealer.tools.`**`sort_cards`**(*cards*, *ranks=None*)

Sorts a given list of cards, either by poker ranks, or big two ranks.

Parameters

- **cards** – The cards to sort.
- **ranks** (*dict*) – The rank dict to reference for sorting. If `None`, it will default to `DEFAULT_RANKS`.

Returns The sorted cards.

2.2.5 `const` Module

Source These are the few constants that are used by the PyDealer package. The poker ranks, and big two ranks could be used for sorting, or by anyone making a game that relies on those ranks. PyDealer references `DEFAULT_RANKS` for sorting order, and ordering of newly instantiated decks by default.

`pydealer.const.`**`SUITS`**

```
["Diamonds", "Clubs", "Hearts", "Spades"]
```

`pydealer.const.`**`VALUES`**

```
["2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"]
```

`pydealer.const.`**`BIG2_RANKS`**

```
{
    "values": {
        "2": 13,
        "Ace": 12,
        "King": 11,
        "Queen": 10,
        "Jack": 9,
        "10": 8,
        "9": 7,
        "8": 6,
        "7": 5,
        "6": 4,
        "5": 3,
        "4": 2,
        "3": 1,
    },
    "suits": {
        "Spades": 4,
        "Hearts": 3,
        "Clubs": 2,
        "Diamonds": 1
    }
}
```

`pydealer.const.`**`DEFAULT_RANKS`**

```
{
    "values": {
        "Ace": 13,
        "King": 12,
        "Queen": 11,
        "Jack": 10,
        "10": 9,
        "9": 8,
        "8": 7,
        "7": 6,
        "6": 5,
        "5": 4,
        "4": 3,
        "3": 2,
        "2": 1
    },
    "suits": {
        "Spades": 4,
        "Hearts": 3,
        "Clubs": 2,
        "Diamonds": 1
    }
}
```

pydealer.const.**POKER_RANKS**

```
{
    "Ace": 13,
    "King": 12,
    "Queen": 11,
    "Jack": 10,
    "10": 9,
    "9": 8,
    "8": 7,
    "7": 6,
    "6": 5,
    "5": 4,
    "4": 3,
    "3": 2,
    "2": 1
}
```

pydealer.const.**TOP**

```
"top"
```

pydealer.const.**BOTTOM**

```
"bottom"
```

2.3 License

of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to

avoid the special danger that patents applied to a free program could
make it effectively proprietary. To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License. Each licensee is addressed as "you". "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy. The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy. Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies. Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License. If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it. "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

```
  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.
```

```
  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
```

"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source. This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge. You need not require recipients to copy the
    Corresponding Source along with the object code. If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source. Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling. In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage. For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product. A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source. The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information. But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed. Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law. If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option

```
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
```

paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License. If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program. Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance. However,
nothing other than this License grants you permission to propagate or
modify any covered work. These actions infringe copyright if you do
not accept this License. Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License. You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations. If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License. For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

```
  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
```

parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all. For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work. The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time. Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number. If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a

later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

```
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

2.4 Indices and tables

- *genindex*
- *modindex*

p

T

V