Chatwolf

Release 0.1.0

Max Schmit

May 12, 2020

CONTENTS:

CHAPTER

ONE

INTRO

CHAPTER

TWO

CHATWOLF

2.1 chatwolf package

2.1.1 chatwolf.game module

class chatwolf.game.**Game**(*sk*, *chatid*, *numwerewolfs*, *amor=False*, *witch=False*, *prostitute=False*, *visionary=False*, *lang='en'*, *wait_mult=1*, *log_dir='C:\\Users\\Max\\AppData\\Local/chatwolf/logs'*, *bkp_dir='C:\\Users\\Max\\AppData\\Local/chatwolf/bkp'*, *do_debug=True*)

Bases: object

This is the main game class, that starts all the other necessary classes to play!

sk
logged in Skype Object of the Game-master

Type skpy.Skype

chatid
chatid of the group-chat, where all players and the game-master are in

Type str

chat
the group chat

Type SkypeChat

skc
object of the SkypeCommands class for the group chat

Type *SkypeCommands*

numwerewolfs
number of werewolfs for the game

Type int

amor
should the amor role be in the game

Type bool

witch
should the witch role be in the game

Type bool

prostitute
should the prostitute role be in the game

Type bool

visionary
should the visionary role be in the game

Type bool

lang
language to use for the messages of the Game-master

Type str

wait_mult
multiplier for the waiting seequences

Type int

log_dir
directory path as str for the logging file

Type str

logfilename
filepath of the logger file

Type str

bkp_dir
directory path as str for the backup file

Type str

do_debug
should a debug logging file be created

Type bool

starttime
starttime of the game (time when the Game object was created)

Type datetime

nn
number of nights played

Type int

nd
number of days played

Type int

log
the Logger of the game

Type Logger

players
list of all players of the game

Type list of Players

roles
list of all the roles in the game

Type list of Roles

bkp()
Backup the game.

continue_bkp()
Continue a game that was loaded from a backup-file.

day()
Do a day phase!

Does: ask whom to kill this day if game not over, start a night phase by calling Game.night()

dist_roles()
Distribute the roles to the players.

end()
End the game!

get_alive()
Get a list of players that are alive!

Returns list of players that are alive.

Return type list of Player

get_alive_string(*noone=True*)
Get a list of players that are still alive as string entries with their number!

Keyword Arguments **noone** (*bool, optional*) – True: add "0: noone" to the list; False: only players without "0: noone" . Defaults to True.

Returns list with one entry per player, each entry is the number in the alive list + 1 and the name of the player

Return type list of str

get_players_role(*all=True*)
Get a list of all players with their roles!

Keyword Arguments **all** (*bool, optional*) – True: every player of the game is listed; False: only the living players are listed . Defaults to True.

Returns a list with one entry per player with: "name (role)"

Return type list of str

get_roles()
Get a list of the activated roles of the game.

Returns list of the activated roles of the game

Return type list of str

is_end()
Check if game is over!

Returns

True: game is over, on party won; False: Noone won yet, the game is still on

Return type bool

static load_bkp(*filepath*)
Load a backup-file.

Parameters **filepath** (*str*) – filepath of the backup-file to be loaded

Returns the old Game object

Return type *Game*

msg(*filename*, *line='all'*)
Get the coresponding message in the selected language.

Parameters **filename** (*str*) – the name of the message file, e.g. "greeting_all" for the first group message, this file needs to exist at least in the "msg/en/" folder

Keyword Arguments **line** (*str or int, optional*) – specify if the whole message should be returned ("all") or only a specific line(int) . Defaults to "all".

Returns message in the selected language (self.lang) or in english if there is no translation

Return type str

night()
Do a night phase.

Does: create a Nightaction object as na call every Role.night(na) resume the night if game not over, start a day phase by calling game.day()

restart()
Start a new game with the same settings.

save_config()

start()
Start the game!

does: check if players did already accept the game-master as contact send greeting to the group distribute roles by calling Game.dist_roles() inform players of them, by calling Role.greeting() start first day

2.1.2 chatwolf.gui module

class chatwolf.gui.**GUI**
Bases: tkinter.Tk

main class for the Graphical User interface

use GUI().mainloop() to start the GUI and play the game

check_e_bkp_dir()

check_e_log_dir()

check_e_numwerewolfs()

check_e_wait_mult()

check_lb_chats()

check_sk()

check_start()

click_about()

click_b_bkp()

click_b_login()

dict_chats()

fill_chatid()

get_chatid()

get_dir(*entry_widget*)

list_chatid()

login_succes()

start_game()

start_w_run()

static w_error(*msg*)

class chatwolf.gui.**TlBkp**(*root*)
Bases: tkinter.Toplevel

check_e_bkp_file()

check_login()

click_b_login()

get_bkp_file()

login_succes()

restart_bkp()

class chatwolf.gui.**TlLog**(*root*)
Bases: tkinter.Toplevel

login_skype()

login_skype_token()

2.1.3 chatwolf.nightactions module

class chatwolf.nightactions.**Nightactions**(*alive*, *game*, *noone=True*)
Bases: object

Class to log all the actions that happen in the night and resume.

game
the main Game object

Type *Game*

alive
list of players that are still alive

Type list of Players

alive_string
list of players, that are still alive as "id: Name" id is place in alive list + 1

Type list of str

lskill
list of one bool for every player if (s)he got killed in the night e.g. lskill[1] says if player[1] got killed

Type list of bool

lstogether
list of players ids that stayed together during the night. always as tuple of two ids, first one is the player who stays at home

Type list of tuple of int

finish_night()
Finish the night and get the name(group) of the plaers that died.

Returns A list of all the players that died this night as name(group)

Return type list of str

get_killed_id()
Get the id of the killed player.

Returns id of the killed player

Return type int

kill(*player_number*)
Kill a player.

Parameters **player_number** (*[type]*) – number of the player in the alive_string

save(*player_number*)
Save a player.

Parameters **player_number** (*[type]*) – number of the player in the alive_string

together(*player_home_number*, *player_visit_number*)
Set 2 people together for this night.

Parameters

- **player_home_number** (*int*) – the number in the alive_string of the player, who's at home
- **player_visit_number** (*int*) – the number in the alive_string of the player, who's visiting the other

2.1.4 chatwolf.player module

class chatwolf.player.**Player**(*id*, *game*)
Bases: object

Class for every player.

chatid
chatid of the corresponding skpy.SkypeSingleChat of the player

Type str

id
Skype id of the player

Type str

game
the main Game object

Type *Game*

chat
the single chat of the player

Type SkypeChat

skc
object of the SkypeCommands class for the single chat of the player

Type *SkypeCommands*

name
Name of the player

Type str

alive
True: the player is alive ; False: the player is dead

Type bool

love
True: the player is in love with someone

Type bool

lover
The player (s)he is in love with

Type *Player*

role
The role the player has got for the game

Type Role

die(*answer=True*)
The player dies.

Keyword Arguments **answer** (*bool, optional*) – should the methode return the name and the group of the player e.g. True: the methode returns "name (group)" . Defaults to True.

Returns "name (group)" of the player or None if the answer argument is False

Return type str or None

get_name_group()
Get a string with the name and the group of the player.

Returns "name (group)" of the player

Return type str

get_name_role()
Get a string with the name and role of the player.

Returns "name (role)" of the player

Return type str

love_arrow(*lover*)
Throw an arrow at this player, so (s)he fells in love.

Parameters **lover** (Player) – The player (s)he fells in love with

2.1.5 chatwolf.roles module

class chatwolf.roles.**Amor**(*player*, *game*)
Bases: chatwolf.roles.Villager

Class for the Amor role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

greeting()
inform player about their role and give amor the oportunity to throw his arrow

name = 'Amor'

class chatwolf.roles.**Prostitute**(*player*, *game*)
Bases: chatwolf.roles.Villager

Class for the Prostitute role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

name = 'Prostitute'

night(*nightactions*)
Do the Prostetutes night phase.

ask where (s)he wants to stay

Parameters **nightactions** (Nightactions) – log of all the actions that happen(d) in the night

class chatwolf.roles.**Role**(*player*, *game*)
Bases: object

Main class for the roles.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

get_names()
Get the names of the players of this role.

Returns list of all the names of the roles players

Return type list of str

greeting()
Inform players about their role.

group = 'not set'

msg_group_night()
Send a notification to the group chat, which role got called.

name = 'not set'

night(*nightactions*)
Do the corresponding night phase.

Parameters **nightactions** (Nightactions) – log of all the actions that happen(d) in the night

class chatwolf.roles.**Villager**(*player*, *game*)
Bases: chatwolf.roles.Role

Class for the Villager role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

group = 'Villager'

name = 'Villager'

class chatwolf.roles.**Visionary**(*player*, *game*)
Bases: chatwolf.roles.Villager

Class for the Visionary role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

name = 'Visionary'

night (*nightactions*)
Do the visionarys night phase.

ask whome (s)he wants to see tell him/her the group of this player

Parameters **nightactions** (Nightactions) – log of all the actions that happen(d) in the night

class chatwolf.roles.**Werewolf**(*player*, *game*)
Bases: chatwolf.roles.Role

Class of the werewolf role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

group = 'Werewolf'

name = 'Werewolf'

night (*nightactions*)
Do the Werewolfs night phase.

ask whome to kill this night

Parameters **nightactions** (Nightactions) – log of all the actions that happen(d) in the night

class chatwolf.roles.**Witch**(*player*, *game*)
Bases: chatwolf.roles.Villager

Class for the Witch role.

name
the name of the role

Type str

group
the name of the group "Werewolf"/"Villager"

Type str

player
all the players that belong to this role

Type list of Player

game
the main Game object

Type *Game*

chatid
SkypeChat id of the player(s) chat

Type str

game
the main Game object

Type *Game*

chat
group/single SkypeChat of the player(s)

Type SkypeChat

skc
object of the SkypeCommands class for this role

Type *SkypeCommands*

elixier
True: the witchs elixier is still available False: the witchs elixier got already used

Type bool

poison
True: the witchs elixier is still available False: the witchs elixier got already used

Type bool

greeting()
Inform player about their role and initialize the poison and elixier.

name = 'Witch'

night(*nightactions*)
Do the witchs night phase.

tell her whos going to die ask if he wants to save, by using her elixier ask if he wants to kill someone, by using her poison

Parameters **nightactions** (Nightactions) – log of all the actions that happen(d) in the night

2.1.6 chatwolf.skypecommands module

class chatwolf.skypecommands.**SkypeCommands**(*chatid*, *game*, *tokenFile='C:\\Users\\Max\\AppData\\Local/chatwolf/temp/token.txt'*)

Bases: skpy.main.SkypeEventLoop

Class to ask players for answers in Skype.

chatid
chatid of the corresponding chat

Type str

game
the main Game object

Type *Game*

chat
the group chat

Type SkypeChat

ask(*command*, *alive=[None]*, *num_ids=1*, *min_id=0*)
Ask for an answer in the corresponding chat.

Parameters **command** (*str*) – command to ask for, e.g. "kill" for "kill: number": return int or "bool" for "yes/no": return bool or "name" for "name:": retrun str

Keyword Arguments

- **alive** (*list of Player, optional*) – all the alive Players that are at disponible . Defaults to [None].
- **num_ids** (*int, optional*) – number of ids that must be asked for and returned . Defaults to 1.
- **min_id** (*int, optional*) – the smallest id possible to choose from, basicaly if theire is an id 0 for "noone" disponible . Defaults to 0.

Returns

either the number(s) of the corresponding player(s) (alive[return-1]) or a bool, depending on the command or a name(str), if command = "name"

Return type int or bool or str

get_bool(*msg*)
Check if the message received was a "yes/no" answer and return it.

Parameters **msg** (*str*) – the message text someone send to the chat

Returns

the answer to the question "yes":True; "no":False or None if the message wasn't a correct answer

Return type bool or None

get_id(*msg*, *command*, *alive=[None]*, *num_ids=1*, *min_id=0*)
Check the message for an id and return it if the message was right.

Parameters

- **msg** (*str*) – the message text someone send to the chat
- **command** (*str*) – command that was asked for, e.g. “kill” for “kill: number”

Keyword Arguments

- **alive** (*list of Player, optional*) – all the alive Players that are at disponible . Defaults to [None].
- **num_ids** (*int, optional*) – number of ids that must be asked for and returned . Defaults to 1.
- **min_id** (*int, optional*) – the smallest id possible to choose from, basicaly if theire is an id 0 for “noone” disponible . Defaults to 0.

Returns

the number(s) of the corresponding player(s) (alive[return-1]) or None if the message wasn’t a correct answer

Return type int or None

get_name(*msg*)

Get the name the player sended.

Parameters **msg** (*str*) – the message text someone send to the chat

Returns

the Name entered or None if the message was not a correct answer

Return type str or None

2.1.7 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX