Honeybot Project Walkthrough 🚡

by Abdur-Rahmaan Janhangeer

PYMUG (Python Mauritius User-Group)

pymug.com

github.com/pymug

@pymugdotcom

The HoneyBot IRC project

Honeybot is an IRC bot built in Mauritius

⚠️ You are required to follow along by browsing the source

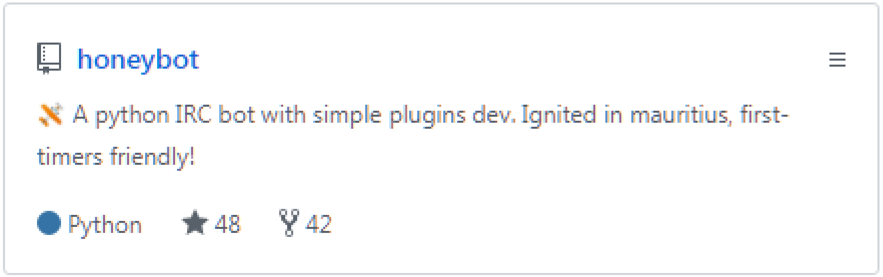
What is IRC?

Internet Relay Chat (IRC) is a protocol that facilitates communication in the form of text. The chat process works on a client/server networking model

What is HoneyBot?

HoneyBot is an IRC bot with simple plugins development

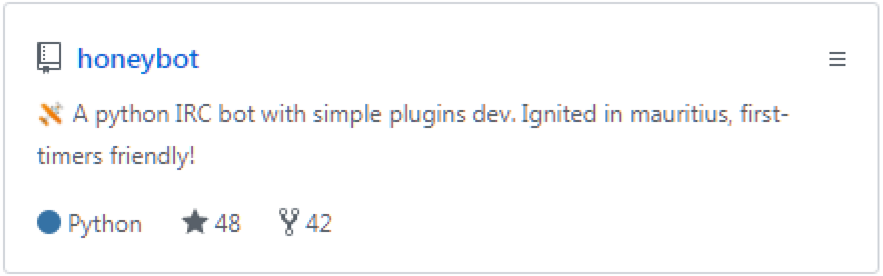
HONEYBOT(C)
IRC BOT



honeybot

A python IRC bot with simple plugins dev. Ignited in mauritius, first-timers friendly!

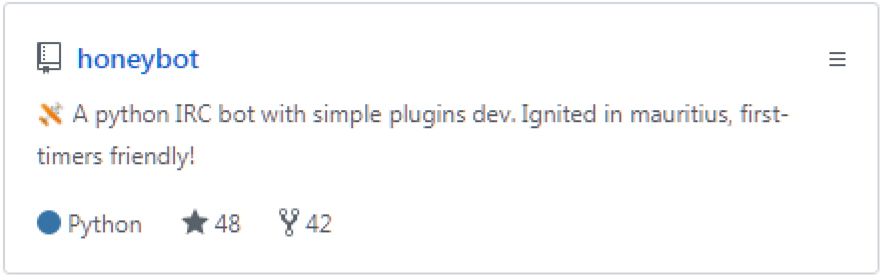
Python ★ 48 42



honeybot

A python IRC bot with simple plugins dev. Ignited in mauritius, first-timers friendly!

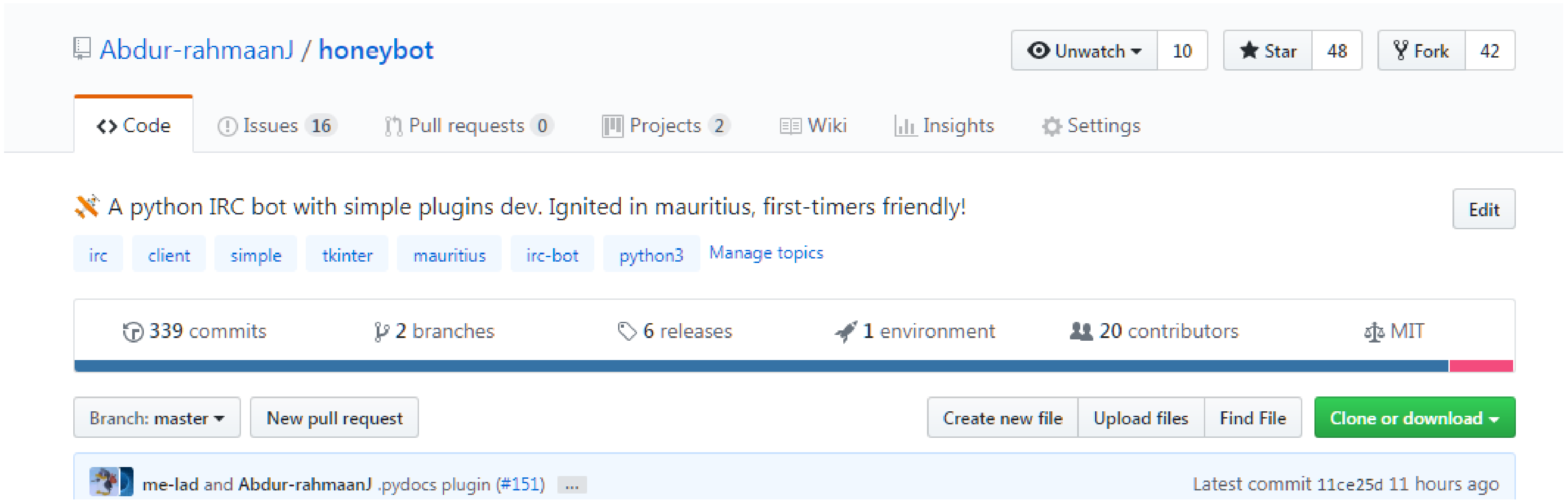
Python ★ 48 42



honeybot

A python IRC bot with simple plugins dev. Ignited in mauritius, first-timers friendly!

Python ★ 48 42


Abdur-rahmaanJ / honeybot

A python IRC bot with simple plugins dev. Ignited in mauritius, first-timers friendly!

irc client simple tkinter mauritius irc-bot python3 Manage topics

339 commits · 2 branches · 6 releases · 1 environment · 20 contributors · MIT

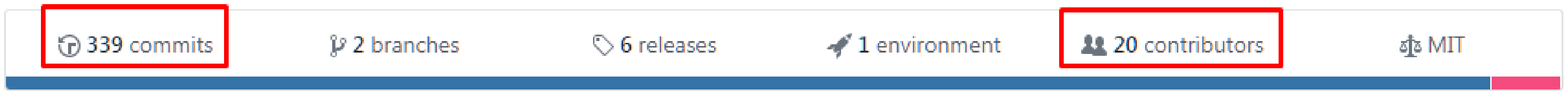
me-lad and Abdur-rahmaanJ .pydocs plugin (#151) ... Latest commit 11ce25d 11 hours ago

Stats

Main stats

- 48 stars
- 42 forks
- kind of special in mauritius

Commit stats



339 commits

2 branches

6 releases

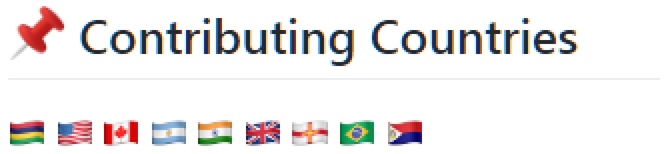
1 environment

20 contributors

MIT

- 339 commits
- 20 contributors
- MIT license

Contributions



📌 Contributing Countries

🇲🇺 🇺🇸 🇨🇦 🇦🇷 🇮🇳 🇬🇧 🇬🇬 🇧🇷 🇸🇽

- contributors from 9 countries

Testimonials

Some testimonials [a]

> With experience in programming in Python, and implementing an SMTP email plugin for a different system, picking up HoneyBot and following the documentation provided for new-comers made it very simple to implement the same SMTP email plugin to the HoneyBot system. This was my first time contributing to an open-source project on GitHub and it was an overall great experience. The welcoming of new contributors and documentation on how to contribute and implement plugins is great for people who have never contributed to a project before, and Abdur-Rahmaan Janhangeer was extremely helpful when answering my questions and helping me along the way.
> -- Tanner Fry

Some testimonials [b]

> HoneyBot is my first time collaborating to an open source project and I'm loving it. Before discovering HoneyBot, I was very intimidated on the idea of working with other people and had no idea what an IRC even was. Now I realize how much fun and rewarding it is to work together on a project with dedicated and friendly individuals. The documentation is easy to follow and everyone is super helpful. I highly recommend any new programmer who want to contribute on an open source project to try out HoneyBot. Personally I enjoy working on this project more than my own schoolwork.
> -- Gico Carlo Evangelista

Some testimonials [c]

> HoneyBot is my first open source project and I had never worked with an IRC before. For school I was required to contribute to projects, but it was always so intimidating to me. I had always heard it gets easier once you've gotten over the fear wall, and that's what HoneyBot did for me. Excellent readme and quick feedback allowed me to make my first plugin. Now I've made many contributions, and look forward to any new issues I can get my hands on. Abdur-Rahmaan Janhangeer has been extremely helpful and I owe him and this project a lot for getting me into the open source world.
> -- Justin Walker

Awesome Plugins

Host of plugins

The project has one of the most awesome collection of plugins.

Plugins Gallery [a]

- 💎 bitcoin by @Macr0Nerd - get price of bitcoin
- 🏺 caesar cipher by @kylegalloway - encode your text
- 0️⃣ calc by @Abdur-rahmaanJ - evaluates maths expressions
- ➗ maths by @Abdur-rahmaanJ - sin cos and the like
- 🏷️ conv sniff by @Abdur-rahmaanJ - set triggers like how many times a word occur for one or more words and send response
- 📼 greet by @Abdur-rahmaanJ - demo plugin

Plugins Gallery [b]

- 💾 installed_modules by @Abdur-rahmaanJ - checks dependencies installed
- 🍩 joke by @Abdur-rahmaanJ, @colbyjayallen - get random joke
- 🥄 self Trivia by @ajimenezUCLA - random trivia
- 🎊 username by @Abdur-rahmaanJ, @sseryani - username generator
- 🖋️ quotes by @German-Corpaz - inspirational quotes
- 📙 dictionary by @iamnishant14 - returns meaning of word

Plugins Gallery [c]

- 💳 password generator by @iamnishant14 - the name tells it all
- 🔬 debug by @Abdur-rahmaanJ - prints all parameters passed to bot
- 📖 wikipedia by @Macr0Nerd - returns a wikipedia article
- 🔄 translate by @a-deeb - google translate plugin
- 🔨 test by @Abdur-rahmaanJ - runs tests
- 🌅 weather by @Macr0Nerd - returns weather info for a given location

Plugins Gallery [d]

- ✉️ mail by @TannerFry - send emails within the chat
- ⛓️ hangman by @JustinWalker4179 - play hangman in the chat
- 🎂 age by @JustinWalker4179 - takes in birthday and outputs age
- ✔️ fact by @JustinWalker4179 - returns a random fact
- 🔍 google by @JustinWalker4179 - returns three search results from google
- 📮 send message by @JustinWalker4179 - sends a message to another channel

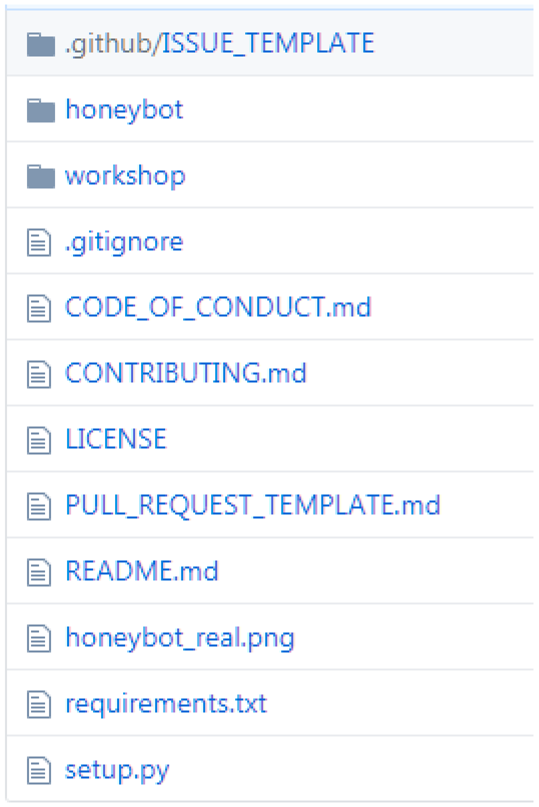
Plugins Gallery [e]

- 📝 log by @RiceAbove - logs the chat into a log file
- 🚪 joins by @RiceAbove - greets everyone who joins the channel
- 📆 date by @RiceAbove - posts the current date
- 🕵 riddle by @AngeloGiacco - returns a random riddle
- 📰 news by @AngeloGiacco - gets the top 10 headlines from bbc world news
- 📝 horoscope by @AngeloGiacco - gets your daily horoscope for your starsign

Plugins Gallery [f]

- 💵 currency converter by @AngeloGiacco - converts currencies
- 🔫 russian_roulette by @AngeloGiacco - may or may not kick you off the channel
- 🏨 monopoly by @AngeloGiacco - Honeybot now supports the world's worst game!
- 🎲 roll by @GlennToms - rolls a dice
- ❓ help by @edumello - show link to plugin's information page
- ✔️ channeljoin by @marceloyb - join command for bot
- 📃 comic by @mboekhold - returns a random comic

Structure


.github/ISSUE_TEMPLATE
honeybot
workshop
.gitignore
CODE_OF_CONDUCT.md
CONTRIBUTING.md
LICENSE
PULL_REQUEST_TEMPLATE.md
README.md
honeybot_real.png
requirements.txt
setup.py

honeybot/honeybot/

- memory
- plugins
- settings
- lab.py
- main.py
- pluginInfo.py
- plugins_info.md
- test.py
- test_plugin_script.py

Brief Overview of IRC

- chat in channels - #python
- you can private message user
- no history

- default connection settings example

```
port - 6667
connection url - chat.freenode.net
```

How an IRC bot works?

How an IRC bot works?

- connect and identify
- constantly check messages
- stay alive
- parse messages
- act in accordance

A look at old

📁 workshop/normalbot/

- 📄 honeybot.py

Basic Mechanisms

How do we connect?

How do we connect?

```
irc.connect((BOT_IRC_SERVER, BOT_IRC_PORT))
```

How do we receive messages?

How do we receive messages?

line 799

```
while 1:
        line = irc.recv(4096)
```

How do we receive messages?

line 768

```
irc = socket.socket()
```

How do we print messages?

line 799

```
while 1:
    line = irc.recv(4096)
    print(line)
```

What do raw IRC messages look like?

What do raw IRC messages look like?

```
:appinv!c5e342c5@gateway/web/cgi-irc/kiwiirc.com/
ip.200.200.22.200 PRIVMSG ##bottestingmu :ef
```

where `ef` is the actual message

How do we stay alive?

How do we stay alive?

the answer is simple, check for ping and return pong

How do we stay alive?

line 803

```
while 1:
        pass
        line = irc.recv(4096)
        print(line)
        pingChecker(line)
```

How do we stay alive?

line 134

```python
def pingChecker(pingLine):
    if pingLine.find(bytes('PING'  ,'utf8')) != -1:
        pingLine = pingLine.rstrip().split()
        if pingLine[0] == bytes("PING"  ,'utf8'):
            irc.send(bytes("PONG "  ,'utf8') +
            pingLine[1] + bytes("\r\n"  ,'utf8')  )
```

How do we send messages?

How do we send messages?

```
irc.send(bytes("PONG "  ,'utf8')
```

Message parsing

Message Parsing

```
while 1:
        pass
        line = irc.recv(4096)
        print(line)
        pingChecker(line)
        if line.find(bytes('PRIVMSG' ,'utf8')) != -1 or line.fi
                messagechecker(line)
                target.write(str(line))
                target.flush()
```

Message Parsing

line 805

```
messagechecker(line)
```

Message Parsing

line 144

```
def messagechecker(msgLine):
    # ...
    completeLine = str(msgLine[1:])
        .replace("'b",'').split(':', 1)
    info = completeLine[0].split()
    message = (completeLine[1].split("\\r")[0]).replace("'b",''
    sender = info[0][2:].split("!", 1)[0]
    refinedmsg = str(message.lower())
    refinedmsgl = len(refinedmsg)

    print("Complete Line-->" + str(completeLine))
    print("Info-->" + str(info))
    print("Message-->" + str(message))
    print("Sender-->" + str(sender) + "\n")
    # ...
```

Message Parsing: Address Deciding

Message Parsing: Address Deciding

```
        address=''
        if (len(info)>=2):
                if ( str(info[2])!= BOT_NICKNAME ):
                        address=str(info[2])
                elif ( str(info[2]) == BOT_NICKNAME):
                        address=str(sender)
```

Message Parsing : Functionality Basics

Message Parsing : Functionality Basics

```
        if(str(message[0:4]) == '.sqr'):
                sqrdata = (message[5:len(message)])
                sqdata = math.sqrt(float(sqrdata))
                sqdatastr = str(sqdata)

                irc.send(bytes('PRIVMSG ' + address + ' :Hey, '
        + str(sender) +' the square root of '+sqrdata+
        + ' is ' +sqdatastr+' !\r\n','utf8'  )  )
```

The IRC Protocol

Message to user

The IRC Protocol

Message to user

```
PRIVMSG <username>: Hi
```

example

```
PRIVMSG mdk: Hi
```

Message to channel

The IRC Protocol

Message to channel

```
PRIVMSG <channelname>: Hi
```

example

```
PRIVMSG #ltch : Hi
```

Join channel

The IRC Protocol

Join channel

```
JOIN <channelname>
```

example

```
JOIN #ltch
```

Leave channel

The IRC Protocol

Leave channel

```
PART <channelname>: <messsage>
```

example

```
PART #ltch: See you soon!
```

Quit IRC

The IRC Protocol

Quit IRC

```
QUIT <channelname>: <messsage>
```

example

```
QUIT #ltch: See you soon!
```

Part 2

honeybot/honeybot/

- memory
- plugins
- settings
- lab.py
- main.py
- pluginInfo.py
- plugins_info.md
- test.py
- test_plugin_script.py

Changes to the new bot

Changes to the new bot

- bot is now OOP
- settings config in files
- plugins

The brain of the bot

The brain of the bot is in main.py

The steps once again

- connect and identify
- constantly check for messages
- stay alive
- parse messages
- act in accordance

Connect

```python
def connect(self):
    self.irc.connect((self.server_url, self.port))
```

identify

```
    def identify(self):
        self.send(self.identify_command())
```

identify command

```python
def identify_command(self):
    return 'msg NickServ identify ' + self.password + ' \r\n'
```

constantly check for messages

```
    def pull(self):
        while self.isListenOn:
            try:
                data = self.irc.recv(2048)
                raw_msg = data.decode("UTF-8")
                msg = raw_msg.strip('\n\r')
```

stay alive

```
    def pull(self):
        while self.isListenOn:
            try:
                data = self.irc.recv(2048)
                raw_msg = data.decode("UTF-8")
                msg = raw_msg.strip('\n\r')
                self.stay_alive(msg)
```

stay alive

```
def stay_alive(self, incoming):
    # ...
    parts = incoming.split(':')
    if parts[0].strip().lower() == 'ping':
        # ...
        self.send(self.pong_return(self.domain))
```

stay alive

```
    def pong_return(self, domain):
        return 'PONG :{}\r\n'.format(domain)
```

parse messages

```
    def pull(self):
        while self.isListenOn:
            try:
                data = self.irc.recv(2048)
                raw_msg = data.decode("UTF-8")
                msg = raw_msg.strip('\n\r')
                self.stay_alive(msg)
                self.core_commands_parse(msg)
```

parse messages

```
    def core_commands_parse(self, incoming):
        self.run_plugins(self.plugins, incoming)
```

parse messages

```
def load_plugins(self, plugins_to_load):
    list_to_add = self.plugins
    logger.info('Loading plugins...')

    to_load = []
    plugs = 'settings/{}.conf'.format(plugins_to_load)
    with open(plugs) as f:
        to_load = f.read().split('\n')
        to_load = list(filter(lambda x: x != '', to_load))
    for file in to_load:
        try:
            module = importlib.import_module('plugins.' +
            file)
        except ModuleNotFoundError as e:
            logger.warning(f"module import error, skipped'
            {e} in {file}")
        obj = module.Plugin
        list_to_add.append(obj)

    self.plugins = list_to_add
    logger.info('Loaded plugins...')
```

parse messages

```
    def run_plugins(self, listfrom, incoming):
        '''
        incoming is the unparsed string. refer to test.py
        '''

        for plugin in listfrom:
            plugin.run(self, incoming, self.methods(),
            self.message_info(incoming), self.bot_info())
```

act in accordance - in enters plugins!

```
class Plugin:
    def __init__(self):
        pass

    def run(self, incoming, methods, info, bot_info):
        try:
            if info['command'] == 'PRIVMSG' and
            info['args'][1] == '.hi':
                methods['send'](info['address'], 'hooo')
        except Exception as e:
            print('woops plug', e)
```

Parameters

Parameter 1 : incoming

incoming contains the raw message in case someone wants to implement something from scratch

```
def run(self, incoming, methods, info, bot_info):
```

incoming

```
:appinv!c5e342c5@gateway/web/cgi-irc/kiwiirc.com/
ip.200.200.22.200 PRIVMSG ##bottestingmu :ef
```

Parameter 2 : methods

```
def run(self, incoming, methods, info, bot_info):
```

methods gives you the following

```
{
 'send_raw': self.send,
 'send': self.send_target,
 'join': self.join,
 'mem_add': self.memory_add_value,
 'mem_rem': self.memory_remove_value,
 'mem_fetch': self.memory_fetch_value
}
```

usage

```
methods['send'](<address>, <message>)
```

Parameter 3 : info

```
def run(self, incoming, methods, info, bot_info):
```

info gives us info about messages coming. helper functions.

```
{
 'prefix': prevent_none(prefix),
 'command': prevent_none(command),
 'args': ['' if e is None else e for e in args],
 'address': prevent_none(address),
 'user': prevent_none(user)
}
```

usage

```
info['address']
```

Parameter 4 : bot_info

```
def run(self, incoming, methods, info, bot_info):
```

bot_info gives information about the bot

```
{
 'name': self.name,
 'special_command': self.sp_command,
 'required_modules': self.required_modules,
 'owners': self.owners
}
```

usage

```
bot_info['name']
```

Plugins Dev Example

Plugins Dev Example

Let's develop a plugin that doubles the number that you input

```
.double 5
```

and the bot sends back

```
10
```

Let's take the example of the greet plugin

```python
class Plugin:
    def __init__(self):
        pass

    def run(self, incoming, methods, info, bot_info):
        try:
            if info['command'] == 'PRIVMSG' and
            info['args'][1] == '.hi':
                methods['send'](info['address'], 'hooo')
        except Exception as e:
            print('woops plug', e)
```

let's modify it

`info['args']` gives us

```
['##bottestingmu', 'ef', 'ab']
```

where index 0 is the channel name and the rest messages.

```
class Plugin:
    def __init__(self):
        pass

    def run(self, incoming, methods, info, bot_info):
        try:
            if info['command'] == 'PRIVMSG' and
            info['args'][1] == '.double':
                to_send = float(info['args'][2]) * 2
                methods['send'](info['address'], to send)
        except Exception as e:
            print('woops plug', e)
```

but `info['args'][0]` can also be simplified to

```
msgs = info['args'][1:]
```

then we use

```
msgs[0]
```

settings/CONNECT.conf

specifies connection details

```
[INFO]

server_url = chat.freenode.net
port = 6667
name = hb_tst5
```

settings/AUTOJOIN_CHANNELS.conf

specifies which channels to join upon connection

```
#ltch
##bottestingmu
```

settings/OWNERS.conf

```
appinventorMu
appinv
```

settings/PLUGINS.conf

specifies which plugin to load

```
age
caesar_cipher
calc
date
debug
dictionary
fact
greet
joke
log
maths
quote
selfTrivia
send_message
help
horoscope
...
```