TG-UserBot

Release stable

Nov 12, 2020

First Steps

CHAPTER 1

About the UserBot

TG-UserBot is a modular Python script for Telegram which uses the Telethon library. It is made to help you do your usual client tasks without the hassle and also has some additional useful features. If a command is documented, it will be on the *Available commands* page including it's description with an example if any.

CHAPTER 2

Prerequisites

- Python 3.7.3 or above.
- Git.
- Telegram API key (API ID and hash) from https://my.telegram.org/apps.
- FFMPEG (optional) for YouTube-DL. Guide covers this requirement.
- Some basic understanding would also be helpful.

CHAPTER 3

Where to look

Everything is self-contained into it's own catergory and can be accessed from the sidebar or by using the *Previous* or *Next* buttons at the end of the page. Main topics are listed below accordingly.

- *Beginners guide*: A follow along guide to set-up everything correctly.
- *Available commands*: All the available UserBot commands will be listed on this page.
- *FAQ*: Most commonly asked questions will be answered on here, hopefully you find what you're looking for on here.

Looking for the details on the code itself?

- *Modules*: All the modules (smart plugins) and their functions can be found here with their descriptions.
- *Helper functions*: All the functions used to reduce bloat in multiple modules can be found here with their examples.
- *Utilities*: Utilities used to keep the bot organized and clean.
- *Creating your own command*: Examples to create your own commands can be found here.

If you want to report an issue, have any feedbacks/suggestions or want to discuss something then you may join the support group on Telegram.

3.1 How to get started

Warning: This will not cover the installation of Python or Git.

Assuming you (already) have Python and Git installed, you can go ahead and follow along.

Contents

3.1.1 Cloning the repository

Clone the TG-UserBot repository to your desired location using the *git clone* command.

```
$ git clone https://www.github.com/kandnub/TG-UserBot
```

Once the cloning has finished, change your directory to the cloned folders directory using the *cd* command.

```
$ cd TG-UserBot
```

3.1.2 Configuration file

Now you need configure the `sample_config.ini` with your API ID and hash using your preferred text editor, these are mandatory fields. You may leave the rest as they are or change them to your liking. Once done, rename `sample_config.ini` to `config.ini` and proceed.

```
$ nano sample_config.ini
$ mv sample_config.ini config.ini
```

3.1.3 Installing requirements

The main part has been done, now you only need to install all the requirements with the *pip install* command, this may take a while to finish. Make sure you execute this command in the cloned directory. If you want to install FFMPEG then have a look at the FAQ's *How to install FFMPEG?* answer before proceeding any further.

```
$ pip3 install --user -r requirements.txt
```

3.1.4 Running the UserBot

That's it. You're all done with the jarring work, you can now run the main UserBot script without any worries. If you encounter any problem, consider reffering to the FAQ's page or join the support group.

```
$ python3 -m userbot
```

3.2 FAQ

These are all the Frequently Asked Questions. Hopefully all the answers you're looking for will be on here, if not then you may join the support group and ask. There's always Google/DuckDuckGo or StackOverflow is also a great place as well.

3.2.1 How to install FFMPEG?

Linux users: https://www.ostechnix.com/install-ffmpeg-linux/

Windows users: https://www.wikihow.com/Install-FFmpeg-on-Windows

3.2.2 Why do we need FFMPEG?

You can use YouTube-DL without it as well but you won't be able to use it's full potential. FFMPEG helps you merge multiple formats and embed metadata of the video/audio to the file.

3.3 Heroku Guide

Note: This guide only shows how to generate a Redis session and deploy the bot to Heroku.

We would encourange you to use Termux, an app available on the Play Store. Open the same and follow along.

Contents

3.3.1 Prerequisites

- API ID and Hash from Telegram
- Redis Endpoint and Password from Redis Labs:
  - Get the 30MB free Database from Redis Labs

3.3.2 Cloning the repository

Clone the TG-UserBot repository to your desired location using the *git clone* command.

```
$ git clone https://www.github.com/kandnub/TG-UserBot
```

Once the cloning has finished, change your directory to the cloned folders directory using the *cd* command.

```
$ cd TG-UserBot
```

3.3.3 Installing Python 3

```
$ pkg install python
```

3.3.4 Installing Redis and Telethon

```
$ pip install redis telethon
```

3.3.5 Generating a Redis Session

To generate the session do:

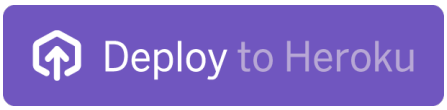
```
$ python3 generate_session.py
```

Simply follow the on-screen instructions and once done you'll get a confirmation message saying:

> Succesfully generated a session for "name"

3.3.6 Depolying to Heroku

To deploy to heroku click on the Auto Deploy button given below:



Fill all the info i.e ENV Vars, double check and click on deploy, thats it you are done! If you get stuck somewhere or get an error while building/deploying the UserBot, don't hesitate to ask us in the support group.

3.4 Available Commands

Here you can find all the commands available in the UserBot including their description, how to use them correctly and what they return.

Note: `.` is the prefix used on here but you can also use `/`, `#`, etc. by setting with with `<prefix>setprefix <newprefix>`. For instance: `.setprefix /` changes `.` to `/`.

Contents

- *Nearest DC*
- *Ping*
- *Ping DC*
- *Profile Picture*
- *Promote*
- *Purge*
- *Regex Ninja*
- *Remind Here*
- *Remind me*
- *Repository*
- *Resolve*
- *Reverse*
- *Remove Background*
- *Remove blacklist*
- *Remove Whitelist*
- *SED*
- *Shiba Inu*
- *Speedtest*
- *Sticker Pack(s)*
- *Temporary Ban*
- *Term*
- *Terminate*
- *Temporary Mute*
- *Un/Dis Approve*
- *Unban*
- *Un Blacklist*
- *Un Mute*
- *Update*
- *Upload*
- *Username*
- *Whitelist*
- *Whitelists*
- *Who is*
- *YouTube-DL*

3.4.1 Main Commands

Help

Lists all the enabled bot commands.

Usage: `.help [all|category|command (dev)]`

Returns: `all` returns a list of all the enabled userbot commands. `category` returns the commands from the mentioned category. `command (dev)` returns the help for the mentioned command, here `dev` is optional and when mentioned it returns developer info for that command

Disable

Disable any bot command with it's name. Restarting will enable all the disabled commands.

Usage: `.disable sed`

Returns: This disables the use of sed by removing it's handler.

Disabled

Lists all the disabled bot commands.

Usage: `.disabled`

Returns: A list of all the disabled bot commands.

Enable

Enable any disabled bot commands with it's name.

Usage: `.enable sed`

Returns: This enables the sed command if it was disabled.

Enabled

Lists all the enabled commands.

Usage: `.enabled`

Returns: This returns a list of all enabled

Restart

Restarts the Telethon client. This reloads all the modules (smart plugins) with it.

Usage: `.restart`

Set Prefix

Change the default prefix for all the commands.

Usage: `.setprefix !`

Returns: The new prefix and how to reset to old prefix.

Reset Prefix

Resets to the default prefix which is ".".

Usage: `resetprefix`

Note: "resetprefix" works without any prefix because it is a fail-safe incase the user forgets the prefix.

Shutdown

Stops the Telethon client and exits the main script completely.

Usage: `.shutdown`

3.4.2 Bot Commands

AFK

Set your status as afk.

Usage: `.afk (reason)`

Returns: If anyone mentions/tags you, the userbot will notify them that you are afk for whatever reason if mentioned.

Approve

Approve a user for them to PM you.

Usage: `.approve @username/reply to a message`

Approved

Returns a list of all approved users **Usage:** `.approved`

Ban

Bans the user from a channel or chat with reason if mentioned.

Usage: `.ban @username/user-id/reply to a message (reason)`

Example: *.ban @shxnpie too goldy to handle*

This will ban @shxnpie with the reason "too godly to handle"

Bio

View or change your bio.

Usage: `.bio (text)`

Returns: If nothing is mentioned the bot will show your current bio and if some text is mentioned it will changed your bio to the same.

Bird

Send a pic of a random bird.

Usage: `.bird`

Blacklist

Add an item to the userbot's blacklist.

Usage: `.(g)bl <value1>..<valuen> or <option>:<value>`

Here "g" stands for global. bl is chat specific while gbl is global

Options/Values: user-id, Bio strings, text strings, domain/url

Example: *.gbl 1007684893 863314639* or *.bl id:863314639 url:https://www.google.com str:kan bad*

Returns: This will (g)ban the user if they match with the blacklisted items.

Blacklisted

Shows a list of all blacklisted users

Usage: `.blacklisted`

Blacklists

Sends a list of all blacklisted items

Usage: `.blacklist`

Block

Block a User.

Usage: `.block @username/user-id`

Cat

Send a random image of a cat.

Usage: `.cat`

Covid

Send info about the Covid-19 Pandemic.

Usage: `.covid (country)`

Returns: If a country is mentioned it will give its stats or World's stats will be shown.

Delete

Deletes the tagged message.

Usage: `.del` in reply to a message.

Delete Me

Deletes your message.

Usage: `.delme [amount=n] [skip=m]` if the number of message is not mentioned it will delete the message above it.

Delete Profile Picture

Deletes your profile picture a.k.a pfp

Usage: `.delpfp (n)` If numberof pfp is not mentioned it will delete the current pfp.

Delete Sticker

Deletes the tagged sticker from your sticker pack

Usage: `.delsticker` in reply to a sticker in your pack.

Demote

Demotes an admin to a user.

Usage: `.demote @username/user-id`

Download

Download a file from TG to the local storage

Usage: `.dl` in reply to a file/sticker

Eval

Evaluates the provided code.

Usage: `.eval 60+9` or `.eval reply`

Returns: *69* or the *Message* object of the replied message.

Exec

Executes the provided Python code.

Usage: `.exec print("TG-UserBot")`

Returns: *TG-UserBot*.

Get sticker

Convert a sticker to a png format.

Usage: `.getsticker` or `.getsticker file` or `.getsticker document`

Returns: Get replied to sticker as an image or as a file if mentioned.

User/Chat/Channel ID

Shows the user/chat/channel's id.

Usage: `.id (@username)` if nothing is mentioned it will give the chat's id.

Kang

Kang a sticker and add it to your pack.

Usage: `.kang` if the command is not in reply to a sticker, the bot will kang the nearest available sticker.

Kick

Kick a user form a chat/channel

Usage: `.kick @username (reason)` reason is optional

Kill

Kill a sub-process

Usage: `.kill` in reply to .eval or .exec sub-processes

Mention

Mention a user without the @

Usage: `@Username[text]`

Returns: This will tag the user within the text.

Mute

Mute a user.

Usage: `.mute @username (reason)` reason is optional

Name

Show/Change your name

Usage: `.name (text)`

Returns: Shows your current name and changes it if a textis specified.

Nearest DC

Get your country, current DC and nearest DC information of account.

Usage: `.nearestdc`

Returns: Country, your current DC and nearest DC.

Ping

Message edit/send response time.

Usage: `.ping`

Returns: The time it took to edit the message.

Ping DC

Gets the average response time of a datacenter (DC).

Usage: `.pingdc` or `.pingdc n` *n refers to the DC (1 - 5)*

Returns: Average response time of your DC or the one you specified.

Profile Picture

Show/Change profile picture.

Usage: `.pfp` in reply to an image, if not replied it will show your current pfp.

Promote

Promote a user to an admin.

Usage: `.promote @username (title="some text")`

Returns: This will promote the user. Title is optional.

Purge

Purge or delete messages.

Usage: `.purge (n) or reply to a message.`

Returns: This will purge the *n* number of messages or if replied to a messaged it will purge that message and all the messages below it.

Regex Ninja

Automatically deletes sed commands for regexbot.

Usage: `regexninja on` or `regexninja off` or `regexninja`

- on or off are used to set the mode. Without it, it'll return the current value.

Returns: New or current mode for Regex Ninja.

Remind Here

Send you a reminder in the current chat

Usage: `.remindhere w,d,h,m,s reply to a text or reason`

Available time units: *w, d, h, m, s*.

Remind me

Set a reminder for yourself.

Usage: `.remindme w,d,h,m,s reply to a text or reason`

Available time units: *w, d, h, m, s*.

Repository

Send a URL of the repository.

Usage: `.repo`

Resolve

Resolve a username/user-id/channel invites

Usage: `.resolve username/user-id/channel invites`

Returns: This returns with the info of the specified item.

Reverse

Do a reverse image search on google.

Usage: `.reverse` in reply to an image

Returns: With similar looking images and a possible related search term.

Remove Background

Removes the background from an image. (Requires Remove.bg API Key)

Usage: `.rmbg` in reply to an image

Remove blacklist

Remove a blacklisted id/string/url.

Usage: `rmblacklist id/string/url`

Remove Whitelist

Removes the user form whitelist.

Usage: `rmwhitelist id`

SED

Perform a regular expression substituion with the provided replacement.

Usage: `s/hi/hello` or `2s/cat/dog; s|boi|boy` or `s\crack\dope\g; 6s/cow/horse/i` *Format: ns/regexp/replacement/flags;*

- *n* refers to a line.
- The line and flags are optional.
- Use your delimeter or a semicolon to end each substituion for multiple replacement.

Returns: The replaced text if it there was successful match. If there was no replied to messages, then the last 10 messages will be used as source and the one which has a match will be used for replacement.

Shiba Inu

Sends a random image of Shiba Inus

Usage: `.shibe`

Speedtest

Perform a Speedtest.

Usage: `.speedtest`

Sticker Pack(s)

Show a list of all of your sticker packs

Usage: `.stickerpack`

Temporary Ban

Temp ban an user.

Usage: `.tban`

Term

Executes terminal commands.

Usage: `.term ls`

Returns: The result of *ls* command.

Terminate

Terminate or kill a sub-process **Usage:** `.terminate` in reply to .eval or .exec sub-processes.

Temporary Mute

Temp mute an user **Usage:** `.tmute`

Un/Dis Approve

Un Approve someone from PM-ing you.

Usage: `.unapprove @username/reply to thier message`

Unban

Unban someone from a chat or channel.

Usage: `.unban @username/reply to thier message`

Un Blacklist

Remove an user from blacklist.

Usage: `.unblacklist user-id`

Un Mute

Un Mute an user.

Usage: `.unmute @username/reply to thier message`

Update

Update the userbot

Usage: `.update`

Returns: This will pull latest changes from the repo and update the bot.

Heroku users need Heroku API ID for update to work.

Upload

Upload local files to Telegram.

Usage: `.upload path to file`

Username

View/Change ypur username.

Usage: `.username (text)` if text is not mentioned it will show your username.

Whitelist

Whitelist an user.

Usage: `.whitelist user-id`

Whitelists

Show a list of all whitelisted users.

Usage: `.whitelists`

Who is

Give all info about an user or a chat.

Usage: `.whois @username/reply to a message/this`

.whois this returns info about the chat.

YouTube-DL

Download videos from supported sites in your choice of format.

Usage: `.yt_dl https://youtu.be/dWhyFfsb74g listformats` or `.yt_dl https://youtu.be/dWhyFfsb74g bestaudio+bestvideo` *Format: .yt_dl url format*

Have a look at YouTube-DL's format selection for more information on formats and merging.

Returns: All the available formats or downloads the specified video's best audio and video, then merges them together.

3.5 Available Modules

All the available modules/smart plugins in the UserBot are listed below

Contents

3.5.1 Evaluators and Exececutors

`userbot.plugins.evaluators.`**`evaluate`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Evaluate Python expressions in the running script.

{prefix}eval (expression) Example: *{prefix}eval 1+1*

`userbot.plugins.evaluators.`**`execute`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Execute Python statements in a subprocess.

{prefix}exec (statement) Example: *{prefix}exec import os; os.system('clear')*

`userbot.plugins.evaluators.`**`killandterminate`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Kill or terminate a running subprocess by replying to the message.

{prefix}kill or *{prefix}terminate*

`userbot.plugins.evaluators.`**`terminal`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Execute terminal commands in a subprocess.

{prefix}term (command) Example: *{prefix}term echo 123*

3.5.2 Misc modules

`userbot.plugins.misc.`**`bot_mention`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Mention a user in the bot like link with a custom name.

Hi @kandnub[kandboob]

`userbot.plugins.misc.`**`deldog`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Paste the content to DelDog.

{prefix}paste in reply to a document or **{prefix}paste (text)**

`userbot.plugins.misc.`**`git_repo`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Get the repo url.

{prefix}repo

`userbot.plugins.misc.`**`removebg_post`**(*API_KEY: str*, *media: bytes*)

`userbot.plugins.misc.`**`resolver`**(*event: telethon.events.newmessage.NewMessage.Event*) → None

Resolve an invite link or a username.

{prefix}resolve (invite link)

userbot.plugins.misc.**rmbg**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Remove the background from an image or sticker.

{prefix}rmbg or **{prefix}rmbg (url)**

3.5.3 Reminder

userbot.plugins.reminder.**remindme**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Set a reminder (scheduled message) to be sent in n amount of time.

{prefix}remindme (time) (text) or {prefix}remindhere (time) (text) Example: {prefix}remindme 2h gts

3.5.4 SED

userbot.plugins.sed.**ninja**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Deletes our sed messages if regexninja is enabled

userbot.plugins.sed.**regex_ninja**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Enable and disable ninja mode for @regexbot

{prefix}regexninja or *{prefix}regexninja on* or *{prefix}regexninja off*

userbot.plugins.sed.**sed_substitute**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Perfom a GNU like SED substitution of the matched text.

{prefix}[line]s[ed]/(expression)/(substitution)/[flags][;] Everything inside the brackets is optional. You can perform case conversions in the substitution as well. The semi-colon is mandatory to perform multiple subs in one go.

3.5.5 Stickers

userbot.plugins.stickers.**delsticker**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Remove a sticker from your existing pack.

{prefix}delsticker in reply to your sticker

userbot.plugins.stickers.**getsticker**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Convert a sticker to a png and also send it as a file if specified.

{prefix}getsticker or **{prefix}getsticker (file|document)**

userbot.plugins.stickers.**kang**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Steal (AKA kang) stickers and images to your Sticker packs.

***{prefix}kang [pack] [emojis]* or {prefix}kang (short)=(title) [emojis]** *pack* and *emojis* can be used as arguments as well.

userbot.plugins.stickers.**stickerpack**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Get your default kang's sticker packs or update them.

***{prefix}stickerpack* or {prefix}stickerpack [args] Arguments:** *basic* or *animated* and *reset* **Examples:**

{prefix}stickerpack basic=1337pack {prefix}stickerpack animated=auto {prefix}stickerpack reset

3.5.6 WWW (Datacenter, etc.)

userbot.plugins.www.**nearestdc**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Get information of your country and data center information.

{prefix} nearestdc

userbot.plugins.www.**pingdc**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Ping your or other data center's IP addresses.

***{prefix}pingdc* or {prefix}pingdc (1|2|3|4|5)** This might not work if your connection blocks the requests

userbot.plugins.www.**speedtest**(*event: telethon.events.newmessage.NewMessage.Event*) → None
Perform a speedtest with the best available server based on ping.

{prefix}speedtest of **{prefix}speedtest (bits|bytes)**

3.5.7 YoutubeDL

userbot.plugins.yt_dl.**fix_attributes**(*path*, *info_dict: dict*, *round_message: bool = False*, *supports_streaming: bool = False*) → list
Avoid multiple instances of an attribute.

userbot.plugins.yt_dl.**yt_dl**(*event*)
Download videos from YouTube with their url in multiple formats.

{prefix}ytdl link1 link2 link3 [kwargs] Stream and progress are set to True, while update is at 10% by default.
Arguments:

format (The format to convert/download the video in), *delete* (Whether to delete the local files or not), *upload* (Whether to upload the downloaded files or not), *update* (The percentage to update the progress at), *stream* (Whether to upload the files as streamable or not), *progress* (Whether to update the progress by edits or not)

3.6 Helper Functions

These are some methods which could be used for more than just one function. This reduces the bloat and makes the overall code less bloated.

Contents

3.6.1 ID's

userbot.helper_funcs.ids.**get_entity_from_msg**(*event: telethon.events.newmessage.NewMessage.Event*) → Tuple[Union[None, telethon.tl.types.User], Union[None, bool, str], Union[None, bool, str]]

Get a User entity and/or a reason from the event's regex pattern

userbot.helper_funcs.ids.**get_user_from_msg**(*event: telethon.events.newmessage.NewMessage.Event*) → Union[int, str, None]

Get a user's ID or username from the event's regex pattern match

3.6.2 Parser

class userbot.helper_funcs.parser.**Parser**

Bases: object

Parse UserFull, ChannelFull and ChatFull objects.

static parse_full_chat(*chat_obj: Union[telethon.tl.types.ChatFull, telethon.tl.types.ChannelFull], event: telethon.events.newmessage.NewMessage.Event*) → str

Human-friendly string of a Chat/Channel obj's attributes

static parse_full_user(*usr_obj: telethon.tl.types.UserFull, event: telethon.events.newmessage.NewMessage.Event*) → str

Human-friendly string of an User obj's attributes

3.6.3 SED

exception userbot.helper_funcs.sed.**UnknownFlagError**(*flag*)

Bases: Exception

Used to raise an Exception for an unknown flag.

userbot.helper_funcs.sed.**convertCharacterCase**(*string: str*, *case: str*) → str

Convert the case of a character if found. Used for u and l.

Parameters

- **string** (str) – The string containing the case.
- **case** (str) – The raw string of the case.

Returns str | None – The replaced string on success, None otherwise.

userbot.helper_funcs.sed.**convertStringCase**(*string: str*, *case: str*) → str

Convert the matched string in the necessary case. Used for U and L.

Parameters

- **string** (str) – The string containing the case.
- **case** (str) – The case to search for.

Returns str | None – The replaced string on success, None otherwise.

userbot.helper_funcs.sed.**convertWordCase**(*string: str*, *case: str*) → str

Convert the matched words in the necessary case. Used for F and I.

Parameters

- **string** (str) – The string containing the case.
- **case** (str) – The case to search for.

Returns str | None – The replaced string on success, None otherwise.

userbot.helper_funcs.sed.**match_splitter**(*match: re.Match*) → Tuple[str, str, str, str]
Splits an re.Match to get the required attributes for substitution. Unescapes the slashes as well because this is Python.

Parameters **match** (Match) – Match object to split.

Returns (str, str, str, str) – A tuple of strings containing line, regexp, replacement and flags respectively.

userbot.helper_funcs.sed.**resolve_flags**(*fl: str*) → Tuple[int, Union[int, enum.Enum]]
Split all flags from the string for substituion.

Parameters **fl** (str) – String containing all the flags.

Raises *UnknownFlagError* – If there's an unknown flag, then this is raised to stop any farther execution.

Returns (int, (int | enum.Enum)) – Count and all the other re flags as an Enum type, if any.

userbot.helper_funcs.sed.**sub_matches**(*matches: list*, *original: str*) → Optional[str]
Iterate over all the matches whilst substituing the string.

Parameters

- **matches** (list) – A list of Match objects.
- **original** (str) – The original string to use for substituion.

Returns str | None – The final substitued string on success, None otherwise.

userbot.helper_funcs.sed.**substitute**(*fr: str*, *to: str*, *original: str*, *line: (<class 'str'>, <class 'int'>, None) = None*, *count: int = 1*, *flags: Union[enum.Enum, int] = 0*) → Optional[str]
Substitute a (specific) string. Match the regular-expression against the content of the pattern space. If found, replace matched string with replacement.

Parameters

- **fr** (str) – The regexp string.
- **to** (str) – The replacement string.
- **original** (str) – Original string to use for substituion.
- **line** (str | int | None, optional) – Line to use for substitution. Defaults to None.
- **count** (int, optional) – The amount of repetitions to do. Defaults to 1.
- **flags** (int | enum.Enum, optional) – Flags to use. Defaults to 0.

Returns str | None – The replaced string on success, None otherwise.

3.6.4 Time

userbot.helper_funcs.time.**amount_to_secs**(*amount: tuple*) → int
Resolves one unit to total seconds.

Parameters **amount** (int, str) – Tuple where str is the unit.

Returns int – Total seconds of the unit on success.

Example

```
>>> await amount_to_secs(("1", "m"))
60
```

userbot.helper_funcs.time.**split_extra_string**(*string: str*) → Tuple[Optional[str], Optional[int]]

userbot.helper_funcs.time.**string_to_secs**(*string: str*) → int

Converts a time string to total seconds.

Parameters **string** (str) – String conatining the time.

Returns int – Total seconds of all the units.

Example

```
>>> await string_to_sec("6h20m")
22800
```

3.6.5 YoutubeDL

class userbot.helper_funcs.yt_dl.**ProgressHook**(*event*, *update=5*)

Bases: object

Custom hook with the event stored for YTDL.

callback(*task*)

Cancel pending tasks else skip them if completed.

edit(**args*, ***kwargs*)

Create a Task of the progress edit.

hook(*d: dict*) → None

YoutubeDL's hook which logs progress and errors to UserBot logger.

class userbot.helper_funcs.yt_dl.**YTdlLogger**

Bases: object

Logger used for YoutubeDL which logs to UserBot logger.

critical(*msg: str*) → None

Logs critical messages with youtube-dl tag to UserBot logger.

debug(*msg: str*) → None

Logs debug messages with youtube-dl tag to UserBot logger.

error(*msg: str*) → None

Logs error messages with youtube-dl tag to UserBot logger.

warning(*msg: str*) → None

Logs warning messages with youtube-dl tag to UserBot logger.

userbot.helper_funcs.yt_dl.**extract_info**(*loop*, *executor: concurrent.futures._base.Executor*, *ydl_opts: dict*, *url: str*, *download: bool = False*) → str

Runs YoutubeDL's extract_info method without blocking the event loop.

Parameters

- **executor** (concurrent.futures.Executor) – Either ThreadPoolExecutor or ProcessPoolExecutor.
- **params** (dict) – Parameters/Keyword arguments to use for YoutubeDL.
- **url** (str) – The url which you want to use for extracting info.
- **download** (bool, optional) – If you want to download the video. Defaults to False.

Returns str – Successfull string or info_dict on success or an exception's string if any occur.

userbot.helper_funcs.yt_dl.**list_formats**(*info_dict: dict*) → str
YoutubeDL's list_formats method but without format notes.

Parameters **info_dict** (dict) – Dictionary which is returned by YoutubeDL's extract_info method.

Returns str – All available formats in order as a string instead of stdout.

3.7 Utilities

All the utilities used to manage plugins, client and events can be found here.

Contents

3.7.1 Client

class userbot.utils.client.**Command**(*func: <built-in function callable>*, *handlers: list*, *info: str*, *usage: str*, *builtin: bool*)
Bases: object

class userbot.utils.client.**UserBotClient**(*session: typing.Union[str, Session], api_id: int, api_hash: str, *, connection: typing.Type[Connection] = <class 'telethon.network.connection.tcpfull.ConnectionTcpFull'>, use_ipv6: bool = False, proxy: Union[tuple, dict] = None, local_addr=None, timeout: int = 10, request_retries: int = 5, connection_retries: int = 5, retry_delay: int = 1, auto_reconnect: bool = True, sequential_updates: bool = False, flood_sleep_threshold: int = 60, raise_last_call_error: bool = False, device_model: str = None, system_version: str = None, app_version: str = None, lang_code: str = 'en', system_lang_code: str = 'en', loop: asyncio.events.AbstractEventLoop = None, base_logger: Union[str, logging.Logger] = None*)

Bases: telethon.client.telegramclient.TelegramClient

UserBot client with additional attributes inheriting TelegramClient

answer(*entity, message: str = '', *, reply_to: Union[int, telethon.tl.patched.Message] = None, parse_mode: Optional[str] = 'markdown', link_preview: bool = False, file: Union[str, bytes, BinaryIO, telethon.tl.types.MessageMediaEmpty, telethon.tl.types.MessageMediaPhoto, telethon.tl.types.MessageMediaGeo, telethon.tl.types.MessageMediaContact, telethon.tl.types.MessageMediaUnsupported, telethon.tl.types.MessageMediaDocument, telethon.tl.types.MessageMediaWebPage, telethon.tl.types.MessageMediaVenue, telethon.tl.types.MessageMediaGame, telethon.tl.types.MessageMediaInvoice, telethon.tl.types.MessageMediaGeoLive, telethon.tl.types.MessageMediaPoll, telethon.tl.types.MessageMediaDice, telethon.tl.types.InputFile, telethon.tl.types.InputFileBig, telethon.tl.types.InputFileLocation, telethon.tl.types.InputEncryptedFileLocation, telethon.tl.types.InputDocumentFileLocation, telethon.tl.types.InputSecureFileLocation, telethon.tl.types.InputTakeoutFileLocation, telethon.tl.types.InputPhotoFileLocation, telethon.tl.types.InputPhotoLegacyFileLocation, telethon.tl.types.InputPeerPhotoFileLocation, telethon.tl.types.InputStickerSetThumb, Sequence[Union[str, bytes, BinaryIO, telethon.tl.types.MessageMediaEmpty, telethon.tl.types.MessageMediaPhoto, telethon.tl.types.MessageMediaGeo, telethon.tl.types.MessageMediaContact, telethon.tl.types.MessageMediaUnsupported, telethon.tl.types.MessageMediaDocument, telethon.tl.types.MessageMediaWebPage, telethon.tl.types.MessageMediaVenue, telethon.tl.types.MessageMediaGame, telethon.tl.types.MessageMediaInvoice, telethon.tl.types.MessageMediaGeoLive, telethon.tl.types.MessageMediaPoll, telethon.tl.types.MessageMediaDice, telethon.tl.types.InputFile, telethon.tl.types.InputFileBig, telethon.tl.types.InputFileLocation, telethon.tl.types.InputEncryptedFileLocation, telethon.tl.types.InputDocumentFileLocation, telethon.tl.types.InputSecureFileLocation, telethon.tl.types.InputTakeoutFileLocation, telethon.tl.types.InputPhotoFileLocation, telethon.tl.types.InputPhotoLegacyFileLocation, telethon.tl.types.InputPeerPhotoFileLocation, telethon.tl.types.InputStickerSetThumb]]] = None, force_document: bool = False, clear_draft: bool = False, buttons: Union[telethon.tl.types.ReplyKeyboardHide, telethon.tl.types.ReplyKeyboardForceReply, telethon.tl.types.ReplyKeyboardMarkup, telethon.tl.types.ReplyInlineMarkup, telethon.tl.types.KeyboardButton, telethon.tl.types.KeyboardButtonUrl, telethon.tl.types.KeyboardButtonCallback, telethon.tl.types.KeyboardButtonRequestPhone, telethon.tl.types.KeyboardButtonRequestGeoLocation, telethon.tl.types.KeyboardButtonSwitchInline, telethon.tl.types.KeyboardButtonGame, telethon.tl.types.KeyboardButtonBuy, telethon.tl.types.KeyboardButtonUrlAuth, telethon.tl.types.InputKeyboardButtonUrlAuth, telethon.tl.types.KeyboardButtonRequestPoll, telethon.tl.custom.button.Button, Sequence[Union[telethon.tl.types.KeyboardButton, telethon.tl.types.KeyboardButtonUrl, telethon.tl.types.KeyboardButtonCallback, telethon.tl.types.KeyboardButtonRequestPhone, telethon.tl.types.KeyboardButtonRequestGeoLocation, telethon.tl.types.KeyboardButtonSwitchInline, telethon.tl.types.KeyboardButtonGame, telethon.tl.types.KeyboardButtonBuy, telethon.tl.types.KeyboardButtonUrlAuth, telethon.tl.types.InputKeyboardButtonUrlAuth, telethon.tl.types.KeyboardButtonRequestPoll, telethon.tl.custom.button.Button]], Sequence[Sequence[Union[telethon.tl.types.KeyboardButton, telethon.tl.types.KeyboardButtonUrl, telethon.tl.types.KeyboardButtonCallback, telethon.tl.types.KeyboardButtonRequestPhone, telethon.tl.types.KeyboardButtonRequestGeoLocation, telethon.tl.types.KeyboardButtonSwitchInline, telethon.tl.types.KeyboardButtonGame, telethon.tl.types.KeyboardButtonBuy, telethon.tl.types.KeyboardButtonUrlAuth, telethon.tl.types.InputKeyboardButtonUrlAuth, telethon.tl.types.KeyboardButtonRequestPoll, telethon.tl.custom.button.Button]]]] = None, silent: bool = True, schedule: Union[float, datetime.datetime, datetime.date, datetime.timedelta, None] = None, log: str = None, reply: bool = False, self_destruct: int = None, event: telethon.tl.custom.message.Message = None*) → Union[None, telethon.tl.custom.message.Message, Sequence[telethon.tl.custom.message.Message]]

Custom bound method for the Message object

commandcategories = {}

commands = {}

config = None

database = True

disabled_commands = {}

failed_imports = []

fast_download_file(*location: Union[telethon.tl.types.Document, telethon.tl.types.InputDocumentFileLocation, telethon.tl.types.InputPeerPhotoFileLocation, telethon.tl.types.InputFileLocation, telethon.tl.types.InputPhotoFileLocation], out: BinaryIO, progress_callback: callable = None*) → BinaryIO

fast_upload_file(*file: BinaryIO, progress_callback: callable = None*) → Union[telethon.tl.types.InputFile, telethon.tl.types.InputFileBig]

get_traceback(*exc: Exception*) → str

logger = False

onMessage(*builtin: bool = False, command: str = None, edited: bool = True, info: str = None, doc_args: dict = {}, **kwargs*) → callable

Method to register a function without the client

static parse_arguments(*arguments: str*) → Tuple[List[Union[int, str, float, list]], Dict[str, Union[int, str, float, list, range, List[Union[int, str, float, list]]]]]

pluginManager = None

plugins = []

prefix = None

reconnect = True

register_commands = False

resanswer(*entity, message: str, plugin: str = None, name: str = None, formats: dict = {}, **kwargs*) → telethon.tl.custom.message.Message

running_processes = {}

version = 0

userbot.utils.client.**parse_arguments**

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C: @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

3.7.2 Events

class userbot.utils.events.**MessageEdited**(*disable_prefix: bool = None*, *regex: Tuple[str, int] = None*, *require_admin: bool = None*, *inline: bool = False*, ***kwargs*)

Bases: *userbot.utils.events.NewMessage*

Custom MessageEdited event inheriting the custom NewMessage event

class Event(*message*)

Bases: telethon.events.newmessage.Event

Overriding the default Event which inherits Telethon's NewMessage

classmethod build(*update*, *others=None*, *self_id=None*)

Required to check if message is edited, double events. Note: Don't handle UpdateEditChannelMessage from channels since the

update doesn't show which user edited the message

class userbot.utils.events.**NewMessage**(*disable_prefix: bool = None*, *regex: Tuple[str, int] = None*, *require_admin: bool = None*, *inline: bool = False*, ***kwargs*)

Bases: telethon.events.newmessage.NewMessage

Custom NewMessage event inheriting the default Telethon event

filter(*event*)

Overriding the default filter to check additional values

userbot.utils.events.**answer**(*self*, **args*, ***kwargs*)

userbot.utils.events.**resanswer**(*self*, **args*, ***kwargs*)

3.7.3 Helpers

class userbot.utils.helpers.**ProgressCallback**(*event*, *start=None*, *filen='unamed'*, *update=5*)

Bases: object

Custom class to handle upload and download progress.

dl_progress(*current*, *total*)

Handle the download progress only.

resolve_prog(*current*, *total*)

Calculate the necessary info and make a dict from it.

up_progress(*current*, *total*)

Handle the upload progress only.

userbot.utils.helpers.**calc_eta**(*elp: float*, *speed: int*, *current: int*, *total: int*) → int

userbot.utils.helpers.**disable_commands**(*client: userbot.utils.client.UserBotClient*, *commands: str*) → None

userbot.utils.helpers.**dl_prog**(*d: dict*, *cb: userbot.utils.helpers.ProgressCallback*) → Tuple[Union[str, bool], bool]

Logs the download progress

userbot.utils.helpers.**format_speed**(*speed_per_second*, *unit*)

userbot.utils.helpers.**get_chat_link**(*arg: Union[telethon.tl.types.User, telethon.tl.types.Chat, telethon.tl.types.Channel, telethon.events.newmessage.NewMessage.Event], reply=None*) → str

userbot.utils.helpers.**isRestart**(*client: userbot.utils.client.UserBotClient*) → None
Check if the script restarted itself and edit the last message

userbot.utils.helpers.**is_ffmpeg_there**()

userbot.utils.helpers.**printUser**(*entity: telethon.tl.types.User*) → None
Print the user's first name + last name upon start

userbot.utils.helpers.**printVersion**(*version: int, prefix: str*) → None
Print the version of the bot with the default prefix

userbot.utils.helpers.**restart**(*event: telethon.events.newmessage.NewMessage.Event*) → None

userbot.utils.helpers.**restarter**(*client: userbot.utils.client.UserBotClient*) → None

userbot.utils.helpers.**ul_prog**(*d: dict, cb: userbot.utils.helpers.ProgressCallback*) → Tuple[Union[str, bool], bool]
Logs the upload progress

3.7.4 Plugin Manager

class userbot.utils.pluginManager.**Callback**(*name: str, callback: <built-in function callable>*)
Bases: object

class userbot.utils.pluginManager.**Plugin**(*name: str, callbacks: List[userbot.utils.pluginManager.Callback], path: str, module: module*)
Bases: object

class userbot.utils.pluginManager.**PluginManager**(*client: telethon.client.telegramclient.TelegramClient*)
Bases: object

active_plugins = []

add_handlers() → None
Apply event handlers to all the found callbacks.

import_all() → None
Import all the (enabled) plugins and skip the rest.

inactive_plugins = []

remove_handlers() → None
Remove event handlers to all the found callbacks.

class userbot.utils.pluginManager.**SourcelessPluginLoader**(*name, data, path: str = '<string>'*)
Bases: importlib.abc.SourceLoader

Loader for (byte) strings which don't have a source.

get_code(*path*)
Return the code object if it exists.

get_data(*path*)
The data isn't stored locally, return the (bytes) string.

get_filename (*fullname*)
Return the origin (GitHub's raw URL).

userbot.utils.pluginManager.**get_pip_packages** (*requirements: str = None*) → list
Get a list of all the pacakage's names.

userbot.utils.pluginManager.**install_pip_packages** (*packages: List[str]*) → bool
Install pip packages.

userbot.utils.pluginManager.**restart_script** () → None
Restart the current script.

userbot.utils.pluginManager.**run_async** (*func: callable*)
Run async functions with the right event loop.

3.8 Creating Commands

To create a command you need to make an async function with a Pyrogram decorator in an individual or an already existing file in the `TG-UserBot/userbot/plugins/` directory. Pyrogram will load the this manually if you use Pyrogram's decorators, else they won't be reloaded on a restart.

Contents

3.8.1 Examples

The events module already has Filters and on_message decorator imported in it so you can import them from their instead of pyrogram. The commands decorator is used to store a command with it's function's handler so it can be used to disable/enable it later via the main commands.

To create function which replies to all your texts with the text you sent. For instance, you say "xyz", it will reply to your text with "You said *xyz*!"

```
from userbot import client

@client.onMessage(from_users="me")
async def echo(event):
    text = "You said __{}__!".format(event.text)
    await event.reply(text)
```

To create function which edits your "hi" text to a "hello".

```
from userbot import client

@client.onMessage(from_users="me" regex="(?i)^hi$")
async def hello(event):
    await event.edit("hello")
```

Python Module Index

u

Index

S

T

U

V

W

Y