# Introduction

The following skeletal increment, provides documentation for both the Autonomous Player class, supporting classes and the tests that have been written and thus far passed. Documentation was generated using Python's docstring format using the Sphinx module for Pycharm. Methods shown in documentation provide the ground work for communication between the server and autonomous player objects.

# Automotan Documentation

The documentation genearted by Sphinx includes both the Player class, supporting modules and unit tests writen to exercise the code.

## auto package

## \*\*\*\*\*\*\*\*\*\*\*\*

### Subpackages

### ===========

#### \* auto.tests package

##### \* Submodules

\* auto.tests.autoclient\_integration\_test module

\* auto.tests.autoclient\_unit\_test module

\* Module contents

Submodules

==========

auto.automaton module

=====================

class class auto.automaton.Player(available\_players\_list)

Bases: "builtins.object"

The player class to be instantiated for each requested Clue-Less computer/AI player

get\_location

Get the location of this player and store it as an instance variable for tracking location.

Returns:

current location

Return type:

str

get\_starting\_location(selected\_player)

Gets the starting position of the selected player.

:param selected\_player <string> :return: the selected player's starting hallway position

:rtype : str

notify\_card\_revealed(response: dict)

Reveal that a card was revealed and optionally, depending on who asked, the card that was revealed.

Parameters:

\*\*response\*\* -- dictionary containing a response with three

keys match, card and player\_name. Match is a Boolean,

player\_name is a string and card is a string. Card will

contain a value only if the server is sending a response to

the Autonomous player who asked if another player had a card.

player\_count = 0

question(question\_asked: dict)

Ask this computer player a question about whether they have one of three cards :param question\_asked: dictionary containing two

keys: player\_name <string> and cards<list>. Cards should

contain three string values.

receive\_cards(dealt\_cards: list)

Receive a set of cards from the dealer and store them.

Parameters:

\*\*dealt\_cards\*\* -- list

Returns:

dealt cards

Return type:

list<str>

Raise:

IndexError if # of cards not between 3 and 6 ValueError if

cards dealt are not in Cards data structure

take\_turn(game\_state: dict)

Take a turn given the game state.

Parameters:

\*\*game\_state\*\* -- dictionary containing the state of the game

position, suggestion and accusation keys. Key values are

dictionaries as described in the interface specification.

Returns:

dictionary containing a moveto, suggest and accuse key.

Suggest and accuse values are lists of string

auto.autoserver module

======================

auto.board module

=================

class class auto.board.Board

Bases: "builtins.object"

The Clue-Less board graph

neighborhood(node, n)

Determines what nodes are n away from the target node.

Parameters:

\* \*\*node\*\* -- name of the target node <string>

\* \*\*n\*\* -- number of nodes away from the target node <int>

Returns:

set of nodes set<string>

auto.pad module

===============

class class auto.pad.Pad

Bases: "builtins.object"

A Clue-Less note pad to allow a computer player to track game activity.

has\_card(card)

Record that another user has a card

Parameters:

\*\*card\*\* --

Raises ValueError:

if card specified is invalid

player\_revealed\_card(cards\_asked)

Record that a player told another player that they had one of the cards suggested.

Parameters:

\*\*cards\_asked\*\* -- dictionary containing weapon, room and

suspect keys

:exception ValueError if three valid cards are not specified