

---

# **Poker Documentation**

***Release 0.23.1***

**Kiss György**

**Aug 27, 2018**



<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Installation	3
1.2	Basic operations	3
1.2.1	Card suits	3
1.2.2	Card ranks	4
1.2.3	Cards	4
1.2.4	Implementing a deck	4
1.2.5	Operations with Hands and Combos	4
1.3	Range parsing	5
1.3.1	Defining ranges	5
1.3.2	Normalization	6
1.3.3	Printing the range as an HTML table	7
1.3.4	Printing the range as an ASCII table	7
1.4	Hand history parsing	7
1.4.1	Parsing from hand history text	8
1.4.2	Parsing from file	8
1.4.3	Example	8
1.4.4	API	9
1.4.5	About hand history changes	9
1.5	Getting information from poker related websites	9
1.5.1	PokerStars status	9
1.5.2	List of upcoming tournaments from PokerStars	9
1.5.3	Information about a Two plus two forum member	10
1.5.4	Getting the top 100 players from Pocketfives	11
1.6	Room specific operations	11
1.6.1	Manipulating PokerStars player notes	11
1.7	Development	12
1.7.1	Git repository	12
1.7.2	Versioning	12
1.7.3	Coding style	12
1.7.4	Dates and times	12
1.7.5	Unicode vs Bytes	13
1.7.6	New hand history parser	13
1.7.7	Testing	14
1.7.8	About the state of Python3	14
1.8	Glossary	14

1.9	License . . . . .	15
<b>2</b>	<b>API</b>	<b>17</b>
2.1	Card API . . . . .	17
2.1.1	Suit . . . . .	17
2.1.2	Rank . . . . .	17
2.1.3	Card . . . . .	18
2.2	Hand API . . . . .	18
2.2.1	Shape . . . . .	18
2.2.2	Hand . . . . .	18
2.2.3	Combo . . . . .	19
2.2.4	Range . . . . .	20
2.3	Hand history parsing API . . . . .	21
2.3.1	Constant values . . . . .	21
2.3.2	Base classes . . . . .	23
2.3.3	PokerStars . . . . .	25
2.3.4	Full Tilt Poker . . . . .	25
2.3.5	PKR . . . . .	26
2.4	Room specific classes API . . . . .	26
2.4.1	Pokerstars player notes . . . . .	26
2.5	Website API . . . . .	27
2.5.1	Two Plus Two Forum API . . . . .	27
2.5.2	Pocketfives API . . . . .	28
2.5.3	PokerStars website API . . . . .	29
<b>3</b>	<b>Repo and contact</b>	<b>31</b>
<b>4</b>	<b>Indices and tables</b>	<b>33</b>
	<b>Python Module Index</b>	<b>35</b>

A Python framework for poker related operations.

It contains classes for parsing card Suits, Cards, Hand combinations (called Combos), construct hand Ranges and check for syntax, parse Hand histories.

It can get information from poker related websites like Pocketfives, TwoplusTwo Forum, or PokerStars website by scraping them. In the long term, it will have a fast hand evaluator and an equity calculator.

It uses the MIT license, so its code can be used in any product without legal consequences.

It aims for quality, fully tested code and easy usability with nice APIs, suitable for beginners to play with.



## 1.1 Installation

Simple from PYPI:

```
$ pip install poker
```

Advanced, directly from package in development mode:

```
$ git clone git@github.com:pokerregion/poker.git
$ cd poker
$ pip install -e .
```

## 1.2 Basic operations

### 1.2.1 Card suits

Enumeration of suits:

```
>>> from poker import Suit
>>> list(Suit)
[ Suit(''), Suit(''), Suit(''), Suit('') ]
```

Suits are comparable:

```
>>> Suit.CLUBS < Suit.DIAMONDS
True
```

## 1.2.2 Card ranks

Enumeration of ranks:

```
>>> from poker import Rank
>>> list(Rank)
[Rank('2'), Rank('3'), Rank('4'), Rank('5'), Rank('6'), Rank('7'), Rank('8'), Rank('9
↪'), Rank('T'), Rank('J'), Rank('Q'), Rank('K'), Rank('A')]
```

Ranks are comparable:

```
>>> Rank('2') < Rank('A')
True
```

Making a random Rank:

```
>> Rank.make_random()
Rank('2')
```

## 1.2.3 Cards

Making a random Card:

```
>>> Card.make_random()
Card('As')
```

Comparing Cards:

```
>>> Card('As') > Card('Ks')
True
>>> Card('Tc') < Card('Td')
True
```

## 1.2.4 Implementing a deck

A deck is just a list of `poker.card.Cards`. Making a new deck and simulating shuffling is easy:

```
import random
from poker import Card

deck = list(Card)
random.shuffle(deck)

flop = [deck.pop() for __ in range(3)]
turn = deck.pop()
river = deck.pop()
```

## 1.2.5 Operations with Hands and Combos

```
>>> from poker.hand import Hand, Combo
```

List of all hands:



```
>>> list(Hand)
[Hand('32o'), Hand('32s'), Hand('42o'), Hand('42s'), Hand('43o'), Hand('43s'), Hand(
  ↪ '52o'),
 ..., Hand('JJ'), Hand('QQ'), Hand('KK'), Hand('AA')]
```

Comparing:

```
>>> Hand('AAd') > Hand('KK')
True
>>> Combo('7s6s') > Combo('6d5d')
True
```

Sorting:

```
>>> sorted([Hand('22'), Hand('66'), Hand('76o')])
[Hand('76o'), Hand('22'), Hand('66')]
```

Making a random hand:

```
>>> Hand.make_random()
Hand('AJs')
```

## 1.3 Range parsing

The `poker.hand.Range` class parses human readable (text) ranges like "22+ 54s 76s 98s AQo+" to a set of *Hands* and hand *Combos*.

Can parse ranges and compose parsed ranges into human readable form again.

It's very fault-tolerant, so it's easy and fast to write ranges manually.

Can normalize unprecise human readable ranges into a precise human readable form, like "22+ AQo+ 33 AKo"  
→ "22+ AQo+"

Can tell how big a range is by *Percentage* or number of *Combos*.

### 1.3.1 Defining ranges

Atomic signs

X	means “any card”
A K Q J T 9 8 7 6 5 4 3 2	Ace, King, Queen, Jack, Ten, 9, ..., deuce
“s” or “o” after hands like AKo or 76s	suited and offsuit. Pairs have no suit ('')
-	hands worse, down to deuces
+	hands better, up to pairs

Available formats for defining ranges:

Format	Parsed range
22	one pair
44+	all pairs better than 33
66-	all pairs worse than 77
55-33	55, 44, 33

None of these below select pairs (for unambiguity):

AKo, J9o	offsuit hands
AKs, 72s	suited hands
AJo+ Q8o+	offsuit hands above this: AJo, AQo, AKo Q8o, Q9o, QTo, QJo
AJs+	same as offsuit
76s+	this is valid, although "+" is not necessary, because there are no suited cards above 76s
A5o-	offsuit hands; A5o-A2o
A5s-	suited hands; A5s-A2s
K7	suited and offsuited version of hand; K7o, K7s
J8o-J4o	J8o, J7o, J6o, J5o, J4o
76s-74s	76s, 75s, 74s
J8-J4	both ranges in suited an offsuited form; J8o, J7o, J6o, J5o, J4o, J8s, J7s, J6s, J5s, J4s
A5+	either suited or offsuited hands that contains an Ace and the other is bigger than 5. Same as "A5o+ A5s+".
A5-	downward, same as above
XX	every hand (100% range) In this special case, pairs are also included (but only this)
AX	Any hand that contains an ace either suited or offsuit (no pairs)
AXo	Any offsuit hand that contains an Ace (equivalent to A2o+)
AXs	Any suited hand that contains an Ace (equivalent to A2s+)
QX+	Any hand that contains a card bigger than a Jack; Q2+, K2+, A2+
5X-	any hand that contains a card lower than 6
KXs+	Any suited hand that contains a card bigger than a Queen
KXo+	same as above with offsuit hands
7Xs-	same as above
8Xo-	same as above
2s2h, AsKc	exact hand <i>Combos</i>

---

**Note:** "Q+" and "Q-" are invalid ranges, because in Hold'em, there are two hands to start with not one.

---

Ranges are case insensitive, so "AKs" and "aks" and "aKS" means the same. Also the order of the cards doesn't matter. "AK" is the same as "KA". Hands can be separated by space (even multiple), comma, colon or semicolon, and Combo of them (multiple spaces, etc.).

### 1.3.2 Normalization

Ranges should be rearranged and parsed according to these rules:

- hands separated with one space only in repr, with " , " in str representation
- in any given hand the first card is bigger than second (except pairs of course)
- pairs first, if hyphenated, bigger first
- suited hands after pairs, descending by rank
- offsuited hands at the end

### 1.3.3 Printing the range as an HTML table

Range has a method `to_html()`. When you print the result of that, you get a simple HTML table representation of it.

`Range('XX').to_html()` looks like this:

You can format it with CSS, you only need to define `td.pair`, `td.offsuit` and `td.suited` selectors. It's easy to recreate PokerStove style colors:

```
<style>
  td {
    /* Make cells same width and height and centered */
    width: 30px;
    height: 30px;
    text-align: center;
    vertical-align: middle;
  }
  td.pair {
    background: #aaff9f;
  }
  td.offsuit {
    background: #bbced3;
  }
  td.suited {
    background: #e37f7d;
  }
</style>
```

### 1.3.4 Printing the range as an ASCII table

`to_ascii()` can print a nicely formatted ASCII table to the terminal:

```
>>> print(Range('22+ A2+ KT+ QJ+ 32 42 52 62 72').to_ascii())
AA  AKs AQs AJs ATs A9s A8s A7s A6s A5s A4s A3s A2s
AKo KK  KQs KJs KTs
AQo KQo QQ  QJs
AJo KJo QJo JJ
ATo KTo      TT
A9o          99
A8o          88
A7o          77          72s
A6o          66          62s
A5o          55          52s
A4o          44          42s
A3o          33          32s
A2o          72o 62o 52o 42o 32o 22
```

## 1.4 Hand history parsing

The classes in `poker.room` can parse hand histories for different poker rooms. Right now for PokerStars, Full Tilt Poker and PKR, very efficiently with a simple API.

### 1.4.1 Parsing from hand history text

In one step:

```
from poker.room.pokerstars import PokerStarsHandHistory
# First step, only raw hand history is saved, no parsing will happen yet
hh = PokerStarsHandHistory(hand_text)
# You need to explicitly parse. This will parse the whole hh at once.
hh.parse()
```

Or in two steps:

```
from poker.room.pokerstars import PokerStarsHandHistory
hh = PokerStarsHandHistory(hand_text)
# parse the basic information only (really fast)
hh.parse_header()
# And later parse the body part. This might happen e.g. in a background task
>>> hh.parse()
```

I decided to implement this way, and not parse right away at object instantiation, because probably the most common operation will be looking into the hand history as fast as possible for basic information like hand id, *or* deferring the parsing e.g. to a message queue. This way, you basically just save the raw hand history in the instance, pass it to the queue and it will take care of parsing by the `parse()` call.

And also because “Explicit is better than implicit.”

### 1.4.2 Parsing from file

```
>>> hh = PokerStarsHandHistory.from_file(filename)
>>> hh.parse()
```

### 1.4.3 Example

```
>>> from poker.room.pokerstars import PokerStarsHandHistory
>>> hh = PokerStarsHandHistory(hand_text)
>>> hh.parse()
>>> hh.players
[_Player(name='flett12', stack=1500, seat=1, combo=None),
 _Player(name='santy312', stack=3000, seat=2, combo=None),
 _Player(name='flavio766', stack=3000, seat=3, combo=None),
 _Player(name='strongi82', stack=3000, seat=4, combo=None),
 _Player(name='W2lkm2n', stack=3000, seat=5, combo=Combo('AJ')),
 _Player(name='MISTRPerfect', stack=3000, seat=6, combo=None),
 _Player(name='blak_douglas', stack=3000, seat=7, combo=None),
 _Player(name='sinus91', stack=1500, seat=8, combo=None),
 _Player(name='STBIJUJA', stack=1500, seat=9, combo=None)]
>>> hh.date
datetime.datetime(2013, 10, 4, 19, 18, 18, tzinfo=<DstTzInfo 'US/Eastern' EDT-1 day,
↳20:00:00 DST>)
>>> hh.hero
_Player(name='W2lkm2n', stack=3000, seat=5, combo=Combo('AJ')),
>>> hh.limit, hh.game
('NL', 'HOLDEM')
>>> hh.board
```

(continues on next page)

(continued from previous page)

```
(Card('2'), Card('6'), Card('6'))
>>> hh.flop.is_rainbow
True
>>> hh.flop.has_pair
True
>>> hh.flop.actions
(('W2lkm2n', <Action.BET: ('bet', 'bets')>, Decimal('80')),
 ('MISTRPerfect', <Action.FOLD: ('fold', 'folded', 'folds')>),
 ('W2lkm2n', <Action.RETURN: ('return', 'returned', 'uncalled')>, Decimal('80')),
 ('W2lkm2n', <Action.WIN: ('win', 'won', 'collected')>, Decimal('150')),
 ('W2lkm2n', <Action.MUCK: ("don't show", "didn't show", "did not show", 'mucks')>))
```

## 1.4.4 API

See all the room specific classes in them *Hand history parsing API* documentation.

## 1.4.5 About hand history changes

Poker rooms sometimes change the hand history format significantly. My goal is to cover all hand histories after 2014.01.01., because it is the best compromise between fast development and good coverage. This way we don't have to deal with ancient hand history files and overcomplicate the code and we can concentrate on the future instead of the past. Also, hopefully hand history formats are stable enough nowadays to follow this plan, less and less new game types coming up.

One of the “recent” changes made by Full Tilt is from 2013.05.10.:

“In the software update from Wednesday, changed the format of the . This means that Hold'em Manager does no longer import these hands, and the HUD is not working. ... B.t.w. They just renamed “No Limit Hold'em” to “NL Hold'em”, and swapped position with the blinds, inside the handhistory files.”

Details: <http://www.bankrollmob.com/forum.asp?mode=thread&id=307215>

## 1.5 Getting information from poker related websites

### 1.5.1 PokerStars status

You can get information about PokerStars online players, active tournaments, number of tables currently running:

```
>>> from poker.website.pokerstars import get_status
>>> status = get_status()
>>> status.players, status.tables
(110430, 16427)
```

See the possible attributes in the *API documentation*.

### 1.5.2 List of upcoming tournaments from PokerStars

```
>>> from poker.website.pokerstars import get_current_tournaments
# get_current_tournaments is a generator, so if you want a list, you need to cast it
# otherwise, you can iterate over it
>>> list(get_current_tournaments())
[_Tournament(start_date=datetime.datetime(2014, 8, 16, 8, 2, tzinfo=tzoffset(None, -
↳14400)), name="Copernicus' FL Omaha H/L Freeroll", game='Omaha', buyin='$0 + $0',
↳players=2509),
_Tournament(start_date=datetime.datetime(2014, 8, 16, 8, 2, tzinfo=tzoffset(None, -
↳14400)), name='500 Cap: $0.55 NLHE', game="Hold'em", buyin='$0.50 + $0.05',
↳players=80),
_Tournament(start_date=datetime.datetime(2014, 8, 16, 8, 2, tzinfo=tzoffset(None, -
↳14400)), name='Sunday Million Sat [Rd 1]: $0.55+R NLHE [2x-Turbo], 3 Seats Gtd',
↳game="Hold'em", buyin='$0.50 + $0.05', players=14),
_Tournament(start_date=datetime.datetime(2014, 8, 16, 8, 2, tzinfo=tzoffset(None, -
↳14400)), name='$11 NLHE [Phase 1] Sat: 5+R FPP NLHE [2x-Turbo], 2 Seats Gtd', game=
↳"Hold'em", buyin='$0 + $0', players=45),
...
]
```

Or you can iterate over it and use specific data:

```
>>> from poker.website.pokerstars import get_current_tournaments
>>> for tournament in get_current_tournaments():
...     print(tournament.name)
Play Money, No Limit Hold'em + Knockout (5,000)
Sunday Million Sat [Rd 1]: $2.20 NLHE [Turbo]
Play Money, No Limit Omaha (100k)
Play Money, Hourly 1K, NLHE
$11 NL Hold'em [Time: 15 Minutes]
$2.20 NL Hold'em [Heads-Up,128 Cap, Winner-Take-All]
$2.20 NL Hold'em [4-Max, Turbo,5x-Shootout]
$11+R NL Hold'em [Action Hour], $5K Gtd
...
```

### 1.5.3 Information about a Two plus two forum member

If you want to download all the available public information about a forum member (e.g. <http://forumserver.twoplustwo.com/members/115014/>) all you need to do is:

```
>>> from poker.website.twoplustwo import ForumMember
>>> forum_member = ForumMember('Walkman_')
>>> vars(forum_member)
{'public_usergroups': ('Marketplace Approved',),
 'username': 'Walkman_',
 'location': 'Hungary',
 'download_date': datetime.datetime(2014, 8, 29, 16, 30, 45, 64197, tzinfo=datetime.
↳timezone.utc),
 'rank': 'enthusiast',
 'total_posts': 92,
 'id': '115014',
 'join_date': datetime.date(2008, 3, 10),
 'posts_per_day': 0.04,
 'profile_picture': 'http://forumserver.twoplustwo.com/customprofilepics/
↳profilepic115014_1.gif',
 'avatar': 'http://forumserver.twoplustwo.com/customavatars/thumbs/avatar115014_1.gif
↳',
 '...
```

(continues on next page)

(continued from previous page)

```
'last_activity': datetime.datetime(2014, 8, 26, 2, 49, tzinfo=<UTC>)}
```

## 1.5.4 Getting the top 100 players from Pocketfives

```
>>> from poker.website.pocketfives import get_ranked_players
>>> list(get_ranked_players())
[_Player(name='plen01', country='United Kingdom', triple_crowns=1, monthly_win=0,
↳ biggest_cash='$110,874.68', plb_score=7740.52, biggest_score=817.0, average_
↳ score=42.93, previous_rank='2nd'),
_Player(name='p0cket00', country='Canada', triple_crowns=6, monthly_win=0, biggest_
↳ cash='$213,000.00', plb_score=7705.61, biggest_score=1000.0, average_score=47.23,
↳ previous_rank='1st'),
_Player(name='r4ndomr4gs', country='Sweden', triple_crowns=2, monthly_win=1, biggest_
↳ cash='$174,150.00', plb_score=7583.38, biggest_score=803.0, average_score=46.59,
↳ previous_rank='3rd'),
_Player(name='huiiiiiiiiiii', country='Austria', triple_crowns=1, monthly_win=0,
↳ biggest_cash='$126,096.00', plb_score=7276.52, biggest_score=676.0, average_
↳ score=39.26, previous_rank='11th'),
_Player(name='TheClaimeer', country='United Kingdom', triple_crowns=1, monthly_win=0,
↳ biggest_cash='$102,296.00', plb_score=6909.56, biggest_score=505.0, average_
↳ score=41.68, previous_rank='4th'),
```

`poker.website.pocketfives._Player` is a named tuple, so you can look up attributes on it:

```
>>> for player in get_ranked_players():
...     print(player.name, player.country)
plen01 United Kingdom
p0cket00 Canada
r4ndomr4gs Sweden
huiiiiiiiiiii Austria
TheClaimeer United Kingdom
Romeopro Ukraine
PokerKaiser Chile
dipthrong Canad
...
```

## 1.6 Room specific operations

### 1.6.1 Manipulating PokerStars player notes

`poker.room.pokerstars.Notes` class is capable of handling PokerStars Players notes.

You can add and delete labels, and notes, save the modifications to a new file or just print the object instance and get the full XML.

```
>>> from poker.room.pokerstars import Notes
>>> notes = Notes.from_file('notes.W21km2n.xml')
>>> notes.players
('regplayer', 'sharkplayer', 'fishplayer', '"htmlchar"', '$dollarsign',
↳ 'nonoteforplayer',
↳ '-=strangename=-', '//ÄMGS', '0bullmarket0', 'CarlGardner', 'µ (x+t)', 'Walkman')
```

(continues on next page)

(continued from previous page)

```
>>> notes.labels
(_Label(id='0', color='30DBFF', name='FISH'),
 _Label(id='1', color='30FF97', name='SHARK'),
 _Label(id='2', color='E1FF80', name='REG'),
 _Label(id='3', color='E1FF80', name='GENERAL'))

>>> notes.add_label('NIT', 'FF0000')
>>> notes.labels
(_Label(id='0', color='30DBFF', name='FISH'),
 _Label(id='1', color='30FF97', name='SHARK'),
 _Label(id='2', color='E1FF80', name='REG'),
 _Label(id='3', color='E1FF80', name='GENERAL'),
 _Label(id='4', color='FF0000', name='NIT'))
```

For the full API, see the *Room specific classes API*.

## 1.7 Development

### 1.7.1 Git repository

You find the repository on github: <https://github.com/pokerregion/poker>

In the `dend/` branches, there are ideas which doesn't work or has been abandoned for some reason. They are there for reference as "this has been tried".

I develop in a very simple [Workflow](#). (Before 662f5d73be1efbf6eaf173da448e0410da431b2c you can see bigger merge bubbles, because I handled hand history parser code and the rest as two separate projects, but made a subtree merge and handle them in this package.) Feature branches with rebases on top of master. Only merge stable code into master.

The repository tags will match PyPi release numbers.

### 1.7.2 Versioning

I use [Semantic Versioning](#), except for major versions like 1.0, 2.0, because I think 1.0.0 looks stupid :)

### 1.7.3 Coding style

PEP8 except for line length, which is 99 max (hard limit). If your code exceeds 99 characters, you do something wrong anyway, you need to refactor it (e.g. to deeply nested, harder to understand)

### 1.7.4 Dates and times

Every datetime throughout the library is in UTC with `tzinfo` set to `pytz.UTC`. If you found a case where it's not, it's a bug, please [report it on GitHub](#)! The right way for setting a date correctly e.g. from PokerStars ET time is:

```
>>> import pytz
>>> ET = pytz.timezone('US/Eastern')
>>> ET.localize(some_datetime).astimezone(pytz.UTC)
```

This will consider DST settings and ambiguous times. For more information, see [pytz documentation](#)!



## 1.7.5 Unicode vs Bytes

The library uses the strategy from Python 3: internally, everything is using unicode, and things are converted only the I/O boundaries (e.g. opening/writing files). This is implemented by the future mechanism:

```
from __future__ import unicode_literals
```

this way, all newly defined text will be unicode by default. I only check for unicode everywhere in the code, not str, please watch this! It might feel strange the first time, but when the time comes to convert the library to Python3, it will be much easier!

## 1.7.6 New hand history parser

**Note:** Hand history parsing API will change for sure until 1.0 is done.

If you want to support a new poker room you have to subclass the appropriate class from `poker.handhistory` like `poker.handhistory._SplittableHandHistory` depending on the type of hand history file, like XML, or similar to pokerstars and FTP, define a couple of methods and done.

```
class NewPokerRoomHandHistory(HandHistory):
    """Implement PokerRoom specific parsing."""

    def parse_header(self):
        # Parse header only! Usually just the first line. The whole purpose is to do it
        ↪ fast!
        # No need to call super()

    def _parse_table(self):
        # parses table name

    def _parse_players(self):
        # parses players, player positions, stacks, etc
        # set self.players attribute

    def _parse_button(self):

    def _parse_hero(self):

    def _parse_preflop(self):

    def _parse_street(self):

    def _parse_showdown(self):

    def _parse_pot(self):

    def _parse_board(self):

    def _parse_winners(self):

    def _parse_extra(self):
```

You **have to** provide all common attributes, and *may* provide PokerRoom specific extra attributes described in the base `poker.handhistory.HandHistory` class API documentation.

### 1.7.7 Testing

The framework contains a lot of tests (over 400). The basic elements like Card, Hand, Range, etc. are fully tested.

All the unit tests are written in `pytest`. I choose it because it offers very nice functionality, and no-boilerplate code for tests. No need to subclass anything, just prefix classes with `Test` and methods with `test_`.

All assertion use the default python `assert` keyword.

You need to install the `poker` package in development mode:

```
# from directory where setup.py file is
$ pip install -e .
```

and install `pytest` and run it directly:

```
$ pip install pytest
$ py.test
```

from the `poker` module directory and `pytest` will automatically pick up all unit tests.

### 1.7.8 About the state of Python3

Originally I started the library in Python3.4 only, and though everything worked well for a while and library support seemed fine, the tooling around Python3 is not there at all yet. PyPy, PyInstaller Kivy, etc, etc. doesn't support Python3.4 at all, but even if they do, there are bugs/problems which are hard to solve, because nobody experienced it before. A nice example is that I found a bug in the 3.4 standard library `enum` module.

As a consequence of this, at v0.22.0 I converted the whole repository to Python2 only. The long-term strategy is to use Python2 for everything until the tooling for Python3.4 catch up, and at that point, convert it to Python3.4 and continue from there. I don't want to develop a 2 and 3 compatible library for such a niche market, because it's very time consuming, please don't even ask me! (E.g. it took straight 10 hours to convert the whole library to Python2, with no new code at all...)

## 1.8 Glossary

**Suit** One of 'c', 'd', 'h', or 's'. Alternatively '♠', '♥', '♦', '♣'. According to Wikipedia, suits are ranked as:

spades > hearts > diamonds > clubs

**Shape** A hand can have three "Shapes" according to Wikipedia.

'o' for offsuit, 's' for suited hands 'p' for pairs.

**Rank** One card without suit. One of '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A'.

**Card** One exact card with a suit. e.g. 'As', '2s'. It has a *Rank* and a *Suit*.

**Hand** Consists two *Ranks* without precise suits like "AKo", "22".

**Hand comparisons** Comparisons in this library has nothing to do with equities or if a hand beats another. They are only defined so that a consistent ordering can be ensured when representing objects. If you want to compare hands by equity, use `pypoker-eval` instead.

**Comparison rules:**

- pairs are 'better' than none-pairs
- non-pairs are better if at least one of the cards are bigger

- suited better than offsuit

**Combo** Exact two cards with suits specified like “2s2c”, “7s6c”. There are total of 1326 Combos.

**Range** A range of hands with either in *Hand* form or *Combo*. e.g. “55+ AJo+ 7c6h 8s6s”, “66-33 76o-73o AsJc 2s2h” or with other special notation. (See above.)

**Range percent** Compared to the total of 1326 hand *Combos*, how many are in the range?

**Range length**

**Range size** How many concrete hand *Combos* are in the range?

**Range is “bigger” than another** If there are more hand *Combos* in it. (Equity vs each other doesn’t matter here.)

**Token** Denote one part of a range. In a “66-33 76o-73o AsJc 2s2h” range, there are 4 tokens: - “66-33” meaning 33, 44, 55, 66 - “AsJc” specific *Combo* - “2s2h” a specific pair of deuces - “76o-73o” several offsuit *Hands*

**Broadway card** T, J, Q, K, A

**Face card** Only: J, Q, K.

**Warning:** Ace is not a face card!

## 1.9 License

The MIT License (MIT)

Copyright (c) 2013–2014 Kiss György

Permission **is** hereby granted, free of charge, to **any** person obtaining a copy of this software **and** associated documentation files (the “**Software**”), to deal **in** the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, **and/or** sell copies of the Software, **and** to permit persons to whom the Software **is** furnished to do so, subject to the following conditions:

The above copyright notice **and** this permission notice shall be included **in all** copies **or** substantial portions of the Software.

THE SOFTWARE IS PROVIDED “**AS IS**”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



## 2.1 Card API

The `poker.card` module has three basic classes for dealing with card suits, card ranks and cards. It also has a `DECK`, which is just a tuple of `Cards`.

### 2.1.1 Suit

**class** `poker.card.Suit`  
Enumeration of the four *Suits*.

### 2.1.2 Rank

**class** `poker.card.Rank`  
Enumeration of the 13 *Ranks*.

**classmethod** `difference` (*first*, *second*)  
Tells the numerical difference between two ranks.

**Parameters**

- **first** (*str*, *Rank*) –
- **second** (*str*, *Rank*) –

**Returns** value of the difference (always positive)

**Return type** `int`

`poker.card.FACE_RANKS`  
See *Face card*

`poker.card.BROADWAY_RANKS`  
See *Broadway card*

### 2.1.3 Card

```
class poker.card.Card
    Represents a Card, which consists a Rank and a Suit.

    classmethod make_random()
        Returns a random Card instance.

        Return type Card

    is_face
        Type bool

    is_broadway
        Type bool

    rank
        Type Rank

    suit
        Type Suit
```

## 2.2 Hand API

### 2.2.1 Shape

```
class poker.hand.Shape
    See: Shape
```

**Warning:** This might be removed in future version for simplify API.

### 2.2.2 Hand

```
class poker.hand.Hand(hand)
    General hand without a precise suit. Only knows about two ranks and shape.

    Parameters hand (str) – e.g. ‘AKo’, ‘22’

    Variables

    • first (Rank) – first Rank

    • second (Rank) – second Rank

    • shape (Shape) – Hand shape (pair, suited or offsuit)

    rank_difference
        The difference between the first and second rank of the Hand.

        Type int

    first
        Type poker.card.Rank

    second
```

Type `poker.card.Rank`

**shape**

Type `Shape`

**is\_broadway**

**is\_connector**

**is\_offsuit**

**is\_one\_gapper**

**is\_pair**

**is\_suited**

**is\_suited\_connector**

**is\_two\_gapper**

**to\_combos()**

`poker.hand.PAIR_HANDS = (Hand('22'), Hand('33'), Hand('44'), Hand('55'), Hand('66'), Hand('77'), Hand('88'), Hand('99'))`  
 Tuple of all pair hands in ascending order.

`poker.hand.OFFSUIT_HANDS = (Hand('32o'), Hand('42o'), Hand('43o'), Hand('52o'), Hand('53o'), Hand('62o'), Hand('63o'), Hand('72o'), Hand('73o'), Hand('82o'), Hand('83o'), Hand('92o'), Hand('93o'))`  
 Tuple of offsuit hands in ascending order.

`poker.hand.SUITED_HANDS = (Hand('32s'), Hand('42s'), Hand('43s'), Hand('52s'), Hand('53s'), Hand('62s'), Hand('63s'), Hand('72s'), Hand('73s'), Hand('82s'), Hand('83s'), Hand('92s'), Hand('93s'))`  
 Tuple of suited hands in ascending order.

## 2.2.3 Combo

**class** `poker.hand.Combo`

Hand combination.

See *Combo*

**first**

Type `poker.card.Card`

**second**

Type `poker.card.Card`

**shape**

Type `Shape`

**classmethod** `from_cards` (*first, second*)

**is\_broadway**

**is\_connector**

**is\_offsuit**

**is\_one\_gapper**

**is\_pair**

**is\_suited**

**is\_suited\_connector**

**is\_two\_gapper**

**rank\_difference**

The difference between the first and second rank of the Combo.

**to\_hand()**

Convert combo to *Hand* object, losing suit information.

## 2.2.4 Range

**class** `poker.hand.Range` (*range=u''*)

Parses a str range into tuple of Combos (or Hands).

**Parameters** **range** (*str*) – Readable range in unicode

---

**Note:** All of the properties below are `cached_property`, so make sure you invalidate the cache if you manipulate them!

---

**hands**

Tuple of hands contained in this range. If only one combo of the same hand is present, it will be shown here. e.g. `Range('2s2c').hands == (Hand('22'),)`

**Type** tuple of *poker.hand.Hands*

**combos**

**Type** tuple of *poker.hand.Combos*

**percent**

What percent of combos does this range have compared to all the possible combos.

There are 1326 total combos in Hold'em:  $52 * 51 / 2$  (because order doesn't matter) Precision: 2 decimal point

**Type** float (1-100)

**rep\_pieces**

List of str pieces how the Range is represented.

**Type** list of str

**to\_html()**

Returns a 13x13 HTML table representing the range.

The table's CSS class is `range`, pair cells (td element) are `pair`, offsuit hands are `offsuit` and suited hand cells has `suited` css class. The HTML contains no extra whitespace at all. Calculating it should not take more than 30ms (which takes calculating a 100% range).

**Return type** `str`

**to\_ascii** (*border=False*)

Returns a nicely formatted ASCII table with optional borders.

**Return type** `str`

**classmethod** **from\_file** (*filename*)

Creates an instance from a given file, containing a range. It can handle the PokerCruncher (.rng extension) format.

**classmethod** **from\_objects** (*iterable*)

Make an instance from an iterable of Combos, Hands or both.



```
slots = (u'_hands', u'_combos')
```

## 2.3 Hand history parsing API

---

**Note:** Hand history parsing API will change for sure until 1.0 is done.

---

### 2.3.1 Constant values

These enumerations are used to identify common values like limit types, game, etc. By unifying these into groups of enumeration classes, it's possible to have common values across the whole framework, even when parsing totally different kind of hand histories, which uses different values. ([Data normalization](#)) It's recommended to use keys (name property) to save in database, and print them to the user. (E.g. in a web application template, `{{ PokerRoom.STARS }}` will be converted to `'PokerStars'`.)

```
class poker.constants.Action
```

```
BET = (u'bet', u'bets')
CALL = (u'call', u'calls')
CHECK = (u'check', u'checks')
FOLD = (u'fold', u'folded', u'folds')
MUCK = (u'don't show', u'didn't show', u'did not show', u'mucks')
RAISE = (u'raise', u'raises')
RETURN = (u'return', u'returned', u'uncalled')
SHOW = (u'show',)
THINK = (u'seconds left to act',)
WIN = (u'win', u'won', u'collected')
```

```
class poker.constants.Currency
```

```
EUR = (u'EUR', u'\u20ac')
GBP = (u'GBP', u'\xa3')
STARS_COIN = (u'SC', u'StarsCoin')
USD = (u'USD', u'$')
```

```
class poker.constants.Game
```

```
HOLDEM = (u'Hold'em', u'HOLDEM')
OHILO = (u'Omaha Hi/Lo',)
OMAHA = (u'Omaha',)
RAZZ = (u'Razz',)
STUD = (u'Stud',)
```

```
class poker.constants.GameType

    CASH = (u'Cash game', u'CASH', u'RING')
    SNG = (u'Sit & Go', u'SNG', u'SIT AND GO', u'Sit&go')
    TOUR = (u'Tournament', u'TOUR')

class poker.constants.Limit

    FL = (u'FL', u'Fixed limit', u'Limit')
    NL = (u'NL', u'No limit')
    PL = (u'PL', u'Pot limit')

class poker.constants.MoneyType

    PLAY = (u'Play money',)
    REAL = (u'Real money',)

class poker.constants.PokerRoom

    EIGHT = (u'888', u'888poker')
    FTP = (u'Full Tilt Poker', u'FTP', u'FULL TILT')
    PKR = (u'PKR', u'PKR POKER')
    STARS = (u'PokerStars', u'STARS', u'PS')

class poker.constants.Position

    BB = (u'BB', u'big blind')
    BTN = (u'BTN', u'bu', u'button')
    CO = (u'CO', u'cutoff', u'cut off')
    HJ = (u'HJ', u'hijack', u'utg+5', u'utg + 5')
    SB = (u'SB', u'small blind')
    UTG = (u'UTG', u'under the gun')
    UTG1 = (u'UTG1', u'utg+1', u'utg + 1')
    UTG2 = (u'UTG2', u'utg+2', u'utg + 2')
    UTG3 = (u'UTG3', u'utg+3', u'utg + 3')
    UTG4 = (u'UTG4', u'utg+4', u'utg + 4')

class poker.constants.TourFormat

    ACTION = (u'Action Hour',)
    ONEREB = (u'1R1A',)
    REBUY = (u'Rebuy', u'+R')
    SECOND = (u'2x Chance',)
```

```
class poker.constants.TourSpeed
```

```
DOUBLE = (u'2x-Turbo',)
HYPER = (u'Hyper-Turbo',)
REGULAR = (u'Regular',)
SLOW = (u'Slow',)
TURBO = (u'Turbo',)
```

## 2.3.2 Base classes

```
class poker.handhistory._BaseHandHistory(hand_text)
```

Abstract base class for *all* kinds of parser.

**Parameters** `hand_text` (*str*) – poker hand text

The attributes can be iterated.

The class can read like a dictionary.

Every attribute default value is None.

### Variables

- **date\_format** (*str*) – default date format for the given poker room
- **ident** (*str*) – hand id
- **game\_type** (`poker.constants.GameType`) – "TOUR" for tournaments or "SNG" for Sit&Go-s
- **tournament\_ident** (*str*) – tournament id
- **tournament\_level** (*str*) – level of tournament blinds
- **currency** (`poker.constants.Currency`) – 3 letter iso code "USD", "HUF", "EUR", etc.
- **buyin** (`decimal.Decimal`) – buyin **without** rake
- **rake** (`decimal.Decimal`) – if game\_type is "TOUR" it's buyin rake, if "CASH" it's rake from pot
- **game** (`poker.constants.Game`) – "HOLDEM", "OMAHA", "STUD", "RAZZ", etc.
- **limit** (`poker.constants.Limit`) – "NL", "PL" or "FL"
- **sb** (`decimal.Decimal`) – amount of small blind
- **bb** (`decimal.Decimal`) – amount of big blind
- **date** (*datetime*) – hand date in UTC
- **table\_name** (*str*) – name of the table. it's "tournament\_number table\_number"
- **max\_players** (*int*) – maximum players can sit on the table, 2, 4, 6, 7, 8, 9
- **button** (`poker.handhistory._Player`) – player on the button
- **hero** (`poker.handhistory._Player`) – hero player

- **players** (*list*) – list of `poker.handhistory._Player`. the sequence is the seating order at the table at the start of the hand
- **flop** (`_Flop`) – room specific Flop object
- **turn** (`poker.card.Card`) – turn card, e.g. `Card('Ah')`
- **river** (`poker.card.Card`) – river card, e.g. `Card('2d')`
- **board** (*tuple*) – board cards, e.g. `(Card('4s'), Card('4d'), Card('4c'), Card('5h'))`
- **preflop\_actions** (*tuple*) – action lines in str
- **turn\_actions** (*tuple*) – turn action lines
- **turn\_pot** (`decimal.Decimal`) – pot size before turn
- **turn\_num\_players** (*int*) – number of players seen the turn
- **river\_actions** (*tuple*) – river action lines
- **river\_pot** (`decimal.Decimal`) – pot size before river
- **river\_num\_players** (*int*) – number of players seen the river
- **tournament\_name** (*str*) – e.g. "\$750 Guarantee", "\$5 Sit & Go (Super Turbo) "
- **total\_pot** (`decimal.Decimal`) – total pot after end of actions (rake included)
- **show\_down** (*bool*) – There was show\_down or wasn't
- **winners** (*tuple*) – winner names, tuple if even when there is only one winner. e.g. `('W21km2n',)`
- **extra** (*dict*) – Contains information which are specific to a concrete hand history and not common accross all. When iterating through the instance, this extra attribute will not be included. default value is None

**class** `poker.handhistory._Player` (*name, stack, seat, combo*)  
Player participating in the hand history.

#### Variables

- **name** (*str*) – Player name
- **stack** (*int*) – Stack size (sometimes called as chips)
- **seat** (*int*) – Seat number
- **combo** (`Combo, None`) – If the player revealed his/her hand, this property hold's it. None for players didn't show... autoclass:: `poker.handhistory._Player`

Every hand history has an attribute `flop` which is an instance of the room specific `_Flop` object which has the following attributes:

**class** `_Flop`

#### Variables

- **cards** (*tuple*) – tuple of `poker.card.Cards`
- **pot** (`decimal.Decimal`) – pot size after actions
- **players** (*tuple*) – tuple of player names
- **actions** (*tuple*) –

tuple of `poker.constants.Action` in the order of happening.

Form:

(Player name, Action, Amount) or

(Player name, Action) if no amount needed (e.g. in case of Check)

It also has properties about flop texture like:

#### Variables

- `is_rainbow` (*bool*) –
- `is_monotone` (*bool*) –
- `is_triplet` (*bool*) –
- `has_pair` (*bool*) –
- `has_straightdraw` (*bool*) –
- `has_gutshot` (*bool*) –
- `has_flushdraw` (*bool*) –

### 2.3.3 PokerStars

`class poker.room.pokerstars.PokerStarsHandHistory` (*hand\_text*)

Parses PokerStars Tournament hands.

### 2.3.4 Full Tilt Poker

`class poker.room.fulltiltpoker.FullTiltPokerHandHistory` (*hand\_text*)

Parses Full Tilt Poker hands the same way as `PokerStarsHandHistory` class.

PokerStars and Full Tilt hand histories are very similar, so parsing them is almost identical. There are small differences though.

#### Class specific

##### Variables

- `tournament_level` – None
- `buyin` – None: it's not in the hand history, but the filename
- `rake` – None: also
- `currency` – None
- `table_name` (*str*) – just a number, but str type

#### Extra

##### Variables

- `turn_pot` (*Decimal*) – pot size before turn
- `turn_num_players` (*int*) – number of players seen the turn
- `river_pot` (*Decimal*) – pot size before river
- `river_num_players` (*int*) – number of players seen the river
- `tournament_name` (*str*) – e.g. "\$750 Guarantee", "\$5 Sit & Go (Super Turbo) "

## 2.3.5 PKR

**class** `poker.room.pkr.PKRHandHistory` (*hand\_text*)

Parses PKR hand histories.

**Class specific**

**Variables** `table_name` (*str*) – “#table\_number - name\_of\_the\_table”

**Extra**

**Variables**

- `last_ident` (*str*) – last hand id
- `money_type` (*str*) – "R" for real money, "P" for play money

## 2.4 Room specific classes API

### 2.4.1 Pokerstars player notes

**class** `poker.room.pokerstars.Notes` (*notes*)

Class for parsing pokerstars XML notes.

**get\_note** (*player*)

Return `_Note` tuple for the player.

**Raises** `poker.room.pokerstars.NoteNotFoundError` –

**add\_note** (*player, text, label=None, update=None*)

Add a note to the xml. If update param is None, it will be the current time.

**Raises** `poker.room.pokerstars.LabelNotFoundError` – if there is no such label  
name

**del\_note** (*player*)

Delete a note by player name.

**Raises** `poker.room.pokerstars.NoteNotFoundError` –

**add\_label** (*name, color*)

Add a new label. It's id will automatically be calculated.

**append\_note** (*player, text*)

Append text to an already existing note.

**del\_label** (*name*)

Delete a label by name.

**classmethod** **from\_file** (*filename*)

Make an instance from a XML file.

**get\_label** (*name*)

Find the label by name.

**get\_note\_text** (*player*)

Return note text for the player.

**label\_names**

Tuple of label names.

**labels**

Tuple of labels.

**notes**

Tuple of notes..

**players**

Tuple of player names.

**prepend\_note** (*player, text*)

Prepend text to an already existing note.

**replace\_note** (*player, text*)

Replace note text with text. (Overwrites previous note!)

**save** (*filename*)

Save the note XML to a file.

**class** poker.room.pokerstars.\_Label (*id, color, name*)

Labels in Player notes.

**Variables**

- **id** (*str*) – numeric id for the label. None when no label ('-1' in XML)
- **color** (*str*) – color code for note
- **name** (*str*) – name of the note

**class** poker.room.pokerstars.\_Note (*player, label, update, text*)

Player note.

**Variables**

- **player** (*str*) – player name
- **label** (*str*) – Label name of note
- **update** (*datetime.datetime*) – when was the note last updated
- **text** (*str*) – Note text

**exception** poker.room.pokerstars.NoteNotFoundError

Note not found for player.

**exception** poker.room.pokerstars.LabelNotFoundError

Label not found in the player notes.

## 2.5 Website API

This package contains mostly scraping tools for well known websites like Two Plus Two forum, Pocketsfives, etc...

### 2.5.1 Two Plus Two Forum API

poker.website.twoplustwo.FORUM\_URL

<http://forumserver.twoplustwo.com>

poker.website.twoplustwo.FORUM\_MEMBER\_URL

<http://forumserver.twoplustwo.com/members>

```
class poker.website.twoplustwo.ForumMember(username)
```

Download and store a member data from the Two Plus Two forum.

**Parameters** `id` (*int*, *str*) –

Forum id (last part of members URL, e.g. in case of <http://forumserver.twoplustwo.com/members/407153/> the id is 407153)

**Variables**

- `id` (*str*) – Forum id
- `username` (*str*) – Forum username
- `rank` (*str*) – Forum rank like 'enthusiast '
- `profile_picture` (*str*, *None*) – URL of profile if set.
- `location` (*str*) – Location (country)
- `total_posts` (*int*) – Total posts
- `posts_per_day` (*float*) – Posts per day on account page
- `last_activity` (*datetime*) – Last activity with the website timezone
- `join_date` (*date*) – Join date on account page
- `public_usergroups` (*tuple*) – Public usergroup permission as in the box on the top right
- `download_date` (*datetime*) – When were the data downloaded from TwoplusTwo

## 2.5.2 Pocketfives API

```
class poker.website.pocketfives._Player(name, country, triple_crowns, monthly_win,  
biggest_cash, plb_score, biggest_score, average_score, previous_rank)
```

Pocketfives player data.

**Variables**

- `name` (*str*) – Player name
- `country` (*str*) – Country name
- `triple_crowns` (*int*) – Number of triple crowns won
- `monthly_win` (*int*) –
- `biggest_cash` (*str*) –
- `plb_score` (*float*) –
- `biggest_score` (*float*) – Biggest Pocketfives score
- `average_score` (*float*) – Average pocketfives score
- `previous_rank` (*str*) – Previous pocketfives rank

```
poker.website.pocketfives.get_ranked_players()
```

Get the list of the first 100 ranked players.

**Returns** generator of `_Players`



---

**Note:** Downloading this list is a slow operation!

---

### 2.5.3 PokerStars website API

`poker.website.pokerstars.WEBSITE_URL`  
<http://www.pokerstars.eu>

`poker.website.pokerstars.TOURNAMENTS_XML_URL`  
[http://www.pokerstars.eu/datafeed\\_global/tournaments/all.xml](http://www.pokerstars.eu/datafeed_global/tournaments/all.xml)

`poker.website.pokerstars.STATUS_URL`  
[http://www.psimg.com/datafeed/dyn\\_banners/summary.json.js](http://www.psimg.com/datafeed/dyn_banners/summary.json.js)

`poker.website.pokerstars.get_current_tournaments()`  
 Get the next 200 tournaments from pokerstars.

**Returns** generator of `_Tournament`

---

**Note:** Downloading this list is an extremely slow operation!

---

`poker.website.pokerstars.get_status()`  
 Get pokerstars status: players online, number of tables, etc.

**Returns** `_Status`

**class** `poker.website.pokerstars._Tournament` (*start\_date, name, game, buyin, players*)  
 Upcoming pokerstars tournament.

#### Variables

- **start\_date** (*datetime*) –
- **name** (*str*) – Tournament name as seen in PokerStars Lobby
- **game** (*str*) – Game Type
- **buyin** (*str*) – Buy in with fee
- **players** (*int*) – Number of players already registered

**class** `poker.website.pokerstars._Status` (*updated, tables, next\_update, players, clubs, active\_tournaments, total\_tournaments, sites, club\_members*)

PokerStars status.

#### Variables

- **updated** (*datetime*) – Status last updated
- **tables** (*int*) – Number of tournament tables
- **players** (*int*) – Number of players logged in to PokerStars
- **clubs** (*int*) – Total number of Home Game clubs created
- **club\_members** (*int*) – Total number of Home Game club members
- **active\_tournaments** (*int*) –
- **total\_tournaments** (*int*) –

- **sites** (*tuple*) – Tuple of *\_SiteStatus*

**class** `poker.website.pokerstars._SiteStatus` (*id, tables, players, active\_tournaments*)  
PokerStars status on different subsites like FR, ES IT or Play Money.

**Variables**

- **id** (*str*) – ID of the site ("*.IT*", "*.FR*", "*Play Money*")
- **tables** (*int*) –
- **player** (*int*) –
- **active\_tournaments** (*int*) –

## CHAPTER 3

---

### Repo and contact

---

Repo: <https://github.com/pokerregion/poker>  
Issues: <https://github.com/pokerregion/poker/issues>  
[@kissgyorgy](#) on twitter  
or you can reach me on my [public Github e-mail](#).



## CHAPTER 4

---

### Indices and tables

---

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



### p

- `poker.constants`, [21](#)
- `poker.website.pocketfives`, [28](#)
- `poker.website.pokerstars`, [29](#)
- `poker.website.twoplustwo`, [27](#)





## Symbols

[\\_BaseHandHistory](#) (class in [poker.handhistory](#)), 23  
[\\_Flop](#) (built-in class), 24  
[\\_Label](#) (class in [poker.room.pokerstars](#)), 27  
[\\_Note](#) (class in [poker.room.pokerstars](#)), 27  
[\\_Player](#) (class in [poker.handhistory](#)), 24  
[\\_Player](#) (class in [poker.website.pocketfives](#)), 28  
[\\_SiteStatus](#) (class in [poker.website.pokerstars](#)), 30  
[\\_Status](#) (class in [poker.website.pokerstars](#)), 29  
[\\_Tournament](#) (class in [poker.website.pokerstars](#)), 29

## A

[Action](#) (class in [poker.constants](#)), 21  
[ACTION](#) ([poker.constants.TourFormat](#) attribute), 22  
[add\\_label\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[add\\_note\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[append\\_note\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26

## B

[BB](#) ([poker.constants.Position](#) attribute), 22  
[BET](#) ([poker.constants.Action](#) attribute), 21  
[Broadway card](#), 15  
[BROADWAY\\_RANKS](#) (in module [poker.card](#)), 17  
[BTN](#) ([poker.constants.Position](#) attribute), 22

## C

[CALL](#) ([poker.constants.Action](#) attribute), 21  
[Card](#), 14  
[Card](#) (class in [poker.card](#)), 18  
[CASH](#) ([poker.constants.GameType](#) attribute), 22  
[CHECK](#) ([poker.constants.Action](#) attribute), 21  
[CO](#) ([poker.constants.Position](#) attribute), 22  
[Combo](#), 15  
[Combo](#) (class in [poker.hand](#)), 19  
[combos](#) ([poker.hand.Range](#) attribute), 20  
[Currency](#) (class in [poker.constants](#)), 21

## D

[del\\_label\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26

[del\\_note\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[difference\(\)](#) ([poker.card.Rank](#) class method), 17  
[DOUBLE](#) ([poker.constants.TourSpeed](#) attribute), 23

## E

[EIGHT](#) ([poker.constants.PokerRoom](#) attribute), 22  
[EUR](#) ([poker.constants.Currency](#) attribute), 21

## F

[Face card](#), 15  
[FACE\\_RANKS](#) (in module [poker.card](#)), 17  
[first](#) ([poker.hand.Combo](#) attribute), 19  
[first](#) ([poker.hand.Hand](#) attribute), 18  
[FL](#) ([poker.constants.Limit](#) attribute), 22  
[FOLD](#) ([poker.constants.Action](#) attribute), 21  
[FORUM\\_MEMBER\\_URL](#) (in module [poker.website.twoplustwo](#)), 27  
[FORUM\\_URL](#) (in module [poker.website.twoplustwo](#)), 27  
[ForumMember](#) (class in [poker.website.twoplustwo](#)), 27  
[from\\_cards\(\)](#) ([poker.hand.Combo](#) class method), 19  
[from\\_file\(\)](#) ([poker.hand.Range](#) class method), 20  
[from\\_file\(\)](#) ([poker.room.pokerstars.Notes](#) class method), 26  
[from\\_objects\(\)](#) ([poker.hand.Range](#) class method), 20  
[FTP](#) ([poker.constants.PokerRoom](#) attribute), 22  
[FullTiltPokerHandHistory](#) (class in [poker.room.fulltiltpoker](#)), 25

## G

[Game](#) (class in [poker.constants](#)), 21  
[GameType](#) (class in [poker.constants](#)), 21  
[GBP](#) ([poker.constants.Currency](#) attribute), 21  
[get\\_current\\_tournaments\(\)](#) (in module [poker.website.pokerstars](#)), 29  
[get\\_label\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[get\\_note\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[get\\_note\\_text\(\)](#) ([poker.room.pokerstars.Notes](#) method), 26  
[get\\_ranked\\_players\(\)](#) (in module [poker.website.pocketfives](#)), 28

get\_status() (in module poker.website.pokerstars), 29

## H

Hand, [14](#)

Hand (class in poker.hand), 18

Hand comparisons, [14](#)

hands (poker.hand.Range attribute), 20

HJ (poker.constants.Position attribute), 22

HOLDEM (poker.constants.Game attribute), 21

HYPER (poker.constants.TourSpeed attribute), 23

## I

is\_broadway (poker.card.Card attribute), 18

is\_broadway (poker.hand.Combo attribute), 19

is\_broadway (poker.hand.Hand attribute), 19

is\_connector (poker.hand.Combo attribute), 19

is\_connector (poker.hand.Hand attribute), 19

is\_face (poker.card.Card attribute), 18

is\_offsuit (poker.hand.Combo attribute), 19

is\_offsuit (poker.hand.Hand attribute), 19

is\_one\_gapper (poker.hand.Combo attribute), 19

is\_one\_gapper (poker.hand.Hand attribute), 19

is\_pair (poker.hand.Combo attribute), 19

is\_pair (poker.hand.Hand attribute), 19

is\_suited (poker.hand.Combo attribute), 19

is\_suited (poker.hand.Hand attribute), 19

is\_suited\_connector (poker.hand.Combo attribute), 19

is\_suited\_connector (poker.hand.Hand attribute), 19

is\_two\_gapper (poker.hand.Combo attribute), 19

is\_two\_gapper (poker.hand.Hand attribute), 19

## L

label\_names (poker.room.pokerstars.Notes attribute), 26

LabelNotFoundError, [27](#)

labels (poker.room.pokerstars.Notes attribute), 26

Limit (class in poker.constants), 22

## M

make\_random() (poker.card.Card class method), 18

MoneyType (class in poker.constants), 22

MUCK (poker.constants.Action attribute), 21

## N

NL (poker.constants.Limit attribute), 22

NoteNotFoundError, [27](#)

Notes (class in poker.room.pokerstars), 26

notes (poker.room.pokerstars.Notes attribute), 27

## O

OFFSUIT\_HANDS (in module poker.hand), 19

OHILO (poker.constants.Game attribute), 21

OMAHA (poker.constants.Game attribute), 21

ONEREB (poker.constants.TourFormat attribute), 22

## P

PAIR\_HANDS (in module poker.hand), 19

percent (poker.hand.Range attribute), 20

PKR (poker.constants.PokerRoom attribute), 22

PKRHandHistory (class in poker.room.pkr), 26

PL (poker.constants.Limit attribute), 22

PLAY (poker.constants.MoneyType attribute), 22

players (poker.room.pokerstars.Notes attribute), 27

poker.constants (module), 21

poker.website.pocketfives (module), 28

poker.website.pokerstars (module), 29

poker.website.twoplustwo (module), 27

PokerRoom (class in poker.constants), 22

PokerStarsHandHistory (class in poker.room.pokerstars),  
[25](#)

Position (class in poker.constants), 22

prepend\_note() (poker.room.pokerstars.Notes method),  
[27](#)

## R

RAISE (poker.constants.Action attribute), 21

Range, [15](#)

Range (class in poker.hand), 20

Range is "bigger" than another, [15](#)

Range length, [15](#)

Range percent, [15](#)

Range size, [15](#)

Rank, [14](#)

Rank (class in poker.card), 17

rank (poker.card.Card attribute), 18

rank\_difference (poker.hand.Combo attribute), 20

rank\_difference (poker.hand.Hand attribute), 18

RAZZ (poker.constants.Game attribute), 21

REAL (poker.constants.MoneyType attribute), 22

REBUY (poker.constants.TourFormat attribute), 22

REGULAR (poker.constants.TourSpeed attribute), 23

rep\_pieces (poker.hand.Range attribute), 20

replace\_note() (poker.room.pokerstars.Notes method), 27

RETURN (poker.constants.Action attribute), 21

## S

save() (poker.room.pokerstars.Notes method), 27

SB (poker.constants.Position attribute), 22

SECOND (poker.constants.TourFormat attribute), 22

second (poker.hand.Combo attribute), 19

second (poker.hand.Hand attribute), 18

Shape, [14](#)

Shape (class in poker.hand), 18

shape (poker.hand.Combo attribute), 19

shape (poker.hand.Hand attribute), 19

SHOW (poker.constants.Action attribute), 21

slots (poker.hand.Range attribute), 20

SLOW (poker.constants.TourSpeed attribute), 23

SNG (poker.constants.GameType attribute), 22

STARS (poker.constants.PokerRoom attribute), 22  
STARS\_COIN (poker.constants.Currency attribute), 21  
STATUS\_URL (in module poker.website.pokerstars), 29  
STUD (poker.constants.Game attribute), 21  
Suit, 14  
Suit (class in poker.card), 17  
suit (poker.card.Card attribute), 18  
SUITED\_HANDS (in module poker.hand), 19

## T

THINK (poker.constants.Action attribute), 21  
to\_ascii() (poker.hand.Range method), 20  
to\_combos() (poker.hand.Hand method), 19  
to\_hand() (poker.hand.Combo method), 20  
to\_html() (poker.hand.Range method), 20  
Token, 15  
TOUR (poker.constants.GameType attribute), 22  
TourFormat (class in poker.constants), 22  
TOURNAMENTS\_XML\_URL (in module  
poker.website.pokerstars), 29  
TourSpeed (class in poker.constants), 22  
TURBO (poker.constants.TourSpeed attribute), 23

## U

USD (poker.constants.Currency attribute), 21  
UTG (poker.constants.Position attribute), 22  
UTG1 (poker.constants.Position attribute), 22  
UTG2 (poker.constants.Position attribute), 22  
UTG3 (poker.constants.Position attribute), 22  
UTG4 (poker.constants.Position attribute), 22

## W

WEBSITE\_URL (in module poker.website.pokerstars),  
29  
WIN (poker.constants.Action attribute), 21