



IRC Library for Python 3

<https://github.com/nutjob-laboratories/erkle>

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

Requirements.....	2
<i>Python libraries.....</i>	<i>2</i>
<i>Low Level.....</i>	<i>2</i>
Erkle.....	3
<i>Object creation.....</i>	<i>3</i>
<i>Methods.....</i>	<i>3</i>
<i>Attributes.....</i>	<i>4</i>
Hook decorator and events.....	5
<i>Events.....</i>	<i>6</i>
<i>Event sets.....</i>	<i>8</i>
erkle.events.dump.....	8
erkle.events.messages.....	8
Hook object.....	9
Using hook.threads().....	9
Examples.....	10
<i>Greeter Bot.....</i>	<i>10</i>
License.....	11

Requirements

Erkle uses, for the most part, only modules in the Python standard library. To use [SSL/TLS](#) to connect to IRC servers, however, the [pyOpenSSL](#) library must be installed. To install this library via the Python package installer, [pip](#), execute this command:

```
pip install pyOpenSSL
```

Python libraries

Erkle uses the following modules from the standard library:

- `socket`
- `collections`
- `string`
- `threading`
- `ssl` (only if it is available)

Low Level

To use *Erkle*, understanding IRC and the IRC protocol is a necessity. *Erkle* is designed to be low level, meaning its interface is influenced by the protocol itself. Since there's no syntactic sugar to hide the difficult or complex parts of the protocol, *Erkle* code should be easy to understand if you understand the underlying protocol.

The IRC protocol is defined in a series of RFC documents:

- [RFC 1459](#)
- [RFC 2812](#)

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

Example Erkle usage

Erkle

Object creation

Erkle is an object that creates and manages an IRC connection. *Erkle()* can take eight arguments (see below). Once the *Erkle* object is created, use the **connect()** or **spawn()** method to cause the object to connect to the IRC server.

Argument	Type	Description
nickname	<i>string</i>	Sets the nickname the IRC client connection will use.
username	<i>string</i>	Sets the username the IRC client connection will use.
realname	<i>string</i>	Sets the realname the IRC client connection will use.
server	<i>string</i>	Sets the IP/hostname of the IRC server to connect to.
port	<i>integer</i>	Sets the port on the IRC server to connect to. <i>Default: 6667</i>
password	<i>string</i>	Sets the password the IRC client connection will send to the server if required. <i>Default: None</i>
SSL	<i>boolean</i>	Sets whether to use SSL to connect to the IRC server; set to True to use SSL. <i>Default: False</i>
encoding	<i>string</i>	What string encoding type the server connection uses. <i>Default: utf-8</i>

Methods

Method	Arguments	Description
connect	<i>None.</i>	Connects to the IRC server.
spawn	<i>None.</i>	Spawns a new thread, and connects to IRC using that thread.
send	<ul style="list-style-type: none">• data (string)	Sends a "raw" message to the IRC server; the message will not be processed in any way before being sent.
privmsg	<ul style="list-style-type: none">• target (string)• message (string)	Sends a chat message to a channel or user. This can also be called via an alias: msg()
action	<ul style="list-style-type: none">• target (string)• message (string)	Sends a CTCP action message to a channel or user. This can also be called via an alias: me()
notice	<ul style="list-style-type: none">• target (string)• message (string)	Sends a notice to a user or channel.
join	<ul style="list-style-type: none">• channel (string)• key (string)	Joins a channel.
part	<ul style="list-style-type: none">• channel (string)• reason (string)	Leaves a channel.
kick	<ul style="list-style-type: none">• target (string)• channel (string)• reason (string)	Kicks a user from a channel (the client must be a channel operator in the channel).
ban	<ul style="list-style-type: none">• channel (string)• mask (string)	Bans any user who's nick/host/username matches a mask from a channel (the client must be a channel operator in the channel). See RFC 1459 for more information on masks.

unban	<ul style="list-style-type: none"> • channel (string) • mask (string) 	Removes a channel ban from a channel (the client must be a channel operator in the channel).
lock	<ul style="list-style-type: none"> • channel (string) • key (string) 	Sets a channel key on a channel (the client must be a channel operator in the channel).
unlock	<ul style="list-style-type: none"> • channel (string) • key (string) 	Removes a channel key from a channel (the client must be a channel operator in the channel).
mode	<ul style="list-style-type: none"> • target (string) • mode (string) 	Sets a mode on a channel or user. See RFC 1459 for more information on modes.
invite	<ul style="list-style-type: none"> • user (string) • channel (string) 	Sends a channel invitation to a user.
away	• message (string)	Sets the client to "away" on the IRC server.
back	<i>None.</i>	Sets the client to "back" on the IRC server.
whois	• user (string)	Requests WHOIS data on a user from the server. When the WHOIS data is received, the whois event will be triggered.
list	<i>None</i>	Requests a list of channels from the server. When the channel list is received, the list event will be triggered.
quit	• reason (string)	Disconnects from the IRC server.

Attributes

An *Erkle* 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.

Attribute	Type	Description
nickname	<i>string</i>	The client's nickname.
username	<i>string</i>	The client's username.
realname	<i>string</i>	The client's realname.
server	<i>string</i>	The server's address.
port	<i>integer</i>	The server's port.
password	<i>string</i>	The password used to connect to the server, if there is one.
usessl	<i>boolean</i>	Whether SSL is being used for this connection or not.
hostname	<i>string</i>	The server's hostname.
software	<i>string</i>	The server's software.
options	<i>list</i>	A list of the options the server supports.
network	<i>string</i>	The network the server belongs to.
commands	<i>list</i>	A list of commands supported by the server.
maxchannels	<i>integer</i>	The maximum number of channels a client can join on the server.

maxnicklen	<i>integer</i>	The maximum number of characters allowed for a nickname on the server.
channellen	<i>integer</i>	The maximum number of characters allowed for a channel name on the server.
topiclen	<i>integer</i>	The maximum number of characters allowed for a channel topic on the server.
kicklen	<i>integer</i>	The maximum number of characters allowed for a kick message on the server.
awaylen	<i>integer</i>	The maximum number of characters allowed for an away message on the server.
maxtargets	<i>integer</i>	The maximum number of targets a message can be sent to on a server.
modes	<i>integer</i>	The maximum number of channel modes that can be set on the server.
chantypes	<i>list</i>	What channel types the server uses.
prefix	<i>list of lists</i>	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	<i>list</i>	What channel modes the server uses.
casemapping	<i>string</i>	The casemapping the server uses.
spoofed	<i>string</i>	If the client's host is spoofed by the server, then the spoofed host name will be stored here.
users	<i>dictionary of lists</i>	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	<i>dictionary</i>	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 <i>None</i> if the topic is blank or unknown).

Hook decorator and events

Included with the *Erkle* object is the *hook* decorator. The *hook* decorator is used to decorate¹ 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 the **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.

¹ <https://www.python.org/dev/peps/pep-0318/>

There are 23 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 *Erkle* object running the IRC connection.

Events

Event	Arguments	Description
connect	<ul style="list-style-type: none">• Erkle object	Triggered when the <i>Erkle</i> object connects to IRC.
motd	<ul style="list-style-type: none">• Erkle object• message (string)	Triggered when the server's message of the day (MOTD) is received.
welcome	<ul style="list-style-type: none">• Erkle object	Triggered when registration with the IRC server is complete.
nick-taken	<ul style="list-style-type: none">• Erkle object• nickname (string)	Triggered when <i>Erkle</i> 's nickname is already taken during registration; nickname contains the new nickname.
ping	<ul style="list-style-type: none">• Erkle object	Triggered when the IRC server sends <i>Erkle</i> a PING command.
join	<ul style="list-style-type: none">• Erkle object• nickname (string)• host (string)• channel (string)	Triggered whenever a user joins a channel <i>Erkle</i> 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 <i>Erkle</i> object joins a channel as well.
part	<ul style="list-style-type: none">• Erkle object• nickname (string)• host (string)• channel (string)• reason (string)	Triggered whenever a user leaves a channel <i>Erkle</i> 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	<ul style="list-style-type: none">• Erkle 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	<ul style="list-style-type: none">• Erkle 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	<ul style="list-style-type: none">• Erkle object• channel (string)• users (list)	Triggered when <i>Erkle</i> 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	<ul style="list-style-type: none">• Erkle object• nickname (string)• host (string)• channel (string)• message (string)	Triggered when <i>Erkle</i> 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	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • message (string) 	Triggered when <i>Erkle</i> receives a private message. nickname contains the sender's nickname, host contains the sender's host, and message contains the message contents.
notice	<ul style="list-style-type: none"> • Erkle object • sender (string) • message (string) 	Triggered when <i>Erkle</i> receives a notice message. sender contains the nickname of the sender, and message contains the message contents.
action	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • target (string) • message (string) 	Triggered when <i>Erkle</i> 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	<ul style="list-style-type: none"> • Erkle object • nickname (string) • reason (string) 	Triggered when <i>Erkle</i> receives an "away" notification.
back	<ul style="list-style-type: none"> • Erkle object 	Triggered when <i>Erkle</i> unsets itself as "away".
topic	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • channel (string) • topic (string) 	Triggered when <i>Erkle</i> 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 <i>None</i> .
mode	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • target (string) • mode (string) 	Triggered when <i>Erkle</i> 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	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • channel (string) • target (string) • reason (string) 	Triggered whenever <i>Erkle</i> 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 <i>None</i> .
kicked	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • channel (string) • reason (string) 	Triggered whenever <i>Erkle</i> 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 <i>None</i> .
invite	<ul style="list-style-type: none"> • Erkle object • nickname (string) • host (string) • channel (string) 	Triggered whenever <i>Erkle</i> receives a channel invitation. nickname contains the inviter's nickname, host contains the inviter's host, and channel contains the channel <i>Erkle</i> is being invited to.

whois	<ul style="list-style-type: none"> • Erkle object • nickname (string) • username (string) • host (string) • realname (string) • server (string) • idle (integer) • signon (string) • channels (list) • privileges (string) 	Triggered whenever <i>Erkle</i> receives WHOIS data from the server. nickname contains the user's nickname, username contains the user's username, host contains the user's host, realname contains the user's realname, server contains the server the user is connected to, idle contains the number of seconds the user has been idle, signon contains the timestamp of when the user signed on to the server, channels contains a list of channels (with status) the user is in, and privileges contains any special privileges the user has (or <i>None</i> if the user has none).
list	<ul style="list-style-type: none"> • Erkle object • channels (list of lists) 	Triggered whenever <i>Erkle</i> receives a channel list from the server. Each entry in channels is a list that contains, in this order: <ol style="list-style-type: none"> 0. channel name (string) 1. number of users in the channel (integer) 2. channel topic (string) (<i>None</i> if there's no topic)
line	<ul style="list-style-type: none"> • Erkle object • line (string) 	Triggered whenever <i>Erkle</i> receives a line of data from the server.
error	<ul style="list-style-type: none"> • Erkle object • code (string) • subject (string) • reason (string) 	Triggered whenever <i>Erkle</i> receives an error message from the server. code is the error's code (from the IRC RFC documents), subject is the "subject" of the error (if there is no "target", subject will be set to <i>None</i>), and reason contains a short description of the error.

Erkle'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.

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

<i>Package</i>	erkle.events.messages
<i>Hooks</i>	notice, private, public
<i>Description</i>	Prints incoming messages to the console.

Hook object

The `hook` object also keeps track of any *Erkle* connections created with the `spawn()` method. To retrieve a list of *Erkle* threads, use the `hook.threads()` method:

Method	Arguments	Returns	Description
<code>threads</code>	<i>None</i>	list of lists	Returns a list of all threads created with <i>Erkle</i> 's <code>spawn()</code> method. Each entry in the list is a list that contains, in this order: <ol style="list-style-type: none">0. The <i>Erkle</i> object that belongs to the thread1. The <i>Thread</i> object representing the thread (see the Python documentation for the Threading library).

Using `hook.threads()`

```
for thread in hook.threads():  
    ERKLE_OBJECT = thread[0]  
    ERKLE_THREAD = thread[1]  
  
# Now, do what you want with each thread/object
```

Examples

Greeter Bot

Here's an example bot that connect to an IRC server, join a channel, and greets 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 = Erkle("greetbot", "greetbot", "Erkle Bot", SERVER, PORT)
bot.connect()
```

...

License

MIT License

Copyright (c) 2019 Dan Hetrick

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.