

---

**BDBF**

***Release 1.2.3***

**Jan 24, 2023**



<b>1</b>	<b>Functions</b>	<b>3</b>
<b>2</b>	<b>Main</b>	<b>5</b>
<b>3</b>	<b>Quick Start</b>	<b>9</b>
3.1	Usage . . . . .	9
<b>4</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>
	<b>Index</b>	<b>15</b>



This is the documentation for BDBF.



# CHAPTER 1

---

## Functions

---

### Functions Docstring

`functions.IntToRgb (RGBint: int)`

Converts a integer color value to a RGB tuple

**Parameters** `RGBint` – int The integer color value.

**Returns** `tuple[int, int, int]` RGB tuple

`functions.RgbToInt (rgb: Tuple[int, int, int])`

Converts a RGB tuple to a integer color value

**Parameters**

- `rgb` – `tuple[int, int, int]` RGB tuple
- `no` – `tuple[int, int, int]` RGB tuple

**Returns** `int` The integer color value.

`functions.embed (title, url=<Mock name='mock.embeds.EmptyEmbed' id='139746189317392'>, description=<Mock name='mock.embeds.EmptyEmbed' id='139746189317392'>, fields=None, image=None, thumbnail=None, author=None, footer=None, color: Union[Tuple[int, int, int], int] = 0)`

Returns discord embed from given parameters

**Parameters**

- `title` – `str` Title of the embed
- `url` – `Optional[str]` url of the title
- `description` – `Optional[str]` Description of the embed
- `fields` – `Optional[List[Tuple[str, str, Optional[bool]]]]` Fields of the embed. A tuple with item 1 being the name of the field, item 2 the value and item 3 weather is inline or not, item 3 is optional
- `image` – `Optional[str]` Url of the embed image

- **thumbnail** – `Optional[str]` Url of the thumbnail image
- **author** – `Optional[Dict[str, str]]` Author of the embed
- **footer** – `Optional[Dict[str, str]]` Footer of the embed
- **color** – `Optional[Union[Tuple[int, int, int], int]]` Color of the embed, either RGB tuple or int

**Returns** `discord.Embed`

`functions.hasLink(text: str)`

Returns if a string contains a link.

**Parameters** **text** – `str` String to check

**Returns** `bool` If the string has a link.



**class** `main.Client` (\*, *loop=None*, \*\**options*)

Discord client class inherited from `discord.Client`. This documentation covers only the changes. For the inherited functions please head to the [discord.py documentation](#)

### Parameters

- **embedFooter** – Optional[dict] The footer of embeds.
- **embedColor** – Optional[tuple[int, int, int]] The color of embeds.
- **botName** – Optional[str] The name of the bot.
- **commandPrefix** – Optional[str] The prefix of all commands.
- **useDefaultHelp** – Optional[bool] Whether to use the default help. Default: True
- **isHelpInline** – Optional[bool] If using the default help. Whether it is inline or not. Default: True
- **logging** – Optional[bool] If message logging is enabled. Default: False
- **caseSensitiveCommands** – Optional[bool] If commands are case sensitive. Default: True
- **sendExceptions** – Optional[bool] If sending exceptions to discord is enabled. Default: True

**command** (*commandName*, \*\**options*)

Wrapper function for making commands.

### Parameters

- **worksOnlyInGuilds** – Optional[List[int]] List of guild where the command will work. List of guild where the command will work.
- **worksOnlyInChannels** – Optional[List[int]] List of channels where the command will work. If not provided will for in all.
- **doesntWorkInGuilds** – Optional[List[int]] List of guilds where the command wont work. List of guild where the command will work.

- **doesntWorkInChannels** – Optional[List[int]] List of guilds where the command wont work. List of guild where the command will work.

```
@client.command("command") def command(message):  
    print(message.content)
```

**embed** (*title*, *\*\*options*)

Returns discord embed from given parameters with automatic footer and color options.

**Parameters**

- **title** – str Title of the embed
- **url** – Optional[str] url of the title
- **description** – Optional[str] Description of the embed
- **fields** – Optional[List[Tuple[str, str, Optional[bool]]]] Fields of the embed. A tuple with item 1 being the name of the field, item 2 the value and item 3 weather is inline or not, item 3 is optional
- **image** – Optional[str] Url of the embed image
- **thumbnail** – Optional[str] Url of the thumbnail image
- **author** – Optional[Dict[str, str]] Author of the embed

**Returns** discord.Embed

**event** (*coro*)

A decorator that registers an event to listen to.

You can find more info about the events on the documentation below.

The events must be a coroutine, if not, TypeError is raised.

```
@client.event  
async def on_ready():  
    print('Ready!')
```

**TypeError** The coroutine passed is not actually a coroutine.

**logCommand** (*function*)

Wrapper fuction for making a logging function. Like

```
@client.logCommand  
def log(command, message, time, state, exception):  
    print(message.content)
```

**logMessage** (*function*)

Wrapper fuction for making a logging function. Like

```
@client.logMessage  
def log(message):  
    print(message.content)
```

**reactionRole** (*msg*, *emoji*, *role*)

Function to add reaction role functions to a message.

**Parameters**

- **msg** – `Union[discord.Message, int]` Message or message id of the message you want to add the reaction role functionality.
- **emoji** – `str` Emoji. If a unicode emoji use it, if a custom emoji use it's name.
- **role** – `int` Role id of the role you want to add to the emoji.



# CHAPTER 3

## Quick Start

This Quick Start assumes you have some knowledge of making a discord bot in python.

### 3.1 Usage

#### 1. Import bdbf

```
import bdbf
```

#### 2. Setup your client

```
client = bdbf.Client(  
    botName="BDBF Bot",  
    commandPrefix="%",  
    embedFooter={  
        "icon_url": "example.com/image.png",  
        "text": "Made using BDBF"  
    },  
    embedColor=(123,123,123))
```

#### 3. Setup the rest like normal

```
@client.event  
async def on_message(message):  
    #some code  
  
client.run("token")
```

#### 4. To make a command use the wrapper client.command

```
@client.command("hi")  
async def hi(message):  
    await message.channel.send(f"Hello {msg.author.mention}")
```

(continues on next page)

(continued from previous page)

```
@client.command("sayHelloTo")
async def sayHelloTo(message, *args):
    await message.channel.send(f"Hello {args[0]}")
```

Beware that the command function has to be a coroutine and that args is a tuple with 0 or 1 items

And that's it, you should now have a working bot.

## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





### **f**

`functions`, 3

### **m**

`main`, 5



### C

`Client` (*class in main*), 5  
`command()` (*main.Client method*), 5

### E

`embed()` (*in module functions*), 3  
`embed()` (*main.Client method*), 6  
`event()` (*main.Client method*), 6

### F

`functions` (*module*), 3

### H

`hasLink()` (*in module functions*), 4

### I

`IntToRgb()` (*in module functions*), 3

### L

`logCommand()` (*main.Client method*), 6  
`logMessage()` (*main.Client method*), 6

### M

`main` (*module*), 5

### R

`reactionRole()` (*main.Client method*), 6  
`RgbToInt()` (*in module functions*), 3