# Othello-CLI

**Amos Law**

**May 03, 2020**

# CONTENTS

CHAPTER

ONE

# LICENSE

MIT License

Copyright (c) 2020 Amos Law

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.

A command-line interface for Othello! Either player can be human or a bot. A list of available agents is available via the help command.

CHAPTER

TWO

# INSTALLATION

Othello-CLI requires Python 3.7+. To install the Othello project, run this command in your terminal:

```
$ pip install othello-cli
```

CHAPTER

# THREE

# USAGE

```
$ othello [OPTIONS]
```

**-b** <agent>, **--black** <agent>
: The agent type for black.

**-w** <agent>, **--white** <agent>
: The agent type for white.

**--version**
: Display the version and exit.

**--help**
: Display a short help message and exit.

CHAPTER

# FOUR

# REFERENCE

## 4.1 Game



These are the modules in the *othello.game* subpackage that deal with tracking the state of an Othello game.

### 4.1.1 othello.game.game_state

Game state module.

**class** othello.game.game_state.**GameState**(*board*, *current_player*, *move=None*, *prev_move=None*)

An Othello game state.

**board**
Board instance reflecting current state.

**current_player**
Player whose turn it is.

**last_move**
The last Move played.

**second_last_move**
The second to last Move played.

**apply_move**(*move*)
Apply Move to GameState.

**Parameters** **move** (*Move*) – Move to be applied.

**Return type** *GameState*

**Returns** New GameState instance.

**Raises** ***InvalidMoveError*** – If the move is illegal given the game state.

**is_over**()
Returns whether game is over given the current state.

**Return type** bool

**Returns** True if game is over, False otherwise.

**legal_moves**()
Returns list of legal plays for the current player.

**Return type** List[*Point*]

**Returns** List of Point instances that are legal plays.

**classmethod new_game**(*board_size=8*)
Constructor for initial game state.

**Parameters** **board_size** (int) – Board size.

**Return type** *GameState*

**Returns** Initial game state.

**winner**()
Returns whether game is over.

If a player has resigned, the other player is the winner. If neither player has resigned, the winner is the player with the most discs on the board. If they have an equal count, it is a draw.

**Return type** Optional[*Player*]

**Returns** Player who won if there is a winner, None if it is a draw.

**exception** othello.game.game_state.**InvalidMoveError**
Raised when passing while there are legal disc placements.

### 4.1.2 othello.game.board

Board module.

**class** othello.game.board.**Board**(*size*)
Othello board.

Board instances keep track of what color discs are on what locations.

**size**
Board size.

**count_discs**(*player*)
Count discs on board corresponding to the given player.

**Parameters** **player** (*Player*) – Player whose discs are counted.

**Return type** int

**Returns** Disc count.

**get_valid_moves**(*player*)
Get valid moves and their captures.

**Parameters** **player** (*Player*) – Player whose moves are considered.

**Return type** Dict[*Point*, List[*Point*]]

**Returns** Dictionary mapping valid disc placements to discs captured by the move.

**place_disc**(*player*, *point*)
Place disc corresponding to player at the given point.

If the move is valid, the Board instance will be updated to reflect the disc being at that point and all outflanked discs being reversed.

**Parameters**

- **player** (*Player*) – The disc placed corresponds to this player.
- **point** (*Point*) – The disc will be placed at this point.

**Raises** ***InvalidDiscPlacementError*** – If disc cannot legally be placed at the given point.

**Return type** None

**exception** othello.game.board.**BoardSizeError**
Raised when initializing Board object with invalid board size.

**exception** othello.game.board.**InvalidDiscPlacementError**
Raised when placing disc on Board at an invalid location.

## 4.2 Types



These are the modules in the *othello.game* subpackage that deal with types.

### 4.2.1 othello.game.player

Player module.

**class** othello.game.player.**Player**
Player class representing a playable color.

**property other**
Returns the other player color.

**Return type** *Player*

**Returns** Player instance of opposite color.

### 4.2.2 othello.game.disc

Disc module.

**class** othello.game.disc.**Disc**
Disc object.

othello.game.disc.**get_disc**(*player=None*)
Returns Disc object based on given player.

Returns the Disc object corresponding to the player color if a Player type is passed, or a blank Disc if no argument is passed.

**Parameters** **player** (Optional[*Player*]) – Disc will be made for this player (blank disc if None).

**Return type** *Disc*

**Returns** Corresponding Disc object.

### 4.2.3 othello.game.point

Point module.

**class** othello.game.point.**Point**
Point on board.

**property col**
Alias for field number 1

**property row**
Alias for field number 0

### 4.2.4 othello.game.move

Move module.

**exception** othello.game.move.**InvalidMoveError**
Raised upon invalid Move object initialization.

**class** othello.game.move.**Move**(*point=None*, *is_pass=False*, *is_resign=False*)
Represents a player's move.

Should be instantiated with one of the defined alternative constructors.

**point**
Point to play if move is play, otherwise None.

**is_play**
A boolean indicating if the move is play disc.

**is_pass**
A boolean indicating if the move is pass.

**is_resign**
A boolean indicating if the move is resign.

**classmethod pass_turn**()
Constructor for pass type Move.

**Return type** *Move*

**Returns** A pass type Move instance.

**classmethod play**(*point*)
Constructor for play type Move.

**Parameters** **point** (*Point*) – A Point representing where to play.

**Return type** *Move*

**Returns** A play type Move instance.

**classmethod resign**()
Constructor for resign type Move.

**Return type** *Move*

**Returns** A resign type Move instance.

## 4.3 Agents



The *othello.agent* package contains an abstract base class *Agent* that all agent classes should be derived from. New agent types can be registered as plugins.

### 4.3.1 othello.agent.base

Abstract base class for agents.

**class** othello.agent.base.**Agent**
Agent abstract base class.

**abstract select_move**(*game_state*)
Select move given game state.

**Return type** *Move*

### 4.3.2 othello.agent.human

Human agent module.

**class** othello.agent.human.**Human**
Human agent.

**static notation_to_move**(*move_input*)
Convert notation string to Point instance.

**Parameters** **move_input** (str) – Notation to be converted.

**Return type** *Move*

**Returns** Move.

**static point_to_notation**(*point*)
Convert Point instance to notation string.

**Parameters** **point** (*Point*) – Point to be converted.

**Return type** str

**Returns** Notation.

**Raises** ***NotationError*** – No notation exists for given point.

**select_move**(*game_state*)
Select move using user input.

**Parameters** **game_state** (*GameState*) – Current game state.

**Return type** *Move*

**Returns** Move instance.

**static validate_input**(*move_input*)
Validate human input.

**Parameters** **move_input** (str) – Notation input.

**Raises** ***InvalidInputError*** – Input is not valid move notation.

**Return type** None

**exception** othello.agent.human.**InvalidInputError**
Exception class for when input is not a valid move selection.

**exception** othello.agent.human.**NotationError**
Exception class for when game notation related errors.

### 4.3.3 othello.agent.random_bot

Random bot agent module.

**class** othello.agent.random_bot.**RandomBot**
RandomBot agent.

**select_move**(*game_state*)
Choose a random valid move.

**Parameters** **game_state** (*GameState*) – Current game state.

**Return type** *Move*

**Returns** Move instance.

# PYTHON MODULE INDEX

## O

# INDEX

## Symbols



## A



## B



## C



## D



## G



## H



## I



## L



## M



## N



## O

## P



## R



## S



## V



## W