
battleofai Documentation

Release 1.0.0a

Linus Bartsch

Oct 11, 2018

Contents:

1	Quickstart	1
1.1	Install	1
1.2	A short summary of the basic features.	1
2	battleofai	3
2.1	battleofai package	3
3	Indices and tables	9
	Python Module Index	11

CHAPTER 1

Quickstart

Some things you need to get started quickly.

1.1 Install

Latest stable version:

```
python3.6 -m pip install battleofai
```

This branch (Alpha state):

```
python3.6 -m pip install -U git+https://github.com/LiBa001/python-battleofai@async
```

Note: This requires [git](#) to be installed.

1.2 A short summary of the basic features.

1.2.1 Configure your client easily.

```
from battleofai import Client

client = Client(credentials=('username', 'password'))

# if you want to export your config
client.config.from_python_file('config.py')

# or use json
client.config.from_json_file('config.json')
```

1.2.2 Play games automatically.

```
from battleofai import Client, Core

client = Client(credentials=('username', 'password'))

@client.callback(Core)
async def turn(board, symbol):
    for x_pos, columns in enumerate(board):
        for y_pos, field in enumerate(columns):
            if field == '#': # if position is free
                return x_pos, y_pos # set my stone

@client.on_ready
async def main():
    # called when the client logged in and finished up

    session = client.create_session(Core) # create and register a new game session

    await session.run() # find a match and play the game

client.run()
```

1.2.3 Play multiple games concurrently.

```
from battleofai import Client, Core, GameSession

client = Client()

# setup callback like above

@client.on_ready
async def main():
    # play multiple games concurrently
    session_1 = GameSession(Core, rejoin_ongoing_games=False, turn_interval=0.5, join_
↪own_games=True)
    session_2 = GameSession(Core, rejoin_ongoing_games=False, turn_interval=0.5, join_
↪own_games=False)

    session_1.register_client(client)
    session_1.run()

    session_2.register_client(client)
    session_2.run()

    await session_1.task
    await session_2.task

client.run()
```

2.1 battleofai package

2.1.1 Submodules

2.1.2 battleofai.abc module

```
class battleofai.abc.GameType (*args, **kwargs)
    Bases: abc.ABCMeta

class battleofai.abc.Match (http_client: battleofai.http.HTTPClient, callback: callable)
    Bases: object

    classmethod create_game (http_client: battleofai.http.HTTPClient)
        creates a game :return: the game

    game
    is_active

    join_game (game: battleofai.game.Game = None, *, join_own_games: bool = False)
        Registers the player for a match

        Parameters
        • game – The match to register on
        • join_own_games – Indicates if you'd like to join a GameState.WAITING game against
          yourself.

    make_turn ()

    classmethod open_games (http_client: battleofai.http.HTTPClient) → AsyncGenerator

    opponents

    play (turn_interval: int = 5, matchmaking_interval: int = 5)
        Plays the game. :return: True if you won, else False
```

ready

rejoin_ongoing_game()
Rejoins the latest ongoing game that the client left. :return: None

won

2.1.3 battleofai.client module

class battleofai.client.**Client** (*username: str = None, password: str = None, *, credentials: tuple = None, loop=None, **options*)

Bases: object

callback (*game_type: battleofai.abc.GameType = None*) → callable

Use this to decorate your callback functions. Alternative to `register_callback()`.

check_and_update_token()

close()

create_session (*game_type: battleofai.abc.GameType, name: str = None, callback: callable = None, **kwargs*) → battleofai.game_session.GameSession

get_callback (*game_type: battleofai.abc.GameType*)

login (*username: str = None, password: str = None, *, credentials: tuple = None*)

on_ready (*func: callable*)

register_callback (*callback: callable, game_type: battleofai.abc.GameType = None*) → None

This is to register a callback for a game type. :type callback: callable :param callback:

A callable used to make a turn.

Parameters **game_type** (*battleofai.abc.GameType*) – The game type to use this callback for. If None it's used for any game type where no other callback is specified. Instance of *battleofai.abc.GameType* (Subclass of *battleofai.abc.Match*)

e.g. `battleofai.Core`.

register_session (*session: battleofai.game_session.GameSession*)

run (**args, **kwargs*)

A blocking call that abstracts away the *event loop* initialisation from you. If you want more control over the event loop then this function should not be used.

This function must be the last function to call due to the fact that it is blocking. That means that registration of events or anything being called after this function call will not execute until it returns.

run_sessions()

sessions

set_credentials (*username: str = None, password: str = None, *, credentials: tuple = None*)

start (**args, **kwargs*)

2.1.4 battleofai.config module

class battleofai.config.**Config**

Bases: dict

```
from_json_file(path)
from_python_file(path)
password
username
```

2.1.5 battleofai.enums module

class battleofai.enums.GameState

Bases: enum.Enum

An enumeration.

ABORTED = 'ABORTED'

FINISHED = 'FINISHED'

NON_EXISTENT = None

STARTED = 'STARTED'

WAITING = 'WAITING'

2.1.6 battleofai.errors module

exception battleofai.errors.BOAIException

Bases: Exception

Base Exception class for battleofai

exception battleofai.errors.ClientException

Bases: *battleofai.errors.BOAIException*

Exception that's thrown when an operation in the :class: *Client* fails.

These are usually for exceptions that happened due to user input.

exception battleofai.errors.Forbidden(*response*, *message*: str = "")

Bases: *battleofai.errors.HTTPException*

Exception that's thrown for when status code 403 occurs.

Subclass of *HTTPException*

exception battleofai.errors.HTTPException(*response*, *message*: str = "")

Bases: *battleofai.errors.BOAIException*

Exception that's thrown when an HTTP request operation fails.

response

The response of the failed HTTP request. This is an instance of *aiohttp.ClientResponse*.

message

The text of the error. Could be an empty string.

exception battleofai.errors.LoginFailure

Bases: *battleofai.errors.ClientException*

Exception that's thrown when the *IAMClient.login()* function fails to log you in from improper credentials or some other misc. failure.

exception `battleofai.errors.NotFound(response, message: str = '')`

Bases: `battleofai.errors.HTTPException`

Exception that's thrown for when status code 404 occurs.

Subclass of `HTTPException`

exception `battleofai.errors.Unauthorized(response, message: str = '')`

Bases: `battleofai.errors.HTTPException`

Exception that's thrown for when status code 401 occurs.

Subclass of `HTTPException`

2.1.7 battleofai.game module

class `battleofai.game.Game(http_client: battleofai.http.HTTPClient)`

Bases: `object`

active_player

classmethod `create(http_client: battleofai.http.HTTPClient, game_name: str)`

creates a game :param game_name: The name of the game (e.g. 'Core') :param http_client: The client to use for registration :return: the game

current_board

classmethod `get(http_client: battleofai.http.HTTPClient, game_id: int)`

classmethod `get_all(http_client: battleofai.http.HTTPClient, **criteria)`

history

id

classmethod `list(http_client: battleofai.http.HTTPClient, **criteria)`

make_turn (`turn_data: tuple`) → `bool`

name

open_slots

players

pull (`game_id: int`)

state

update ()

winning_player

2.1.8 battleofai.game_session module

class `battleofai.game_session.GameSession(game_type: battleofai.abc.GameType, name: str = None, callback: callable = None, *, rejoin_ongoing_games=False, join_own_games=False, turn_interval=1, matchmaking_interval=5)`

Bases: `object`

Represents a client's game session. (The client playing a match.)

```
cancel()
client
find_match()
is_running
match
register_client(client)
result
run()
run_for_client(client)
run_match()
set_match(match: battleofai.abc.Match)
setup(client, match: battleofai.abc.Match)
task
```

2.1.9 battleofai.game_types module

```
class battleofai.game_types.Core(http_client: battleofai.http.HTTPClient, callback: callable)
    Bases: battleofai.abc.Match
    static get_symbol(active_player)
    make_turn() → bool
    play(turn_interval: int = 5, matchmaking_interval: int = 5) → bool
        Plays the game. :return: True if you won, else False
    symbol
```

2.1.10 battleofai.http module

```
class battleofai.http.GamesRoute(method: battleofai.http.Method, path, **parameters)
    Bases: battleofai.http.Route
    BASE = 'https://games.battleofai.net/api/games'
    MAJOR_PARAMS = ['game_id']
class battleofai.http.HTTPClient(connector=None, *, loop=None)
    Bases: object
    Represents an HTTP client sending HTTP requests to the BOAI APIs.
    close()
    create_game(game_name: str) → int
        Creates a new game. :param game_name: The name of the game (e.g. 'Core') :return: The game id
    get_game(game_id) → dict
        Gets all info about the given game. :param game_id: The ID of the game. :return: A dict containing all
        info about the game.
```

```
get_games (**criteria) → [<class 'int'>]
    Gets a list of all existing games meeting the given criteria. :param criteria: Select games meeting specific
    criteria e.g. game_state='WAITING' :return: List of game IDs

login (username: str, password: str)
    Logs a user in with credentials of the registration for obtaining valid credentials for playing a game. :return:
    A tuple containing player_id and login token. Credentials are valid for 1 day.

make_turn (game_id, turn_data: tuple) → bool

player_auth

register_player (game_id) → bool
    Registers the player for a match by id :param game_id: The match to register on :return: True -> success;
    False -> failure

request (route, **kwargs) → dict

token

validate_token () → bool

class battleofai.http.IAMRoute (method: battleofai.http.Method, path, **parameters)
    Bases: battleofai.http.Route

    BASE = 'https://iam.battleofai.net/api/iam'

    MAJOR_PARAMS = ['user_id']

class battleofai.http.Method
    Bases: enum.Enum

    An enumeration.

    GET = 'GET'

    POST = 'POST'

class battleofai.http.Route (method: battleofai.http.Method, path, **parameters)
    Bases: object

    BASE = ''

    MAJOR_PARAMS = []

    bucket
```

2.1.11 Module contents

CHAPTER 3

Indices and tables

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

b

- `battleofai`, 8
- `battleofai.abc`, 3
- `battleofai.client`, 4
- `battleofai.config`, 4
- `battleofai.enums`, 5
- `battleofai.errors`, 5
- `battleofai.game`, 6
- `battleofai.game_session`, 6
- `battleofai.game_types`, 7
- `battleofai.http`, 7

A

ABORTED (battleofai.enums.GameState attribute), 5
 active_player (battleofai.game.Game attribute), 6

B

BASE (battleofai.http.GameRoute attribute), 7
 BASE (battleofai.http.IAMRoute attribute), 8
 BASE (battleofai.http.Route attribute), 8
 battleofai (module), 8
 battleofai.abc (module), 3
 battleofai.client (module), 4
 battleofai.config (module), 4
 battleofai.enums (module), 5
 battleofai.errors (module), 5
 battleofai.game (module), 6
 battleofai.game_session (module), 6
 battleofai.game_types (module), 7
 battleofai.http (module), 7
 BOAException, 5
 bucket (battleofai.http.Route attribute), 8

C

callback() (battleofai.client.Client method), 4
 cancel() (battleofai.game_session.GameSession method), 6
 check_and_update_token() (battleofai.client.Client method), 4
 client (battleofai.game_session.GameSession attribute), 7
 Client (class in battleofai.client), 4
 ClientException, 5
 close() (battleofai.client.Client method), 4
 close() (battleofai.http.HTTPClient method), 7
 Config (class in battleofai.config), 4
 Core (class in battleofai.game_types), 7
 create() (battleofai.game.Game class method), 6
 create_game() (battleofai.abc.Match class method), 3
 create_game() (battleofai.http.HTTPClient method), 7
 create_session() (battleofai.client.Client method), 4
 current_board (battleofai.game.Game attribute), 6

F

find_match() (battleofai.game_session.GameSession method), 7
 FINISHED (battleofai.enums.GameState attribute), 5
 Forbidden, 5
 from_json_file() (battleofai.config.Config method), 4
 from_python_file() (battleofai.config.Config method), 5

G

game (battleofai.abc.Match attribute), 3
 Game (class in battleofai.game), 6
 GameSession (class in battleofai.game_session), 6
 GameRoute (class in battleofai.http), 7
 GameState (class in battleofai.enums), 5
 GameType (class in battleofai.abc), 3
 GET (battleofai.http.Method attribute), 8
 get() (battleofai.game.Game class method), 6
 get_all() (battleofai.game.Game class method), 6
 get_callback() (battleofai.client.Client method), 4
 get_game() (battleofai.http.HTTPClient method), 7
 get_games() (battleofai.http.HTTPClient method), 7
 get_symbol() (battleofai.game_types.Core static method), 7

H

history (battleofai.game.Game attribute), 6
 HTTPClient (class in battleofai.http), 7
 HTTPException, 5

I

IAMRoute (class in battleofai.http), 8
 id (battleofai.game.Game attribute), 6
 is_active (battleofai.abc.Match attribute), 3
 is_running (battleofai.game_session.GameSession attribute), 7

J

join_game() (battleofai.abc.Match method), 3

L

list() (battleofai.game.Game class method), 6
login() (battleofai.client.Client method), 4
login() (battleofai.http.HTTPClient method), 8
LoginFailure, 5

M

MAJOR_PARAMS (battleofai.http.GamesRoute attribute), 7
MAJOR_PARAMS (battleofai.http.IAMRoute attribute), 8
MAJOR_PARAMS (battleofai.http.Route attribute), 8
make_turn() (battleofai.abc.Match method), 3
make_turn() (battleofai.game.Game method), 6
make_turn() (battleofai.game_types.Core method), 7
make_turn() (battleofai.http.HTTPClient method), 8
match (battleofai.game_session.GameSession attribute), 7
Match (class in battleofai.abc), 3
message (battleofai.errors.HTTPException attribute), 5
Method (class in battleofai.http), 8

N

name (battleofai.game.Game attribute), 6
NON_EXISTENT (battleofai.enums.GameState attribute), 5
NotFound, 5

O

on_ready() (battleofai.client.Client method), 4
open_games() (battleofai.abc.Match class method), 3
open_slots (battleofai.game.Game attribute), 6
opponents (battleofai.abc.Match attribute), 3

P

password (battleofai.config.Config attribute), 5
play() (battleofai.abc.Match method), 3
play() (battleofai.game_types.Core method), 7
player_auth (battleofai.http.HTTPClient attribute), 8
players (battleofai.game.Game attribute), 6
POST (battleofai.http.Method attribute), 8
pull() (battleofai.game.Game method), 6

R

ready (battleofai.abc.Match attribute), 3
register_callback() (battleofai.client.Client method), 4
register_client() (battleofai.game_session.GameSession method), 7
register_player() (battleofai.http.HTTPClient method), 8
register_session() (battleofai.client.Client method), 4
rejoin_ongoing_game() (battleofai.abc.Match method), 4
request() (battleofai.http.HTTPClient method), 8
response (battleofai.errors.HTTPException attribute), 5

result (battleofai.game_session.GameSession attribute), 7
Route (class in battleofai.http), 8
run() (battleofai.client.Client method), 4
run() (battleofai.game_session.GameSession method), 7
run_for_client() (battleofai.game_session.GameSession method), 7
run_match() (battleofai.game_session.GameSession method), 7
run_sessions() (battleofai.client.Client method), 4

S

sessions (battleofai.client.Client attribute), 4
set_credentials() (battleofai.client.Client method), 4
set_match() (battleofai.game_session.GameSession method), 7
setup() (battleofai.game_session.GameSession method), 7
start() (battleofai.client.Client method), 4
STARTED (battleofai.enums.GameState attribute), 5
state (battleofai.game.Game attribute), 6
symbol (battleofai.game_types.Core attribute), 7

T

task (battleofai.game_session.GameSession attribute), 7
token (battleofai.http.HTTPClient attribute), 8

U

Unauthorized, 6
update() (battleofai.game.Game method), 6
username (battleofai.config.Config attribute), 5

V

validate_token() (battleofai.http.HTTPClient method), 8

W

WAITING (battleofai.enums.GameState attribute), 5
winning_player (battleofai.game.Game attribute), 6
won (battleofai.abc.Match attribute), 4