berserk Documentation

Release 0.10.0

Robert Grant

Apr 26, 2020

Contents

Python client for the Lichess API.

CHAPTER 1

berserk

Python client for the Lichess API.

- Free software: GNU General Public License v3
- Documentation: https://berserk.readthedocs.io.

1.1 Features

- handles JSON and PGN formats at user's discretion
- token auth session
- easy integration with OAuth2
- automatically converts time values to datetimes

1.2 Usage

You can use any `requests.Session`-like object as a session, including those from `requests_oauth`. A simple token session is included, as shown below:

```
import berserk

session = berserk.TokenSession(API_TOKEN)
client = berserk.Client(session=session)
```

Most if not all of the API is available:

```
client.account.get
client.account.get_email
client.account.get_preferences
client.account.get_kid_mode
client.account.set_kid_mode
client.account.upgrade_to_bot

client.users.get_puzzle_activity
client.users.get_realtime_statuses
client.users.get_all_top_10
client.users.get_leaderboard
client.users.get_public_data
client.users.get_activity_feed
client.users.get_by_id
client.users.get_by_team
client.users.get_live_streamers
client.users.get_users_followed
client.users.get_users_following
client.users.get_rating_history

client.teams.get_members
client.teams.join
client.teams.leave
client.teams.kick_member

client.games.export
client.games.export_by_player
client.games.export_multi
client.games.get_among_players
client.games.get_ongoing
client.games.get_tv_channels

client.challenges.create
client.challenges.create_ai
client.challenges.create_open
client.challenges.create_with_accept
client.challenges.accept
client.challenges.decline

client.board.stream_incoming_events
client.board.seek
client.board.stream_game_state
client.board.make_move
client.board.post_message
client.board.abort_game
client.board.resign_game
client.board.handle_draw_offer
client.board.offer_draw
client.board.accept_draw
client.board.decline_draw

client.bots.stream_incoming_events
client.bots.stream_game_state
client.bots.make_move
client.bots.post_message
client.bots.abort_game
```

(continues on next page)

(continued from previous page)

```
client.bots.resign_game
client.bots.accept_challenge
client.bots.decline_challenge

client.tournaments.get
client.tournaments.create
client.tournaments.export_games
client.tournaments.stream_results
client.tournaments.stream_by_creator

client.broadcasts.create
client.broadcasts.get
client.broadcasts.update
client.broadcasts.push_pgn_update

client.simuls.get

client.studies.export_chapter
client.studies.export
```

Details for each function can be found in the full documentation.

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install berserk, run this command in your terminal:

```
$ pip install berserk
```

This is the preferred method to install berserk, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for berserk can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/rhgrant10/berserk
```

Or download the tarball:

```
$ curl  -OL https://github.com/rhgrant10/berserk/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

Use `berserk` by creating an API client:

```
>>> import berserk
>>> client = berserk.Client()
```

3.1 Authenticating

By default the client does not perform any authentication. However many of the endpoints are not open. To use a form of authentication, just pass the appropriate `requests.Session`-like object:

- using an API token: `berserk.TokenSession`
- using oauth: `requests_oauthlib.Oauth2Session`

Note: Some endpoints require specific Oauth2 permissions.

3.1.1 Using an API token

If you have a personal API token, you can simply use the `TokenSession` provided. For example, assuming you have written your token to `'./lichess.token'`:

```
>>> with open('./lichess.token') as f:
...     token = f.read()
...
>>> session = berserk.TokenSession(token)
>>> client = berserk.Client(session)
```

3.1.2 Using Oauth2

Some of the endpoints require OAuth2 authentication. Although outside the scope of this documentation, you can use `requests_oauthlib.Oauth2Session` for this.

```
>>> from requests_oauthlib import OAuth2Session
>>> session = OAuth2Session(...)
>>> client = berserk.Client(session)
```

3.2 Accounts

3.2.1 Information and Preferences

```
>>> client.account.get()
{'blocking': False,
 'count': {...},
 'createdAt': datetime.datetime(2018, 5, 16, 8, 9, 18, 187000),
 'followable': True,
 'following': False,
 'followsYou': False,
 'id': 'rhgrant10',
 'nbFollowers': 1,
 'nbFollowing': 1,
 'online': True,
 'perfs': {...},
 'playTime': {...},
 'seenAt': datetime.datetime(2018, 12, 9, 10, 28, 30, 221000),
 'url': 'https://lichess.org/@/rhgrant10',
 'username': 'rhgrant10'}

>>> client.account.get_email()
'rhgrant10@gmail.com'

>>> client.account.get_preferences()
{'animation': 2,
 'autoQueen': 1,
 ...
 'transp': False,
 'zen': 0}}
```

3.2.2 Kid Mode

Using Oauth2, you can set the kid mode.

```
>>> client.account.set_kid_mode(True)   # enable
True
>>> client.account.set_kid_mode(False)   # disable
True
```

Note that the `set_kid_mode` method returns an indicator of success and *not* the current or previous status.

```
>>> def show_kid_mode():
...     is_enabled = client.account.get_kid_mode()
...     print('enabled' if is_enabled else 'disabled')
...
>>> show_kid_mode()
disabled

>>> # try to enable, but the request fails
>>> client.account.set_kid_mode(True)
False
>>> show_kid_mode()
disabled

>>> # try again, this time it succeeds
>>> client.account.set_kid_mode(True)
True
>>> show_kid_mode()
enabled
```

3.2.3 Bot Account Upgrade

If this is a new account that has not yet played a game, and if you have the required OAuth2 permission, you can upgrade the account to a bot account:

```
>>> client.account.upgrade_to_bot()
```

Read more below about how to use bot functionality.

3.3 Users and Teams

3.3.1 Realtime Statuses

Get realtime information about one or more players:

```
>>> players = ['Sasageyo', 'Voinikonis_Nikita', 'Zugzwangerz', 'DOES-NOT-EXIST']
>>> client.users.get_realtime_statuses(players)
[{'id': 'sasageyo',
'name': 'Sasageyo',
'title': 'IM',
'online': True,
'playing': True},
{'id': 'voinikonis_nikita',
'name': 'Voinikonis_Nikita',
'title': 'FM',
'online': True,
'playing': True},
{'id': 'zugzwangerz', 'name': 'Zugzwangerz'}]
```

3.3.2 Top 10 Lists

```
>>> top10 = client.users.get_all_top_10()
>>> list(top10)
['bullet',
 'blitz',
 'rapid',
 'classical',
 'ultraBullet',
 'crazyhouse',
 'chess960',
 'kingOfTheHill',
 'threeCheck',
 'antichess',
 'atomic',
 'horde',
 'racingKings']
>>> top10['horde'][0]
{'id': 'ingrid-vengeance',
 'perfs': {'horde': {'progress': 22, 'rating': 2443}},
 'username': 'Ingrid-Vengeance'}
```

3.3.3 Leaderboards

```
>>> client.users.get_leaderboard('horde', count=11)[-1]
{'id': 'philippesaner',
 'perfs': {'horde': {'progress': 10, 'rating': 2230}},
 'username': 'PhilippeSaner'}
```

3.3.4 Public Data

```
>>> client.users.get_public_data('PhilippeSaner')
{'completionRate': 87,
 'count': {...},
 'createdAt': datetime.datetime(2017, 1, 9, 16, 14, 31, 140000),
 'id': 'philippesaner',
 'nbFollowers': 40,
 'nbFollowing': 13,
 'online': False,
 'perfs': {...},
 'playTime': {'total': 1505020, 'tv': 1038007},
 'profile': {'country': 'CA', 'location': 'Ottawa'},
 'seenAt': datetime.datetime(2018, 12, 9, 10, 26, 28, 22000),
 'url': 'https://lichess.org/@/PhilippeSaner',
 'username': 'PhilippeSaner'}
```

3.3.5 Activity Feeds

```
>>> feed = client.users.get_activity_feed('PhilippeSaner')
>>> feed[0]
{'games': {'horde': {'draw': 0,
```

(continues on next page)

(continued from previous page)

```
   'loss': 1,
   'rp': {'after': 2230, 'before': 2198},
   'win': 12}},
 'interval': {'end': datetime.datetime(2018, 12, 9, 16, 0),
  'start': datetime.datetime(2018, 12, 8, 16, 0)},
 'tournaments': {'best': [{'nbGames': 1,
    'rank': 6,
    'rankPercent': 33,
    'score': 2,
    'tournament': {'id': '9zm2uIdP', 'name': 'Daily Horde Arena'}}],
  'nb': 1}}
```

3.3.6 Team Members

```
>>> client.users.get_by_team('coders')
<map at 0x107c1acc0>
>>> members = list(_)
>>> len(members)
228
```

3.3.7 Live Streamers

```
>>> client.users.get_live_streamers()
[{'id': 'chesspatzerwal', 'name': 'ChesspatzerWAL', 'patron': True},
 {'id': 'ayrtontwigg', 'name': 'AyrtonTwigg', 'playing': True},
 {'id': 'fanatikchess', 'name': 'FanatikChess', 'patron': True},
 {'id': 'jwizzy74', 'name': 'Jwizzy74', 'patron': True, 'playing': True},
 {'id': 'devjamesb', 'name': 'DevJamesB', 'playing': True},
 {'id': 'kafka4x', 'name': 'Kafka4x', 'playing': True},
 {'id': 'sparklehorse', 'name': 'Sparklehorse', 'patron': True, 'title': 'IM'},
 {'id': 'ivarcode', 'name': 'ivarcode', 'playing': True},
 {'id': 'pepellou', 'name': 'pepellou', 'patron': True, 'playing': True},
 {'id': 'videogamepianist', 'name': 'VideoGamePianist', 'playing': True}]
```

3.4 Exporting Games

3.4.1 By Player

Finished games can be exported and current games can be listed. Let's take a look at the most recent 300 games played by "LeelaChess" on Dec. 8th, 2018:

```
>>> start = berserk.utils.to_millis(datetime(2018, 12, 8))
>>> end = berserk.utils.to_millis(datetime(2018, 12, 9))
>>> client.games.export_by_player('LeelaChess', since=start, until=end,
                                  max=300))
<generator object Games.export_by_player at 0x10c24b048>
>>> games = list(_)
>>> games[0]['createdAt']
datetime.datetime(2018, 12, 9, 22, 54, 24, 195000, tzinfo=datetime.timezone.utc)
```

(continues on next page)

(continued from previous page)

```
>>> games[-1]['createdAt']
datetime.datetime(2018, 12, 8, 9, 11, 42, 229000, tzinfo=datetime.timezone.utc)
```

Wow, they play a lot of chess :)

3.4.2 By ID

You can export games too using their IDs. Let's export the last game LeelaChess played that day:

```
>>> game_id = games[0]['id']
>>> client.games.export(game_id)
{'analysis': [...],
 'clock': {'increment': 8, 'initial': 300, 'totalTime': 620},
 'createdAt': datetime.datetime(2018, 12, 9, 22, 54, 24, 195000, tzinfo=datetime.
↪timezone.utc),
 'id': 'WatQhhbJ',
 'lastMoveAt': datetime.datetime(2018, 12, 9, 23, 5, 59, 396000, tzinfo=datetime.
↪timezone.utc),
 'moves': ...
 'opening': {'eco': 'D38',
  'name': "Queen's Gambit Declined: Ragozin Defense",
  'ply': 8},
 'perf': 'rapid',
 'players': {'black': {'analysis': {'acpl': 44,
    'blunder': 1,
    'inaccuracy': 4,
    'mistake': 2},
   'rating': 1333,
   'ratingDiff': 0,
   'user': {'id': 'fsoto', 'name': 'fsoto'}},
  'white': {'analysis': {'acpl': 11,
    'blunder': 0,
    'inaccuracy': 2,
    'mistake': 0},
   'provisional': True,
   'rating': 2490,
   'ratingDiff': 0,
   'user': {'id': 'leelachess', 'name': 'LeelaChess', 'title': 'BOT'}}},
 'rated': True,
 'speed': 'rapid',
 'status': 'mate',
 'variant': 'standard',
 'winner': 'white'}
```

3.4.3 PGN vs JSON

Of course sometimes PGN format is desirable. Just pass `as_pgn=True` to any of the export methods:

```
>>> pgn = client.games.export(game_id, as_pgn=True)
>>> print(pgn)
[Event "Rated Rapid game"]
[Site "https://lichess.org/WatQhhbJ"]
[Date "2018.12.09"]
[Round "-"]
```

(continues on next page)

(continued from previous page)

```
[White "LeelaChess"]
[Black "fsoto"]
[Result "1-0"]
[UTCDate "2018.12.09"]
[UTCTime "22:54:24"]
[WhiteElo "2490"]
[BlackElo "1333"]
[WhiteRatingDiff "+0"]
[BlackRatingDiff "+0"]
[WhiteTitle "BOT"]
[Variant "Standard"]
[TimeControl "300+8"]
[ECO "D38"]
[Opening "Queen's Gambit Declined: Ragozin Defense"]
[Termination "Normal"]

1. d4 { [%eval 0.08] [%clk 0:05:00] } 1... d5 ...
```

3.4.4 TV Channels

```
>>> channels = client.games.get_tv_channels()
>>> list(channels)
['Bot',
 'Blitz',
 'Racing Kings',
 'UltraBullet',
 'Bullet',
 'Classical',
 'Three-check',
 'Antichess',
 'Computer',
 'Horde',
 'Rapid',
 'Atomic',
 'Crazyhouse',
 'Chess960',
 'King of the Hill',
 'Top Rated']
>>> channels['King of the Hill']
{'gameId': 'YPL6tP2K',
 'rating': 1554,
 'user': {'id': 'linischoki', 'name': 'linischoki'}}
```

3.5 Working with tournaments

You have to specify the clock time, increment, and minutes, but creating a new tournament is easy:

```
>>> client.tournaments.create(clock_time=10, clock_increment=3, minutes=180)
{'berserkable': True,
 'clock': {'increment': 3, 'limit': 600},
 'createdBy': 'rhgrant10',
 'duels': [],
```

(continues on next page)

(continued from previous page)

```
 'fullName': "O'Kelly Arena",
 'greatPlayer': {'name': "O'Kelly",
  'url': "https://wikipedia.org/wiki/Alb%C3%A9ric_O'Kelly_de_Galway"},
 'id': '3uwyXjiC',
 'minutes': 180,
 'nbPlayers': 0,
 'perf': {'icon': '#', 'name': 'Rapid'},
 'quote': {'author': 'Bent Larsen',
  'text': 'I often play a move I know how to refute.'},
 'secondsToStart': 300,
 'standing': {'page': 1, 'players': []},
 'startsAt': '2018-12-10T00:32:12.116Z',
 'system': 'arena',
 'variant': 'standard',
 'verdicts': {'accepted': True, 'list': []}}
```

You can specify the starting position for new tournaments using one of the provided enum value in `berserk.enums.Position`:

```
>>> client.tournaments.create(clock_time=10, clock_increment=3, minutes=180,
                              position=berserk.enums.Position.KINGS_PAWN)
```

Additionally you can see tournaments that have recently finished, are in progress, and are about to start:

```
>>> tournaments = client.tournaments.get()
>>> list(tournaments)
['created', 'started', 'finished']
>>> len(tournaments['created'])
19
>>> tournaments['created'][0]
{'clock': {'increment': 0, 'limit': 300},
 'createdBy': 'bashkimneziri',
 'finishesAt': datetime.datetime(2018, 12, 24, 0, 21, 2, 179000, tzinfo=datetime.
↪timezone.utc),
 'fullName': 'GM Arena',
 'id': 'COnVgmKH',
 'minutes': 45,
 'nbPlayers': 1,
 'perf': {'icon': ')', 'key': 'blitz', 'name': 'Blitz', 'position': 1},
 'rated': True,
 'secondsToStart': 160,
 'startsAt': datetime.datetime(2018, 12, 23, 23, 36, 2, 179000, tzinfo=datetime.
↪timezone.utc),
 'status': 10,
 'system': 'arena',
 'variant': {'key': 'standard', 'name': 'Standard', 'short': 'Std'},
 'winner': None}
```

3.6 Being a bot

Warning: These commands only work using bot accounts. Make sure you have converted the account with which you authenticate into a bot account first. See above for details.

Bots stream game information and react by calling various endpoints. There are two streams of information:

1. incoming events
2. state of a particular game

In general, a bot will listen to the stream of incoming events, determine which challenges to accept, and once accepted, listen to the stream of game states and respond with the best moves in an attempt to win as many games as possible. You *can* create a bot that looses intentionally if that makes you happy, but regardless you will need to listen to both streams of information.

The typical pattern is to have one main thread that listens to the event stream and spawns new threads when accepting challenges. Each challenge thread then listens to the stream of state for that particular game and plays it to completion.

3.6.1 Responding to challenges

Here the goal is to respond to challenges and spawn workers to play those accepted. Here's a bit of sample code that hits the highlights:

```
>>> is_polite = True
>>> for event in client.bots.stream_incoming_events():
...     if event['type'] == 'challenge':
...         if should_accept(event):
...             client.bots.accept_challenge(event['id'])
...         elif is_polite:
...             client.bots.decline_challenge(event['id'])
...     elif event['type'] == 'gameStart':
...         game = Game(event['id'])
...         game.start()
...
```

3.6.2 Playing a game

Having accepted a challenge and recieved the gameStart event for it, the main job here is to listen and react to the stream of the game state:

```
>>> class Game(threading.Thread):
...     def __init__(self, client, game_id, **kwargs):
...         super().__init__(**kwargs)
...         self.game_id = game_id
...         self.client = client
...         self.stream = client.bots.stream_game_state(game_id)
...         self.current_state = next(self.stream)
...
...     def run(self):
...         for event in self.stream:
...             if event['type'] == 'gameState':
...                 self.handle_state_change(event)
...             elif event['type'] == 'chatLine':
...                 self.handle_chat_line(event)
...
...     def handle_state_change(self, game_state):
...         pass
...
...     def handle_chat_line(self, chat_line):
```

(continues on next page)

(continued from previous page)

```
...         pass
...
```

Obviously the code above is just to communicate the gist of what is required. But once you have your framework for reacting to changes in game state, there are a variety of actions you can take:

```
>>> client.bots.make_move(game_id, 'e2e4')
True
>>> client.bots.abort_game(game_id)
True
>>> client.bots.resign_game(game_id)
True
>>> client.bots.post_message(game_id, 'Prepare to loose')
True
```

CHAPTER 4

Developer Interface

4.1 Clients

class berserk.clients.**Client**(*session=None*, *base_url=None*, *pgn_as_default=False*)

Bases: berserk.clients.BaseClient

Main touchpoint for the API.

All endpoints are namespaced into the clients below:

- *account* - managing account information
- *bots* - performing bot operations
- *broadcasts* - getting and creating broadcasts
- *challenges* - using challenges
- *games* - getting and exporting games
- *simuls* - getting simultaneous exhibition games
- *studies* - exporting studies
- *teams* - getting information about teams
- *tournaments* - getting and creating tournaments
- *users* - getting information about users

Parameters

- **session** (requests.Session) – request session, authenticated as needed
- **base_url** (*str*) – base API URL to use (if other than the default)
- **pgn_as_default** (*bool*) – True if PGN should be the default format for game exports when possible. This defaults to False and is used as a fallback when as_pgn is left as None for methods that support it.

class berserk.clients.**Account**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Client for account-related endpoints.

get()
Get your public information.

Returns public information about the authenticated user

Return type dict

get_email()
Get your email address.

Returns email address of the authenticated user

Return type str

get_kid_mode()
Get your kid mode status.

Returns current kid mode status

Return type bool

get_preferences()
Get your account preferences.

Returns preferences of the authenticated user

Return type dict

set_kid_mode(*value*)
Set your kid mode status.

Parameters **value** (*bool*) – whether to enable or disable kid mode

Returns success

Return type bool

upgrade_to_bot()
Upgrade your account to a bot account.

Requires bot:play oauth scope. User cannot have any previously played games.

Returns success

Return type bool

class berserk.clients.**Board**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Client for physical board or external application endpoints.

abort_game(*game_id*)
Abort a board game.

Parameters **game_id** (*str*) – ID of a game

Returns success

Return type bool

accept_draw(*game_id*)
Accept an already offered draw in the given game.

Parameters **game_id** (*str*) – ID of an in-progress game

Returns True if successful

Return type bool

decline_draw(*game_id*)
Decline an already offered draw in the given game.

Parameters **game_id** (*str*) – ID of an in-progress game

Returns True if successful

Return type bool

handle_draw_offer(*game_id*, *accept*)
Create, accept, or decline a draw offer.

To offer a draw, pass `accept=True` and a game ID of an in-progress game. To response to a draw offer, pass either `accept=True` or `accept=False` and the ID of a game in which you have recieved a draw offer.

Often, it's easier to call *offer_draw()*, *accept_draw()*, or *decline_draw()*.

Parameters

- **game_id** (*str*) – ID of an in-progress game
- **accept** (*bool*) – whether to accept

Returns True if successful

Return type bool

make_move(*game_id*, *move*)
Make a move in a board game.

Parameters

- **game_id** (*str*) – ID of a game
- **move** (*str*) – move to make

Returns success

Return type bool

offer_draw(*game_id*)
Offer a draw in the given game.

Parameters **game_id** (*str*) – ID of an in-progress game

Returns True if successful

Return type bool

post_message(*game_id*, *text*, *spectator=False*)
Post a message in a board game.

Parameters

- **game_id** (*str*) – ID of a game
- **text** (*str*) – text of the message
- **spectator** (*bool*) – post to spectator room (else player room)

Returns success

Return type bool

resign_game(*game_id*)
Resign a board game.

Parameters **game_id** (*str*) – ID of a game

Returns success

Return type bool

seek(*time*, *increment*, *rated=False*, *variant='standard'*, *color='random'*, *rating_range=None*)
Create a public seek to start a game with a random opponent.

Parameters

- **time** (*int*) – intial clock time in minutes
- **increment** (*int*) – clock increment in minutes
- **rated** (*bool*) – whether the game is rated (impacts ratings)
- **variant** (*str*) – game variant to use
- **color** (*str*) – color to play
- **rating_range** – range of opponent ratings

Returns duration of the seek

Return type float

stream_game_state(*game_id*)
Get the stream of events for a board game.

Parameters **game_id** (*str*) – ID of a game

Returns iterator over game states

stream_incoming_events()
Get your realtime stream of incoming events.

Returns stream of incoming events

Return type iterator over the stream of events

class berserk.clients.**Bots**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Client for bot-related endpoints.

abort_game(*game_id*)
Abort a bot game.

Parameters **game_id** (*str*) – ID of a game

Returns success

Return type bool

accept_challenge(*challenge_id*)
Accept an incoming challenge.

Parameters **challenge_id** (*str*) – ID of a challenge

Returns success

Return type bool

decline_challenge(*challenge_id*)
Decline an incoming challenge.

Parameters **challenge_id** (*str*) – ID of a challenge

Returns success

Return type bool

make_move(*game_id*, *move*)
Make a move in a bot game.

Parameters

- **game_id** (*str*) – ID of a game
- **move** (*str*) – move to make

Returns success

Return type bool

post_message(*game_id*, *text*, *spectator=False*)
Post a message in a bot game.

Parameters

- **game_id** (*str*) – ID of a game
- **text** (*str*) – text of the message
- **spectator** (*bool*) – post to spectator room (else player room)

Returns success

Return type bool

resign_game(*game_id*)
Resign a bot game.

Parameters **game_id** (*str*) – ID of a game

Returns success

Return type bool

stream_game_state(*game_id*)
Get the stream of events for a bot game.

Parameters **game_id** (*str*) – ID of a game

Returns iterator over game states

stream_incoming_events()
Get your realtime stream of incoming events.

Returns stream of incoming events

Return type iterator over the stream of events

class berserk.clients.**Broadcasts**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Broadcast of one or more games.

create(*name*, *description*, *sync_url=None*, *markdown=None*, *credit=None*, *starts_at=None*, *official=None*, *throttle=None*)
Create a new broadcast.

Note: `sync_url` must be publicly accessible. If not provided, you must periodically push new PGN to update the broadcast manually.

Parameters

- **name** (*str*) – name of the broadcast
- **description** (*str*) – short description
- **markdown** (*str*) – long description
- **sync_url** (*str*) – URL by which Lichess can poll for updates
- **credit** (*str*) – short text to give credit to the source provider
- **starts_at** (*int*) – start time as millis
- **official** (*bool*) – DO NOT USE
- **throttle** (*int*) – DO NOT USE

Returns created tournament info

Return type dict

get(*broadcast_id*, *slug='-'*)
Get a broadcast by ID.

Parameters

- **broadcast_id** (*str*) – ID of a broadcast
- **slug** (*str*) – slug for SEO

Returns broadcast information

Return type dict

push_pgn_update(*broadcast_id*, *pgn_games*, *slug='-'*)
Manually update an existing broadcast by ID.

Parameters

- **broadcast_id** (*str*) – ID of a broadcast
- **pgn_games** (*list*) – one or more games in PGN format

Returns success

Return type bool

update(*broadcast_id*, *name*, *description*, *sync_url*, *markdown=None*, *credit=None*, *starts_at=None*, *official=None*, *throttle=None*, *slug='-'*)
Update an existing broadcast by ID.

Note: Provide all fields. Values in missing fields will be erased.

Parameters

- **broadcast_id** (*str*) – ID of a broadcast
- **name** (*str*) – name of the broadcast

- **description** (*str*) – short description
- **sync_url** (*str*) – URL by which Lichess can poll for updates
- **markdown** (*str*) – long description
- **credit** (*str*) – short text to give credit to the source provider
- **starts_at** (*int*) – start time as millis
- **official** (*bool*) – DO NOT USE
- **throttle** (*int*) – DO NOT USE
- **slug** (*str*) – slug for SEO

Returns updated broadcast information

Return type dict

class berserk.clients.**Challenges**(*session*, *base_url=None*)

Bases: berserk.clients.BaseClient

accept(*challenge_id*)

Accept an incoming challenge.

Parameters **challenge_id** (*str*) – id of the challenge to accept

Returns success indicator

Return type bool

create(*username*, *rated*, *clock_limit=None*, *clock_increment=None*, *days=None*, *color=None*, *variant=None*, *position=None*)

Challenge another player to a game.

Parameters

- **username** (*str*) – username of the player to challege
- **rated** (*bool*) – whether or not the game will be rated
- **clock_limit** (*int*) – clock initial time (in seconds)
- **clock_increment** (*int*) – clock increment (in seconds)
- **days** (*int*) – days per move (for correspondence games; omit clock)
- **color** (*Color*) – color of the accepting player
- **variant** (*Variant*) – game variant to use
- **position** (*str*) – custom intial position in FEN (variant must be standard and the game cannot be rated)

Returns challenge data

Return type dict

create_ai(*level=8*, *clock_limit=None*, *clock_increment=None*, *days=None*, *color=None*, *variant=None*, *position=None*)

Challenge AI to a game.

Parameters

- **level** (*int*) – level of the AI (1 to 8)
- **clock_limit** (*int*) – clock initial time (in seconds)

- **clock_increment** (*int*) – clock increment (in seconds)
- **days** (*int*) – days per move (for correspondence games; omit clock)
- **color** (*Color*) – color of the accepting player
- **variant** (*Variant*) – game variant to use
- **position** (*str*) – use one of the custom initial positions (variant must be standard and cannot be rated)

Returns success indicator

Return type bool

create_open(*clock_limit=None*, *clock_increment=None*, *variant=None*, *position=None*)
Create a challenge that any two players can join.

Parameters

- **clock_limit** (*int*) – clock initial time (in seconds)
- **clock_increment** (*int*) – clock increment (in seconds)
- **variant** (*Variant*) – game variant to use
- **position** (*str*) – custom intial position in FEN (variant must be standard and the game cannot be rated)

Returns challenge data

Return type dict

create_with_accept(*username*, *rated*, *token*, *clock_limit=None*, *clock_increment=None*, *days=None*, *color=None*, *variant=None*, *position=None*)
Start a game with another player.

This is just like the regular challenge create except it forces the opponent to accept. You must provide the OAuth token of the opponent and it must have the challenge:write scope.

Parameters

- **username** (*str*) – username of the opponent
- **rated** (*bool*) – whether or not the game will be rated
- **token** (*str*) – opponent's OAuth token
- **clock_limit** (*int*) – clock initial time (in seconds)
- **clock_increment** (*int*) – clock increment (in seconds)
- **days** (*int*) – days per move (for correspondence games; omit clock)
- **color** (*Color*) – color of the accepting player
- **variant** (*Variant*) – game variant to use
- **position** (*Position*) – custom intial position in FEN (variant must be standard and the game cannot be rated)

Returns game data

Return type dict

decline(*challenge_id*)
Decline an incoming challenge.

Parameters **challenge_id** (*str*) – id of the challenge to decline

Returns success indicator

Return type bool

class berserk.clients.**Games**(*session*, *base_url=None*, *pgn_as_default=False*)
Bases: berserk.clients.FmtClient

Client for games-related endpoints.

export(*game_id*, *as_pgn=None*, *moves=None*, *tags=None*, *clocks=None*, *evals=None*, *opening=None*, *literate=None*)
Get one finished game as PGN or JSON.

Parameters

- **game_id** (*str*) – the ID of the game to export
- **as_pgn** (*bool*) – whether to return the game in PGN format
- **moves** (*bool*) – whether to include the PGN moves
- **tags** (*bool*) – whether to include the PGN tags
- **clocks** (*bool*) – whether to include clock comments in the PGN moves
- **evals** (*bool*) – whether to include analysis evaluation comments in the PGN moves when available
- **opening** (*bool*) – whether to include the opening name
- **literate** (*bool*) – whether to include literate the PGN

Returns exported game, as JSON or PGN

export_by_player(*username*, *as_pgn=None*, *since=None*, *until=None*, *max=None*, *vs=None*, *rated=None*, *perf_type=None*, *color=None*, *analysed=None*, *moves=None*, *tags=None*, *evals=None*, *opening=None*)
Get games by player.

Parameters

- **username** (*str*) – which player's games to return
- **as_pgn** (*bool*) – whether to return the game in PGN format
- **since** (*int*) – lowerbound on the game timestamp
- **until** (*int*) – upperbound on the game timestamp
- **max** (*int*) – limit the number of games returned
- **vs** (*str*) – filter by username of the opponent
- **rated** (*bool*) – filter by game mode (`True` for rated, `False` for casual)
- **perf_type** (*PerfType*) – filter by speed or variant
- **color** (*Color*) – filter by the color of the player
- **analysed** (*bool*) – filter by analysis availability
- **moves** (*bool*) – whether to include the PGN moves
- **tags** (*bool*) – whether to include the PGN tags
- **clocks** (*bool*) – whether to include clock comments in the PGN moves
- **evals** (*bool*) – whether to include analysis evaluation comments in the PGN moves when available

- **opening** (*bool*) – whether to include the opening name
- **literate** (*bool*) – whether to include literate the PGN

Returns iterator over the exported games, as JSON or PGN

export_multi(**game_ids*, *as_pgn=None*, *moves=None*, *tags=None*, *clocks=None*, *evals=None*, *opening=None*)

Get multiple games by ID.

Parameters

- **game_ids** – one or more game IDs to export
- **as_pgn** (*bool*) – whether to return the game in PGN format
- **moves** (*bool*) – whether to include the PGN moves
- **tags** (*bool*) – whether to include the PGN tags
- **clocks** (*bool*) – whether to include clock comments in the PGN moves
- **evals** (*bool*) – whether to include analysis evaluation comments in the PGN moves when available
- **opening** (*bool*) – whether to include the opening name

Returns iterator over the exported games, as JSON or PGN

get_among_players(**usernames*)

Get the games currently being played among players.

Note this will not includes games where only one player is in the given list of usernames.

Parameters **usernames** – two or more usernames

Returns iterator over all games played among the given players

get_ongoing(*count=10*)

Get your currently ongoing games.

Parameters **count** (*int*) – number of games to get

Returns some number of currently ongoing games

Return type list

get_tv_channels()

Get basic information about the best games being played.

Returns best ongoing games in each speed and variant

Return type dict

class berserk.clients.**Simuls**(*session*, *base_url=None*)

Bases: berserk.clients.BaseClient

Simultaneous exhibitions - one vs many.

get()

Get recently finished, ongoing, and upcoming simuls.

Returns current simuls

Return type list

class berserk.clients.**Studies**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Study chess the Lichess way.

export(*study_id*)
Export all chapters of a study.

Returns all chapters as PGN

Return type list

export_chapter(*study_id*, *chapter_id*)
Export one chapter of a study.

Returns chapter

Return type PGN

class berserk.clients.**Teams**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

get_members(*team_id*)
Get members of a team.

Parameters **team_id** (*str*) – ID of a team

Returns users on the given team

Return type iter

join(*team_id*)
Join a team.

Parameters **team_id** (*str*) – ID of a team

Returns success

Return type bool

kick_member(*team_id*, *user_id*)
Kick a member out of your team.

Parameters

- **team_id** (*str*) – ID of a team
- **user_id** (*str*) – ID of a team member

Returns success

Return type bool

leave(*team_id*)
Leave a team.

Parameters **team_id** (*str*) – ID of a team

Returns success

Return type bool

class berserk.clients.**Tournaments**(*session*, *base_url=None*, *pgn_as_default=False*)
Bases: berserk.clients.FmtClient

Client for tournament-related endpoints.

create(*clock_time*, *clock_increment*, *minutes*, *name=None*, *wait_minutes=None*, *variant=None*, *berserkable=None*, *rated=None*, *start_date=None*, *position=None*, *password=None*, *conditions=None*)

Create a new tournament.

Note: `wait_minutes` is always relative to now and is overriden by `start_time`.

Note: If `name` is left blank then one is automatically created.

Parameters

- **clock_time** (*int*) – intial clock time in minutes
- **clock_increment** (*int*) – clock increment in seconds
- **minutes** (*int*) – length of the tournament in minutes
- **name** (*str*) – tournament name
- **wait_minutes** (*int*) – future start time in minutes
- **start_date** (*str*) – when to start the tournament
- **variant** (*str*) – variant to use if other than standard
- **rated** (*bool*) – whether the game affects player ratings
- **berserkable** (*str*) – whether players can use berserk
- **position** (*str*) – custom initial position in FEN
- **password** (*str*) – password (makes the tournament private)
- **conditions** (*dict*) – conditions for participation

Returns created tournament info

Return type dict

export_games(*id_*, *as_pgn=False*, *moves=None*, *tags=None*, *clocks=None*, *evals=None*, *opening=None*)

Export games from a tournament.

Parameters

- **id** (*str*) – tournament ID
- **as_pgn** (*bool*) – whether to return PGN instead of JSON
- **moves** (*bool*) – include moves
- **tags** (*bool*) – include tags
- **clocks** (*bool*) – include clock comments in the PGN moves, when available
- **evals** (*bool*) – include analysis evalulation comments in the PGN moves, when available
- **opening** (*bool*) – include the opening name

Returns games

Return type list

get()
Get recently finished, ongoing, and upcoming tournaments.

Returns current tournaments

Return type list

stream_by_creator(*username*)
Stream the tournaments created by a player.

Parameters **username** (*str*) – username of the player

Returns tournaments

Return type iter

stream_results(*id_*, *limit=None*)
Stream the results of a tournament.

Results are the players of a tournament with their scores and performance in rank order. Note that results for ongoing tournaments can be inconsistent due to ranking changes.

Parameters

- **id** (*str*) – tournament ID
- **limit** (*int*) – maximum number of results to stream

Returns iterator over the stream of results

Return type iter

class berserk.clients.**Users**(*session*, *base_url=None*)
Bases: berserk.clients.BaseClient

Client for user-related endpoints.

get_activity_feed(*username*)
Get the activity feed of a user.

Parameters **username** (*str*) – username

Returns activity feed of the given user

Return type list

get_all_top_10()
Get the top 10 players for each speed and variant.

Returns top 10 players in each speed and variant

Return type dict

get_by_id(**usernames*)
Get multiple users by their IDs.

Parameters **usernames** – one or more usernames

Returns user data for the given usernames

Return type list

get_by_team(*team_id*)
Get members of a team.

Parameters **team_id** (*str*) – ID of a team

Returns users on the given team

Return type iter

get_leaderboard(*perf_type*, *count=10*)
Get the leaderboard for one speed or variant.

Parameters

- **perf_type** (*PerfType*) – speed or variant
- **count** (*int*) – number of players to get

Returns top players for one speed or variant

Return type list

get_live_streamers()
Get basic information about currently streaming users.

Returns users currently streaming a game

Return type list

get_public_data(*username*)
Get the public data for a user.

Parameters **username** (*str*) – username

Returns public data available for the given user

Return type dict

get_puzzle_activity(*max=None*)
Stream puzzle activity history starting with the most recent.

Parameters **max** (*int*) – maximum number of entries to stream

Returns puzzle activity history

Return type iter

get_rating_history(*username*)
Get the rating history of a user.

Parameters **username** (*str*) – a username

Returns rating history for all game types

Return type list

get_realtime_statuses(**user_ids*)
Get the online, playing, and streaming statuses of players.

Only id and name fields are returned for offline users.

Parameters **user_ids** – one or more user IDs (names)

Returns statuses of given players

Return type list

get_users_followed(*username*)
Stream users followed by a user.

Parameters **username** (*str*) – a username

Returns iterator over the users the given user follows

Return type iter

get_users_following(*username*)
Stream users who follow a user.

Parameters **username** (*str*) – a username

Returns iterator over the users that follow the given user

Return type iter

4.2 Session

class berserk.session.**Requestor**(*session*, *base_url*, *default_fmt*)
Bases: object

Encapsulates the logic for making a request.

Parameters

- **session** (requests.Session) – the authenticated session object
- **base_url** (*str*) – the base URL for requests
- **fmt** (*FormatHandler*) – default format handler to use

get(**args*, ***kwargs*)
Convenience method to make a GET request.

post(**args*, ***kwargs*)
Convenience method to make a POST request.

request(*method*, *path*, **args*, *fmt=None*, *converter=<function noop>*, ***kwargs*)
Make a request for a resource in a paticular format.

Parameters

- **method** (*str*) – HTTP verb
- **path** (*str*) – the URL suffix
- **fmt** (*FormatHandler*) – the format handler
- **converter** (*func*) – function to handle field conversions

Returns response

Raises ***berserk.exceptions.ResponseError*** – if the status is >=400

class berserk.session.**TokenSession**(*token*)
Bases: requests.sessions.Session

Session capable of personal API token authentication.

Parameters **token** (*str*) – personal API token

4.3 Enums

class berserk.enums.**PerfType**
Bases: berserk.enums.GameType

ANTICHESS = 'antichess'

ATOMIC = 'atomic'

BLITZ = 'blitz'

BULLET = 'bullet'

CHESS960 = 'chess960'

CLASSICAL = 'classical'

CRAZYHOUSE = 'crazyhouse'

HORDE = 'horde'

KING_OF_THE_HILL = 'kingOfTheHill'

RACING_KINGS = 'racingKings'

RAPID = 'rapid'

THREE_CHECK = 'threeCheck'

ULTRA_BULLET = 'ultraBullet'

class berserk.enums.**Variant**
Bases: berserk.enums.GameType

ANTICHESS = 'antichess'

ATOMIC = 'atomic'

CHESS960 = 'chess960'

CRAZYHOUSE = 'crazyhouse'

HORDE = 'horde'

KING_OF_THE_HILL = 'kingOfTheHill'

RACING_KINGS = 'racingKings'

STANDARD = 'standard'

THREE_CHECK = 'threeCheck'

class berserk.enums.**Color**
Bases: object

BLACK = 'black'

WHITE = 'white'

class berserk.enums.**Room**
Bases: object

PLAYER = 'player'

SPECTATOR = 'spectator'

class berserk.enums.**Mode**
Bases: object

CASUAL = 'casual'

RATED = 'rated'

class berserk.enums.**Position**
Bases: object

ALEKHINES_DEFENCE = 'rnbqkb1r/pppppppp/5n2/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 2 2'

ALEKHINES_DEFENCE__MODERN_VARIATION = 'rnbqkb1r/ppp1pppp/3p4/3nP3/3P4/5N2/PPP2PPP/RNBQ

```
BENKO_GAMBIT = 'rnbqkb1r/p2ppppp/5n2/1ppP4/2P5/8/PP2PPPP/RNBQKBNR w KQkq b6 1 4'
BENONI_DEFENCE__CZECH_BENONI = 'rnbqkb1r/pp1p1ppp/5n2/2pPp3/2P5/8/PP2PPPP/RNBQKBNR w K
BENONI_DEFENCE__MODERN_BENONI = 'rnbqkb1r/pp1p1ppp/4pn2/2pP4/2P5/8/PP2PPPP/RNBQKBNR w
BISHOPS_OPENING = 'rnbqkbnr/pppp1ppp/8/4p3/2B1P3/8/PPPP1PPP/RNBQK1NR b KQkq - 2 2'
BLACKMAR_DIEMER_GAMBIT = 'rnbqkbnr/ppp1pppp/8/3p4/3PP3/8/PPP2PPP/RNBQKBNR b KQkq e3 1
BOGO_INDIAN_DEFENCE = 'rnbqk2r/pppp1ppp/4pn2/8/1bPP4/5N2/PP2PPPP/RNBQKB1R w KQkq - 3 4
BONGCLOUD_ATTACK = 'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPPKPPP/RNBQ1BNR b kq - 0 2'
BUDAPEST_DEFENCE = 'rnbqkb1r/pppp1ppp/5n2/4p3/2PP4/8/PP2PPPP/RNBQKBNR w KQkq - 0 3'
CARO_KANN_DEFENCE = 'rnbqkbnr/pp1ppppp/2p5/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 1 2'
CARO_KANN_DEFENCE__ADVANCE_VARIATION = 'rnbqkbnr/pp2pppp/2p5/3pP3/3P4/8/PPP2PPP/RNBQKB
CARO_KANN_DEFENCE__CLASSICAL_VARIATION = 'rn1qkbnr/pp2pppp/2p5/5b2/3PN3/8/PPP2PPP/R1BQ
CARO_KANN_DEFENCE__EXCHANGE_VARIATION = 'rnbqkbnr/pp2pppp/2p5/3P4/3P4/8/PPP2PPP/RNBQKB
CARO_KANN_DEFENCE__PANOV_BOTVINNIK_ATTACK = 'rnbqkb1r/pp3ppp/4pn2/3p4/2PP4/2N5/PP3PPP/
CARO_KANN_DEFENCE__STEINITZ_VARIATION = 'rnbqkb1r/pp3ppp/4pn2/3p4/2PP4/2N5/PP3PPP/R1BQ
CATALAN_OPENING = 'rnbqkb1r/pppp1ppp/4pn2/8/2PP4/6P1/PP2PP1P/RNBQKBNR b KQkq - 1 3'
CATALAN_OPENING__CLOSED_VARIATION = 'rnbqk2r/ppp1bppp/4pn2/3p4/2PP4/5NP1/PP2PPBP/RNBQK
CLOSED_GAME = 'rnbqkbnr/ppp1pppp/8/3p4/3P4/8/PPP1PPPP/RNBQKBNR w KQkq - 0 2'
DANISH_GAMBIT = 'rnbqkbnr/pppp1ppp/8/8/3pP3/2P5/PP3PPP/RNBQKBNR b KQkq - 1 3'
DUTCH_DEFENCE = 'rnbqkbnr/ppppp1pp/8/5p2/3P4/8/PPP1PPPP/RNBQKBNR w KQkq f6 1 2'
DUTCH_DEFENCE__LENINGRAD_VARIATION = 'rnbqk2r/ppppp1bp/5np1/5p2/2PP4/5NP1/PP2PPBP/RNBQ
DUTCH_DEFENCE__STAUNTON_GAMBIT = 'rnbqkb1r/ppppp1pp/5n2/6B1/3Pp3/2N5/PPP2PPP/R2QKBNR b
DUTCH_DEFENCE__STONEWALL_VARIATION = 'rnbq1rk1/ppp1b1pp/4pn2/3p1p2/2PP4/5NP1/PP2PPBP/R
ENGLISH_OPENING = 'rnbqkbnr/pppppppp/8/8/2P5/8/PP1PPPPP/RNBQKBNR b KQkq c3 1 1'
ENGLISH_OPENING__CLOSED_SYSTEM = 'r1bqk1nr/ppp2pbp/2np2p1/4p3/2P5/2NP2P1/PP2PPBP/R1BQK
ENGLISH_OPENING__REVERSED_SICILIAN = 'rnbqkbnr/pppp1ppp/8/4p3/2P5/8/PP1PPPPP/RNBQKBNR
ENGLISH_OPENING__SYMMETRICAL_VARIATION = 'rnbqkbnr/pp1ppppp/8/2p5/2P5/8/PP1PPPPP/RNBQK
FOUR_KNIGHTS_GAME = 'r1bqkb1r/pppp1ppp/2n2n2/4p3/4P3/2N2N2/PPPP1PPP/R1BQKB1R w KQkq -
FOUR_KNIGHTS_GAME__SCOTCH_VARIATION = 'r1bqkb1r/pppp1ppp/2n2n2/4p3/3PP3/2N2N2/PPP2PPP/
FOUR_KNIGHTS_GAME__SPANISH_VARIATION = 'r1bqkb1r/pppp1ppp/2n2n2/1B2p3/4P3/2N2N2/PPPP1P
FRANKENSTEIN_DRACULA_VARIATION = 'rnbqkb1r/pppp1ppp/8/4p3/2B1n3/2N5/PPPP1PPP/R1BQK1NR
FRENCH_DEFENCE = 'rnbqkbnr/pppp1ppp/4p3/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 1 2'
FRENCH_DEFENCE__ADVANCE_VARIATION = 'rnbqkbnr/ppp2ppp/4p3/3pP3/3P4/8/PPP2PPP/RNBQKBNR
FRENCH_DEFENCE__BURN_VARIATION = 'rnbqkb1r/ppp2ppp/4pn2/3p2B1/3PP3/2N5/PPP2PPP/R2QKBNR
FRENCH_DEFENCE__CLASSICAL_VARIATION = 'rnbqkb1r/ppp2ppp/4pn2/3p4/3PP3/2N5/PPP2PPP/R1BQ
FRENCH_DEFENCE__EXCHANGE_VARIATION = 'rnbqkbnr/ppp2ppp/4p3/3P4/3P4/8/PPP2PPP/RNBQKBNR
FRENCH_DEFENCE__RUBINSTEIN_VARIATION = 'rnbqkbnr/ppp2ppp/4p3/8/3Pp3/2N5/PPP2PPP/R1BQKB
```

```
FRENCH_DEFENCE__TARRASCH_VARIATION = 'rnbqkbnr/ppp2ppp/4p3/3p4/3PP3/8/PPPN1PPP/R1BQKBN
FRENCH_DEFENCE__WINAWER_VARIATION = 'rnbqk1nr/ppp2ppp/4p3/3p4/1b1PP3/2N5/PPP2PPP/R1BQK
GIUOCO_PIANO = 'r1bqk1nr/pppp1ppp/2n5/2b1p3/2B1P3/5N2/PPPP1PPP/RNBQK2R w KQkq - 5 4'
GRUNFELD_DEFENCE = 'rnbqkb1r/ppp1pp1p/5np1/3p4/2PP4/2N5/PP2PPPP/R1BQKBNR w KQkq d6 1 4
GRUNFELD_DEFENCE__BRINCKMANN_ATTACK = 'rnbqkb1r/ppp1pp1p/5np1/3p4/2PP1B2/2N5/PP2PPPP/R
GRUNFELD_DEFENCE__EXCHANGE_VARIATION = 'rnbqkb1r/ppp1pp1p/6p1/3n4/3P4/2N5/PP2PPPP/R1BQ
GRUNFELD_DEFENCE__RUSSIAN_VARIATION = 'rnbqkb1r/ppp1pp1p/5np1/3p4/2PP4/1QN5/PP2PPPP/R1
GRUNFELD_DEFENCE__TAIMANOV_VARIATION = 'rnbqk2r/ppp1ppbp/5np1/3p2B1/2PP4/2N2N2/PP2PPPP
HALLOWEEN_GAMBIT = 'r1bqkb1r/pppp1ppp/2n2n2/4N3/4P3/2N5/PPPP1PPP/R1BQKB1R b KQkq - 1 4
HUNGARIAN_OPENING = 'rnbqkbnr/pppppppp/8/8/8/6P1/PPPPPPP1P/RNBQKBNR b KQkq - 1 1'
ITALIAN_GAME = 'r1bqkbnr/pppp1ppp/2n5/4p3/2B1P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 4 3'
ITALIAN_GAME__EVANS_GAMBIT = 'r1bqk1nr/pppp1ppp/2n5/2b1p3/1PB1P3/5N2/P1PP1PPP/RNBQK2R
ITALIAN_GAME__HUNGARIAN_DEFENCE = 'r1bqk1nr/ppppbppp/2n5/4p3/2B1P3/5N2/PPPP1PPP/RNBQK2
ITALIAN_GAME__TWO_KNIGHTS_DEFENCE = 'r1bqkb1r/pppp1ppp/2n2n2/4p3/2B1P3/5N2/PPPP1PPP/RN
KINGS_GAMBIT = 'rnbqkbnr/pppp1ppp/8/4p3/4PP2/8/PPPP2PP/RNBQKBNR b KQkq f3 1 2'
KINGS_GAMBIT_ACCEPTED = 'rnbqkbnr/pppp1ppp/8/8/4Pp2/8/PPPP2PP/RNBQKBNR w KQkq - 1 3'
KINGS_GAMBIT_ACCEPTED__BISHOPS_GAMBIT = 'rnbqkbnr/pppp1ppp/8/8/2B1Pp2/8/PPPP2PP/RNBQK1
KINGS_GAMBIT_ACCEPTED__CLASSICAL_VARIATION = 'rnbqkbnr/pppp1p1p/8/6p1/4Pp2/5N2/PPPP2PP
KINGS_GAMBIT_ACCEPTED__MODERN_DEFENCE = 'rnbqkbnr/ppp2ppp/8/3p4/4Pp2/5N2/PPPP2PP/RNBQK
KINGS_GAMBIT_DECLINED__CLASSICAL_VARIATION = 'rnbqk1nr/pppp1ppp/8/2b1p3/4PP2/8/PPPP2PP
KINGS_GAMBIT_DECLINED__FALKBEER_COUNTERGAMBIT = 'rnbqkbnr/ppp2ppp/8/3pp3/4PP2/8/PPPP2P
KINGS_INDIAN_ATTACK = 'rnbqkbnr/ppp1pppp/8/3p4/8/5NP1/PPPPPP1P/RNBQKB1R b KQkq - 1 2'
KINGS_INDIAN_DEFENCE = 'rnbqkb1r/pppppp1p/5np1/8/2PP4/8/PP2PPPP/RNBQKBNR w KQkq - 1 3'
KINGS_INDIAN_DEFENCE__4E4 = 'rnbqk2r/ppp1ppbp/3p1np1/8/2PPP3/2N5/PP3PPP/R1BQKBNR w KQk
KINGS_INDIAN_DEFENCE__AVERBAKH_VARIATION = 'rnbq1rk1/ppp1ppbp/3p1np1/6B1/2PPP3/2N5/PP2
KINGS_INDIAN_DEFENCE__CLASSICAL_VARIATION = 'rnbq1rk1/ppp1ppbp/3p1np1/8/2PPP3/2N2N2/PP
KINGS_INDIAN_DEFENCE__FIANCHETTO_VARIATION = 'rnbqk2r/ppp1ppbp/3p1np1/8/2PP4/2N2NP1/PP
KINGS_INDIAN_DEFENCE__FOUR_PAWNS_ATTACK = 'rnbqk2r/ppp1ppbp/3p1np1/8/2PPPP2/2N5/PP4PP/
KINGS_INDIAN_DEFENCE__SAMISCH_VARIATION = 'rnbqk2r/ppp1ppbp/3p1np1/8/2PPP3/2N2P2/PP4PP
KINGS_PAWN = 'rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 1 1'
LONDON_SYSTEM = 'rnbqkb1r/ppp1pppp/5n2/3p4/3P1B2/5N2/PPP1PPPP/RN1QKB1R b KQkq - 4 3'
MODERN_DEFENCE = 'rnbqkbnr/pppppp1p/6p1/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 2'
MODERN_DEFENCE__ROBATSCH_DEFENCE = 'rnbqk1nr/pppppppbp/6p1/8/3PP3/2N5/PPP2PPP/R1BQKBNR
NIMZO_INDIAN_DEFENCE = 'rnbqk2r/pppp1ppp/4pn2/8/1bPP4/2N5/PP2PPPP/R1BQKBNR w KQkq - 3
NIMZO_INDIAN_DEFENCE__CLASSICAL_VARIATION = 'rnbqk2r/pppp1ppp/4pn2/8/1bPP4/2N5/PPQ1PPP
NIMZO_INDIAN_DEFENCE__FISCHER_VARIATION = 'rnbqk2r/p1pp1ppp/1p2pn2/8/1bPP4/2N1P3/PP3PP
```

```
NIMZO_INDIAN_DEFENCE__HUBNER_VARIATION = 'r1bqk2r/pp3ppp/2nppn2/2p5/2PP4/2PBPN2/P4PPP/
NIMZO_INDIAN_DEFENCE__KASPAROV_VARIATION = 'rnbqk2r/pppp1ppp/4pn2/8/1bPP4/2N2N2/PP2PPP
NIMZO_INDIAN_DEFENCE__LENINGRAD_VARIATION = 'rnbqk2r/pppp1ppp/4pn2/6B1/1bPP4/2N5/PP2PP
NIMZO_INDIAN_DEFENCE__SAMISCH_VARIATION = 'rnbqk2r/pppp1ppp/4pn2/8/2PP4/P1P5/4PPPP/R1B
NIMZO_LARSEN_ATTACK = 'rnbqkbnr/pppppppp/8/8/8/1P6/P1PPPPPP/RNBQKBNR b KQkq - 1 1'
OLD_INDIAN_DEFENCE = 'rnbqkb1r/ppp1pppp/3p1n2/8/2PP4/8/PP2PPPP/RNBQKBNR w KQkq - 1 3'
OPEN_GAME = 'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 2'
PETROVS_DEFENCE = 'rnbqkb1r/pppp1ppp/5n2/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R w KQkq - 3 3'
PETROVS_DEFENCE__CLASSICAL_ATTACK = 'rnbqkb1r/ppp2ppp/3p4/8/3Pn3/5N2/PPP2PPP/RNBQKB1R
PETROVS_DEFENCE__STEINITZ_ATTACK = 'rnbqkb1r/pppp1ppp/5n2/4p3/3PP3/5N2/PPP2PPP/RNBQKB1
PETROVS_DEFENCE__THREE_KNIGHTS_GAME = 'rnbqkb1r/pppp1ppp/5n2/4p3/4P3/2N2N2/PPPP1PPP/R1
PHILIDOR_DEFENCE = 'rnbqkbnr/ppp2ppp/3p4/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R w KQkq - 1 3'
PIRC_DEFENCE = 'rnbqkb1r/ppp1pppp/3p1n2/8/3PP3/8/PPP2PPP/RNBQKBNR w KQkq - 2 3'
PIRC_DEFENCE__AUSTRIAN_ATTACK = 'rnbqkb1r/ppp1pp1p/3p1np1/8/3PPP2/2N5/PPP3PP/R1BQKBNR
PIRC_DEFENCE__CLASSICAL_VARIATION = 'rnbqkb1r/ppp1pp1p/3p1np1/8/3PP3/2N2N2/PPP2PPP/R1B
QUEENSS_PAWN_GAME__MODERN_DEFENCE = 'rnbqk1nr/ppp1ppbp/3p2p1/8/2PP4/2N5/PP2PPPP/R1BQKB
QUEENS_GAMBIT = 'rnbqkbnr/ppp1pppp/8/3p4/2PP4/8/PP2PPPP/RNBQKBNR b KQkq c3 1 2'
QUEENS_GAMBIT_ACCEPTED = 'rnbqkbnr/ppp1pppp/8/8/2pP4/8/PP2PPPP/RNBQKBNR w KQkq - 1 3'
QUEENS_GAMBIT_DECLINED__ALBIN_COUNTERGAMBIT = 'rnbqkbnr/ppp2ppp/8/3pp3/2PP4/8/PP2PPPP/
QUEENS_GAMBIT_DECLINED__CHIGORIN_DEFENCE = 'r1bqkbnr/ppp1pppp/2n5/3p4/2PP4/8/PP2PPPP/R
QUEENS_GAMBIT_DECLINED__SEMI_SLAV_DEFENCE = 'rnbqkb1r/pp3ppp/2p1pn2/3p4/2PP4/2N2N2/PP2
QUEENS_GAMBIT_DECLINED__SEMI_TARRASCH_DEFENCE = 'rnbqkb1r/pp3ppp/4pn2/2pp4/2PP4/2N2N2/
QUEENS_GAMBIT_DECLINED__SLAV_DEFENCE = 'rnbqkbnr/pp2pppp/2p5/3p4/2PP4/8/PP2PPPP/RNBQKB
QUEENS_GAMBIT_DECLINED__TARRASCH_DEFENCE = 'rnbqkbnr/pp3ppp/4p3/2pp4/2PP4/2N5/PP2PPPP/
QUEENS_INDIAN_DEFENCE = 'rnbqkb1r/p1pp1ppp/1p2pn2/8/2PP4/5N2/PP2PPPP/RNBQKB1R w KQkq -
QUEENS_PAWN = 'rnbqkbnr/pppppppp/8/8/3P4/8/PPP1PPPP/RNBQKBNR b KQkq d3 1 1'
RETI_OPENING = 'rnbqkbnr/ppp1pppp/8/3p4/2P5/5N2/PP1PPPPP/RNBQKB1R b KQkq c3 1 2'
RICHTER_VERESOV_ATTACK = 'rnbqkb1r/ppp1pppp/5n2/3p2B1/3P4/2N5/PPP1PPPP/R2QKBNR b KQkq
RUY_LOPEZ = 'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 4 3'
RUY_LOPEZ__BERLIN_DEFENCE = 'r1bqkb1r/pppp1ppp/2n2n2/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R w
RUY_LOPEZ__CLASSICAL_VARIATION = 'r1bqk1nr/pppp1ppp/2n5/1Bb1p3/4P3/5N2/PPPP1PPP/RNBQK2
RUY_LOPEZ__CLOSED_VARIATION = 'r1bqk2r/2ppbppp/p1n2n2/1p2p3/4P3/1B3N2/PPPP1PPP/RNBQR1K
RUY_LOPEZ__EXCHANGE_VARIATION = 'r1bqkbnr/1ppp1ppp/p1B5/4p3/4P3/5N2/PPPP1PPP/RNBQK2R b
RUY_LOPEZ__MARSHALL_ATTACK = 'r1bq1rk1/2p1bppp/p1n2n2/1p1pp3/4P3/1BP2N2/PP1P1PPP/RNBQR
RUY_LOPEZ__SCHLIEMANN_DEFENCE = 'r1bqkbnr/pppp2pp/2n5/1B2pp2/4P3/5N2/PPPP1PPP/RNBQK2R
SCANDINAVIAN_DEFENCE = 'rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR w KQkq d6 1 2'
```

```
SCANDINAVIAN_DEFENCE__MODERN_VARIATION = 'rnbqkb1r/ppp1pppp/5n2/3P4/3P4/8/PPP2PPP/RNBQ
SCOTCH_GAME = 'r1bqkbnr/pppp1ppp/2n5/4p3/3PP3/5N2/PPP2PPP/RNBQKB1R b KQkq d3 1 3'
SCOTCH_GAME__CLASSICAL_VARIATION = 'r1bqk1nr/pppp1ppp/2n5/2b5/3NP3/8/PPP2PPP/RNBQKB1R
SCOTCH_GAME__MIESES_VARIATION = 'r1bqkb1r/p1pp1ppp/2p2n2/4P3/8/8/PPP2PPP/RNBQKB1R b KQ
SCOTCH_GAME__STEINITZ_VARIATION = 'r1b1kbnr/pppp1ppp/2n5/8/3NP2q/8/PPP2PPP/RNBQKB1R w
SICILIAN_DEFENCE = 'rnbqkbnr/pp1ppppp/8/2p5/4P3/8/PPPP1PPP/RNBQKBNR w KQkq c6 1 2'
SICILIAN_DEFENCE__ACCELERATED_DRAGON = 'r1bqkbnr/pp1ppp1p/2n3p1/8/3NP3/8/PPP2PPP/RNBQK
SICILIAN_DEFENCE__ALAPIN_VARIATION = 'rnbqkbnr/pp1ppppp/8/2p5/4P3/2P5/PP1P1PPP/RNBQKBN
SICILIAN_DEFENCE__CLOSED_VARIATION = 'rnbqkbnr/pp1ppppp/8/2p5/4P3/2N5/PPPP1PPP/R1BQKBN
SICILIAN_DEFENCE__DRAGON_VARIATION = 'rnbqkb1r/pp2pp1p/3p1np1/8/3NP3/2N5/PPP2PPP/R1BQK
SICILIAN_DEFENCE__GRAND_PRIX_ATTACK = 'r1bqkbnr/pp1ppppp/2n5/2p5/4PP2/2N5/PPPP2PP/R1BQ
SICILIAN_DEFENCE__HYPER_ACCELERATED_DRAGON = 'rnbqkbnr/pp1ppp1p/6p1/2p5/4P3/5N2/PPPP1P
SICILIAN_DEFENCE__KAN_VARIATION = 'rnbqkbnr/1p1p1ppp/p3p3/8/3NP3/8/PPP2PPP/RNBQKB1R w
SICILIAN_DEFENCE__NAJDORF_VARIATION = 'rnbqkb1r/1p2pppp/p2p1n2/8/3NP3/2N5/PPP2PPP/R1BQ
SICILIAN_DEFENCE__RICHTER_RAUZER_VARIATION = 'r1bqkb1r/pp2pppp/2np1n2/6B1/3NP3/2N5/PPP
SICILIAN_DEFENCE__SCHEVENINGEN_VARIATION = 'rnbqkb1r/pp3ppp/3ppn2/8/3NP3/2N5/PPP2PPP/R
SICILIAN_DEFENCE__SMITH_MORRA_GAMBIT = 'rnbqkbnr/pp1ppppp/8/8/3pP3/2P5/PP3PPP/RNBQKBNR
SOKOLSKY_OPENING = 'rnbqkbnr/pppppppp/8/8/1P6/8/P1PPPPPP/RNBQKBNR b KQkq - 1 1'
TORRE_ATTACK = 'rnbqkb1r/ppp1pppp/5n2/3p2B1/3P4/5N2/PPP1PPPP/RN1QKB1R b KQkq - 4 3'
TROMPOWSKY_ATTACK = 'rnbqkb1r/pppppppp/5n2/6B1/3P4/8/PPP1PPPP/RN1QKBNR b KQkq - 3 2'
VIENNA_GAME = 'rnbqkbnr/pppp1ppp/8/4p3/4P3/2N5/PPPP1PPP/R1BQKBNR b KQkq - 2 2'
ZUKERTORT_OPENING = 'rnbqkbnr/pppppppp/8/8/8/5N2/PPPPPPPP/RNBQKB1R b KQkq - 1 1'
```

4.4 Formats

class berserk.formats.**FormatHandler**(*mime_type*)

Bases: object

Provide request headers and parse responses for a particular format.

Instances of this class should override the *parse_stream()* and *parse()* methods to support handling both streaming and non-streaming responses.

Parameters **mime_type** (*str*) – the MIME type for the format

handle(*response*, *is_stream*, *converter=<function noop>*)

Handle the response by returning the data.

Parameters

- **response** (requests.Response) – raw response
- **is_stream** (*bool*) – True if the response is a stream
- **converter** (*func*) – function to handle field conversions

Returns either all response data or an iterator of response data

parse(*response*)
Parse all data from a response.

Parameters **response** (requests.Response) – raw response

Returns response data

parse_stream(*response*)
Yield the parsed data from a stream response.

Parameters **response** (requests.Response) – raw response

Returns iterator over the response data

berserk.formats.**JSON = <berserk.formats.JsonHandler object>**
Handles vanilla JSON

class berserk.formats.**JsonHandler**(*mime_type*, *decoder=<class 'json.decoder.JSONDecoder'>*)
Bases: *berserk.formats.FormatHandler*

Handle JSON data.

Parameters

- **mime_type** (*str*) – the MIME type for the format
- **decoder** (json.JSONDecoder) – the decoder to use for the JSON format

parse(*response*)
Parse all JSON data from a response.

Parameters **response** (requests.Response) – raw response

Returns response data

Return type JSON

parse_stream(*response*)
Yield the parsed data from a stream response.

Parameters **response** (requests.Response) – raw response

Returns iterator over multiple JSON objects

berserk.formats.**LIJSON = <berserk.formats.JsonHandler object>**
Handles oddball LiChess JSON (normal JSON, crazy MIME type)

berserk.formats.**NDJSON = <berserk.formats.JsonHandler object>**
Handles newline-delimited JSON

berserk.formats.**PGN = <berserk.formats.PgnHandler object>**
Handles PGN

class berserk.formats.**PgnHandler**
Bases: *berserk.formats.FormatHandler*

Handle PGN data.

handle(**args*, ***kwargs*)
Handle the response by returning the data.

Parameters

- **response** (requests.Response) – raw response

- **is_stream** (*bool*) – `True` if the response is a stream
- **converter** (*func*) – function to handle field conversions

Returns either all response data or an iterator of response data

parse(*response*)
Parse all text data from a response.

Parameters **response** (`requests.Response`) – raw response

Returns response text

Return type str

parse_stream(*response*)
Yield the parsed PGN games from a stream response.

Parameters **response** (`requests.Response`) – raw response

Returns iterator over multiple PGN texts

berserk.formats.**TEXT = <berserk.formats.TextHandler object>**
Basic text

class berserk.formats.**TextHandler**
Bases: *berserk.formats.FormatHandler*

parse(*response*)
Parse all data from a response.

Parameters **response** (`requests.Response`) – raw response

Returns response data

parse_stream(*response*)
Yield the parsed data from a stream response.

Parameters **response** (`requests.Response`) – raw response

Returns iterator over the response data

4.5 Exceptions

exception berserk.exceptions.**ApiError**(*error*)
Bases: *berserk.exceptions.BerserkError*

exception berserk.exceptions.**BerserkError**
Bases: `Exception`

message

exception berserk.exceptions.**ResponseError**(*response*)
Bases: *berserk.exceptions.ApiError*

Response that indicates an error.

cause

reason
HTTP status text of the response.

status_code
HTTP status code of the response.

berserk.exceptions.**get_message**(*e*)

berserk.exceptions.**set_message**(*e*, *value*)

4.6 Utils

berserk.utils.**build_adapter**(*mapper*, *sep='.'*)

Build a data adapter.

Uses a map to pull values from an object and assign them to keys. For example:

```
>>> mapping = {
...   'broadcast_id': 'broadcast.id',
...   'slug': 'broadcast.slug',
...   'name': 'broadcast.name',
...   'description': 'broadcast.description',
...   'syncUrl': 'broadcast.sync.url',
... }

>>> cast = {'broadcast': {'id': 'WxOb8OUT',
...   'slug': 'test-tourney',
...   'name': 'Test Tourney',
...   'description': 'Just a test',
...   'ownerId': 'rhgrant10',
...   'sync': {'ongoing': False, 'log': [], 'url': None}},
...  'url': 'https://lichess.org/broadcast/test-tourney/WxOb8OUT'}

>>> adapt = build_adapter(mapping)
>>> adapt(cast)
{'broadcast_id': 'WxOb8OUT',
'slug': 'test-tourney',
'name': 'Test Tourney',
'description': 'Just a test',
'syncUrl': None}
```

Parameters

- **mapper** (*dict*) – map of keys to their location in an object
- **sep** (*str*) – nested key delimiter

Returns adapted data

Return type dict

berserk.utils.**datetime_from_millis**(*millis*)

Return the datetime for the given millis since the epoch.

UTC is assumed. The returned datetime is timezone aware.

Returns timezone aware datetime

Return type datetime

berserk.utils.**datetime_from_seconds**(*ts*)

Return the datetime for the given seconds since the epoch.

UTC is assumed. The returned datetime is timezone aware.

Returns timezone aware datetime

Return type datetime

berserk.utils.**datetime_from_str**(*dt_str*)
Convert the time in a string to a datetime.

UTC is assumed. The returned datetime is timezone aware. The format must match %Y-%m-%dT%H:%M:%S.%fZ.

Returns timezone aware datetime

Return type datetime

berserk.utils.**to_millis**(*dt*)
Return the milliseconds between the given datetime and the epoch.

Parameters **dt** (*datetime*) – a datetime

Returns milliseconds since the epoch

Return type int

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/rhgrant10/berserk/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/rhgrant10/berserk/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/berserk.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv berserk
$ cd berserk/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 berserk tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/rhgrant10/berserk/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_berserk
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Authors

6.1 Development Lead

- Robert Grant

6.2 Developers

- Robert Graham

6.3 Contributors

- Harald Klein
- *your name here* :)

CHAPTER 7

History

7.1 0.10.0 (2020-04-26)

- Add `Challenge.create_ai` for creating an AI challenge
- Add `Challenge.create_open` for creating an open challenge
- Add `Challenge.create_with_accept` auto-acceptance of challenges using OAuth token
- Bugfix for passing initial board positions in FEN for challenges
- Minor fixes for docstrings

7.2 0.9.0 (2020-04-14)

- Add remaining `Board` endpoints: seek, handle_draw_offer, offer_draw, accept_draw, and decline_draw
- Multiple doc updates/fixes
- Add codecov reporting

7.3 0.8.0 (2020-03-08)

- Add new `Board` client: stream_incoming_events, stream_game_state, make_move, post_message, abort_game, and resign_game

7.4 0.7.0 (2020-01-26)

- Add simuls
- Add studies export and export chapter

- Add tournament results, games export, and list by creator
- Add user followers, users following, rating history, and puzzle activity
- Add new `Teams` client: join, get members, kick member, and leave
- Updated documentation, including new docs for some useful utils
- Fixed bugs in `Tournaments.export_games`
- Deprecated `Users.get_by_team` - use `Teams.get_members` instead

7.5 0.6.1 (2020-01-20)

- Add py37 to the travis build
- Update development status classifier to 4 - Beta
- Fix py36 issue preventing successful build
- Make updates to the Makefile

7.6 0.6.0 (2020-01-20)

- Add logging to the `berserk.session` module
- Fix exception message when no cause
- Fix bug in `Broadcasts.push_pgn_update`
- Update documentation and tweak the theme

7.7 0.5.0 (2020-01-20)

- Add `ResponseError` for 4xx and 5xx responses with status code, reason, and cause
- Add `ApiError` for all other request errors
- Fix test case broken by 0.4.0 release
- Put all utils code under test

7.8 0.4.0 (2020-01-19)

- Add support for the broadcast endpoints
- Add a utility for easily converting API objects into update params
- Fix multiple bugs with the tournament create endpoint
- Improve the reusability of some conversion utilities
- Improve many docstrings in the client classes

7.9 0.3.2 (2020-01-04)

- Fix bug where options not passed for challenge creation
- Convert requirements from pinned to sematically compatible
- Bump all developer dependencies
- Use pytest instead of the older py.test
- Use py37 in tox

7.10 0.3.1 (2018-12-23)

- Convert datetime string in tournament creation response into datetime object

7.11 0.3.0 (2018-12-23)

- Convert all timestamps to datetime in all responses
- Provide support for challenging other players to a game

7.12 0.2.1 (2018-12-08)

- Bump requests dependency to >-2.20.0 (CVE-2018-18074)

7.13 0.2.0 (2018-12-08)

- Add *position* and *start_date* params to *Tournament.create*
- Add *Position* enum

7.14 0.1.2 (2018-07-14)

- Fix an asine bug in the docs

7.15 0.1.1 (2018-07-14)

- Added tests for session and formats modules
- Fixed mispelled PgnHandler class (!)
- Fixed issue with trailing whitespace when splitting multiple PGN texts
- Fixed the usage overview in the README
- Fixed the versions for travis-ci
- Made it easier to test the *JsonHandler* class

- Salted the bumpversion config to taste

7.16 0.1.0 (2018-07-10)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R