Erkle is an event-driven IRC library for Python 3. Functions can be "hooked" to a specific event by using a function decorator named **hook**.

Requirements

Erkle uses, for the most part, only modules that are included by default with Python. To use SSL to connect to IRC servers, however, the *pyOpenSSL*[1] library must be installed. To install this library via pip, execute this command:

```
pip install pyOpenSSL
```


1 https://www.pyopenssl.org

Example usage

```
from erkle import *

@hook.event("welcome")
def welcome(connection):
  connection.join("#erklelib")

@hook.event("join")
def cjoin(connection,nickname,host,channel):
  connection.msg(channel,"Hello world!")
  connection.quit()

bot = ErkleClient("mybot","mybot","Erkle Bot","irc.efnet.org",6667)
bot.connect()
```

ErkleClient

ErkleClient is an object that creates and manages an IRC connection. *ErkleClient()* can take seven arguments:

Table 1: Arguments to ErkleClient()

Argument	Type	Description
`nickname`	*string*	Sets the nickname the IRC client connection will use.
`username`	*string*	Sets the username the IRC client connection will use.
`realname`	*string*	Sets the realname the IRC client connection will use.
`server`	*string*	Sets the IP/hostname of the IRC server to connect to.
`port`	*integer*	Sets the port on the IRC server to connect to. *Default: 6667*
`password`	*string*	Sets the password the IRC client connection will send to the server if required. *Default: empty string*
`SSL`	*boolean*	Sets whether to use SSL to connect to the IRC server; set to True to use SSL. *Default: False*

Once the *ErkleClient* object is created, use the **`connect()`** function to cause the object to connect to the IRC server. Once connected, *ErkleClient* has 7 other methods that can be used to send messages or perform actions on the IRC server. Optional arguments are in italics:

Table 2: ErkleClient methods

Method	Arguments	Description
`connect`	*None.*	Connects to the IRC server.
`send`	• **data** (string)	Sends a "raw" message to the IRC server; the message will not be processed in any way before being sent.
`msg`	• **target** (string) • **message** (string)	Sends a chat message to a channel or user.
`action`	• **target** (string) • **message** (string)	Sends a CTCP action message to a channel or user.
`notice`	• **target** (string) • **message** (string)	Sends a notice to a user or channel.
`join`	• **channel** (string) • ***key*** (string)	Joins a channel.
`part`	• **channel** (string) • ***reason*** (string)	Leaves a channel.
`kick`	• **target** (string) • **channel** (string) • ***reason*** (string)	Kicks a user from a channel (the client must be a channel operator in the channel).
`quit`	• ***reason*** (string)	Disconnects from the IRC server.

ErkleClient also has a number of attributes that store information about the server and client. Not all of these values will be available immediately; the values are populated as the server sends the appropriate data to the client. Most of these values should be available by the time the **`welcome`** event is triggered.

Table 3: ErkleClient attributes

Attribute	Type	Description
`nickname`	*string*	The client's nickname.
`username`	*string*	The client's username.
`realname`	*string*	The client's realname.
`server`	*string*	The server's address.
`port`	*integer*	The server's port.
`password`	*string*	The password used to connect to the server, if there is one.
`usessl`	*boolean*	Whether SSL is being used for this connection or not.
`hostname`	*string*	The server's hostname.
`software`	*string*	The server's software.
`options`	*list*	A list of the options the server supports.
`network`	*string*	The network the server belongs to.
`commands`	*list*	A list of commands supported by the server.
`maxchannels`	*integer*	The maximum number of channels a client can join on the server.
`maxnicklen`	*integer*	The maximum number of characters allowed for a nickname on the server.
`channellen`	*integer*	The maximum number of characters allowed for a channel name on the server.
`topiclen`	*integer*	The maximum number of characters allowed for a channel topic on the server.
`kicklen`	*integer*	The maximum number of characters allowed for a kick message on the server.
`awaylen`	*integer*	The maximum number of characters allowed for an away message on the server.
`maxtargets`	*integer*	The maximum number of targets a message can be sent to on a server.
`modes`	*integer*	The maximum number of channel modes that can be set on the server.
`chantypes`	*list*	What channel types the server uses.

prefix	*list of lists*	What channel status prefixes the server uses; each entry contains a list with the first value being the status type, and the second value being the prefix used for that type.
chanmodes	*list*	What channel modes the server uses.
casemapping	*string*	The casemapping the server uses.
spoofed	*string*	If the client's host is spoofed by the server, then the spoofed host name will be stored here.
users	*dictionary of lists*	An in-memory database of channel users. The dictionary uses channel names for keys, and each dictionary entry is a list of the named channel's users.
topic	*dictionary*	An in-memory database of channel topics. The dictionary uses channel names for keys, and each dictionary entry is a string containing the named channel's topic (or *None* if the topic is blank or unknown).

Hook Decorator

Included with the *ErkleClient* object is the *hook* object. The *hook* object is used to decorate[2] functions that should be executed when specific events occur; this is called "hooking" an event. *hook* exposes one method: **event**. To hook an event, pass the name of the event (as a string) as the only argument to **event** method. For example, to hook an event named "connect", the decorator required would look like:

```
@hook.event("connect")
```

Events can be hooked to an unlimited number of functions. Function hooks will be executed in the order in which they were hooked.

There are 22 IRC events that can be hooked. The hooked function can take a number of different arguments, depending on the event. The first (and sometimes only) argument passed to every hooked function is **connection**, which is the *ErkleClient* object running the IRC connection.

Table 4: Events

Event	Arguments	Description
connect	• **ErkleClient object**	Triggered when the *ErkleClient* object connects to IRC.
motd	• **ErkleClient object** • **message** (string)	Triggered when the server's message of the day (MOTD) is received.

2 https://www.python.org/dev/peps/pep-0318/

`welcome`	• **ErkleClient object**	Triggered when registration with the IRC server is complete.
`nick-taken`	• **ErkleClient object** • **nickname** (string)	Triggered when the client's nickname is already taken during registration; **nickname** contains the new nickname.
`ping`	• **ErkleClient object**	Triggered when the IRC server sends *ErkleClient* a PING command.
`join`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string)	Triggered whenever a user joins a channel the client is in. **nickname** contains the user's nickname, **host** contains the user's host, and **channel** contains the name of the channel joined. This event will trigger when the *ErkleClient* object joins a channel as well.
`part`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string) • **reason** (string)	Triggered whenever a user leaves a channel the client is in. **nickname** contains the nickname of the user, **host** contains the user's host, **channel** contains the name of the channel, and **reason** contains the reason why the user quit. If no reason has been provided, **reason** will be set to **`None`**.
`quit`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **reason** (string)	Triggered when a user quits the IRC server. **nickname** contains the user's nickname, **host** contains the user's host, and **reason** contains the reason why the user quit. If no reason has been provided, **reason** will be set to **`None`**.
`nick`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **new_nickname** (string)	Triggered when a user changes their nickname. **nickname** contains the user's original nickname, **host** contains the user's host, and **new_nickname** contains the user's new nickname.
`names`	• **ErkleClient object** • **channel** (string) • **users** (list)	Triggered when *ErkleClient* generates a list of users in a specific channel. This list will be regenerated every time a user changes their nick, quits IRC, or leaves a channel. **channel** contains the name of the channel, and **users** contains a list of users in that channel. If the server is configured for it, each user entry will contain the user's nickname and host, in the form ***nickname*!*username*@*hostname***; otherwise, the entry will only contain the user's nickname. Channel status symbols ('@' for channel operators, '+' for voiced users, etc.) are prefixed to each user's nickname.
`public`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string) • **message** (string)	Triggered when *ErkleClient* receives a public message. **nickname** contains the sender's nickname, **host** contains the sender's host, **channel** contains the name of the channel the message was sent to, and **message** contains the message contents.

`private`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **message** (string)	Triggered when *ErkleClient* receives a private message. **nickname** contains the sender's nickname, **host** contains the sender's host, and **message** contains the message contents.
`notice`	• **ErkleClient object** • **sender** (string) • **message** (string)	Triggered when *ErkleClient* receives a notice message. **sender** contains the nickname of the sender, and **message** contains the message contents.
`action`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **target** (string) • **message** (string)	Triggered when *ErkleClient* receives a CTCP action message. **nickname** contains the sender's nickname, **host** contains the sender's host, **target** contains the name of the channel or username the message was sent to, and **message** contains the message contents.
`away`	• **ErkleClient object** • **nickname** (string) • **reason** (string)	Triggered when *ErkleClient* receives an "away" notification.
`back`	• **ErkleClient object**	Triggered when *ErkleClient* unsets itself as "away".
`topic`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string) • **topic** (string)	Triggered when *ErkleClient* receives a channel topic update. **nickname** contains the topic setter's nickname, **host** contains the setter's host, **channel** contains channel name, and **topic** contains the channel's topic. If the topic is set to an empty string, **topic** is set to *None*.
`mode`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **target** (string) • **mode** (string)	Triggered when *ErkleClient* receives a channel or user mode change notification. **nickname** contains the mode setter's nickname, **host** contains the setter's host, **target** contains the user or channel the mode applies to, and **mode** contains the modes (and mode parameters) being set. If the mode is being set by the server, **nickname** and **host** will be set to the server's hostname.
`kick`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string) • **target** (string) • **reason** (string)	Triggered whenever *ErkleClient* receives a kick notification. **nickname** contains the kicker's nickname, **host** contains the kicker's host, **channel** contains the channel being kicked from, **target** contains the nickname of the user being kicked, and **reason** contains the reason given for the kick. If no reason is provided, **reason** will be set to *None*.
`kicked`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string) • **reason** (string)	Triggered whenever *ErkleClient* is kicked from a channel. **nickname** contains the kicker's nickname, **host** contains the kicker's host, **channel** contains the channel being kicked from, and **reason** contains the reason given for the kick. If no reason is provided, **reason** will be set to *None*.

`invite`	• **ErkleClient object** • **nickname** (string) • **host** (string) • **channel** (string)	Triggered whenever *ErkleClient* receives a channel invitation. **nickname** contains the inviter's nickname, **host** contains the inviter's host, and **channel** contains the channel *ErkleClient* is being invited to.
`line`	• **ErkleClient object** • **line** (string)	Triggered whenever *ErkleClient* receives a line of data from the server.

ErkleClient's **`connect()`** is a blocking function, so hooked functions should be declared *before* **`connect()`** is called.

Event sets

Erkle contains a few sets of pre-written event handlers; they reside in the **erkle.events** package. To use an event set, simply import it.

Package	**erkle.events.dump**
Hooks	**action**, **away**, **back**, **connect**, **join**, **kick**, **kicked**, **mode**, **motd**, **names**, **nick**, **nick-taken**, **notice**, **part**, **ping**, **private**, **public**, **quit**, **topic**, **welcome**
Description	Prints event-specific data from every **Erkle** event to the console.

Package	**erkle.events.messages**
Hooks	**notice**, **private**, **public**
Description	Prints incoming messages to the console.

Examples

Greeter Bot

Here's an example bot that connect to an IRC server, join a channel, and greet everyone who joins that channel by name:

```
from erkle import *

SERVER = "irc.efnet.org"
PORT = 6667
CHANNEL = "#erklebot"

@hook.event("welcome")
def welcomed(connection):
  connection.join(CHANNEL)

@hook.event("join")
def joined(connection,nickname,host,channel):
  connection.msg("Welcome to "+CHANNEL+", "+nickname+"!")

bot = ErkleClient("greetbot","greetbot","Erkle Bot",SERVER,PORT)
bot.connect()
```

...