Poloniexlendingbot Documentation

Release 1

Mikadily, Rnevet, Evanito and Contributors

Aug 21, 2017

Contents

Poloniex Lending Bot is an open-source program for automated lending on Poloniex and Bitfinex cryptocurrency exchange.

Contents:

CHAPTER 1

Installation

Installing on a Computer

Installing the bot on a computer is drag-and-drop and platform independent.

Prerequisites

You will need:

- Python 2.7.x (Must be added to PATH)

Recommended for easier use:

- git
- pip (to install following required Python modules)
- Numpy (if using Analysis module)
- requests (HTTPS communication)
- pytz (Timezone calculations)

It is possible to install all required Python modules **after downloading** of the bot running:

```
pip install -r requirements.txt
```

or, if you need to run it as root under Linux:

```
sudo pip install -r requirements.txt
```

Downloading

To download the bot you can either:

- (Recommended) Run `git clone https://github.com/BitBotFactory/poloniexlendingbot` if you have git installed. Using this method will allow you to do `git pull` at any time to grab updates.
- Download the source .zip file from the GitHub repo page or from this link. Extract it into an empty folder you won't accidentally delete.

(Optional) Automatically Run on Startup

- Windows using Startup Folder:

  Add a shortcut to `lendingbot.py` to the startup folder of the start menu. Its location may change with OS version, but for Windows 8/10 is `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp`

- Linux using systemd:

  Create the file `/lib/systemd/system/lendingbot.service` which contains the following text:

```
[Unit]
Description=LendingBot service
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/python <INSTALLATION DIRECTORY>/lendingbot.py
WorkingDirectory=<INSTALLATION DIRECTORY>
RestartSec=10
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

  Credit to GitHub user utdrmac.

  Modify the ExecStart and WorkingDirectory to match your setup.

  Enable the service using `sudo systemctl enable lendingbot.service`

- OSx:

  Help needed! If you have a solution for OSx and would like to share, you can either share it directly with us or make a PR with the edits.

Configuring

You have to configure the bot, especially choosing the exchange and api key/secret to use.

To configure the bot with your settings:

1. Copy `default.cfg.example` to `default.cfg` (Running lendingbot.py also does this for you if default.cfg doesn't already exist.)
2. Open `default.cfg` and enter your desired settings (information on settings here).
3. Save `default.cfg`

You are now ready to run the bot.

Running

To run, either:

- Double-click lendingbot.py (if you have .py associated with the Python executable)
- Run `python lendingbot.py` in command prompt or terminal.

Note: You can use arguments to specify a specific config file `-cfg` or to do dry runs `-dry`. To see these args do: `python lendingbot.py -h`

Installing on Pythonanywhere.com

Pythonanywhere.com is a useful website that will host and run Python code for you. This is perfect for our bot.

Prerequisites

You will need:

- A pythonanywhere.com account (Free version works fine)

Downloading the bot's files to Pythonanywhere

1. Start a new `bash` console from the "Consoles" tab.
2. Get the source code from git GitHub by running `git clone https://github.com/Mikadily/poloniexlendingbot`.
3. You should see some output with counters increasing.
4. Change directory to the source code `cd poloniexlendingbot`
5. You should now see `~/poloniexlendingbot (master)$` this means you are looking at the master branch and things are ok to continue.
6. Run the command `python2.7 lendingbot.py` once to generate the default.cfg
7. Modify the default.cfg with your settings (See Configuration.) You can do this with a tool called nano.
8. Run `nano default.cfg`, then use the arrow keys and backspace key to change `YourAPIKey` and `YourSecret`. Make sure the layout of the file stays the same as it was. They should both be on separate lines.
9. Press `Ctr+x` to exit, then press `y` to save the file, then press enter to accept the file name as `default.cfg`.
10. Now you can start up the bot. Run `python2.7 lendingbot.py`
11. If it's working you will see `Welcome to Poloniex Lending Bot` displayed in the console.
12. To update the bot just enter its directory, `cd poloniexlendingbot` and type, `git pull`. This will not change the `default.cfg` file.

Note: If you are running out of CPU time every day: It is recommended to use a high sleeptimeinactive time for this website, as they meter your CPU usage.

Creating the Web App (Optional)

1. If you would like to use the Webserver to view your bot's status, navigate to the "Web" tab.
2. Add a new web app.
3. Set the working directory to `/home/<username>/poloniexlendingbot/www/`
4. Set the static files to URL: `/static/` Directory: `/home/<username>/poloniexlendingbot/www`
5. Reload your website with the button at the top of the page.
6. You will be able to access the webapp at `http://<username>.pythonanywhere.com/static/lendingbot.html` once it finishes setting up.
7. To have the webserver communicate with your bot, you need to edit your settings (`default.cfg`) and uncomment (remove the `#` in front of) the following settings: `jsonfile` and `jsonlogsize`. Make sure that `startWebServer` REMAINS commented.

Warning: Do not use the built-in Simple Web Server on any host you do not control.

Running the Bot

To run the bot continuously (Recommended for free accounts):

1. Navigate to the "Consoles" tab.
2. Add a new "Custom console," name it "Poloniexlendingbot" and set the path to `python /home/<username>/poloniexlendingbot/lendingbot.py`
3. Click this link whenever you want to start the bot, it will run continuously until the website goes down for maintenance or the bot experiences an unexpected error.

To have the bot restart itself every 24 hours, you need to have a premium pythonanywhere account. This will make the bot more or less invincible to crashes and resets, but is not necessary.

1. Navigate to the "Schedule" tab.
2. Create a new task to run daily (time does not matter) set the path to: `python /home/<username>/poloniexlendingbot/lendingbot.py`
3. The bot will start once the time comes (UTC) and run indefinitely.

Note: If you are a free user, it will allow you to make the scheduled restart, but then it will only run for one hour and stop for 23.

Note: Free users are also limited to the number of output currencies they can use as blockchain.info is blocked from their servers. You can always use the pairs listed on poloniex, BTC, USDT. But will not have access to currencies such as EUR, GBP.

Using Docker

There is a `docker-compose.yaml` file in the root of the source that can be used to start the bot via docker. Compose is a tool for defining and running docker applications using a single file to configure the application's services.

To use this file:-

1. Install and setup docker for your platform, available on linux, mac and windows.
2. If you are using linux or windows server, you'll need to install docker-compose separately, see here.
3. If you don't already have a `default.cfg` created, then copy the example one and change the values as required using the instructions in this document.
4. You can now start the service with `docker-compose up -d`. It may take a minute or two on the first run as it has to download the required image and then some packages for that image when it starts.
5. If all went well you should see something like `Starting bitbotfactory_bot_1`.
6. When you see that message it just means that the container was started successfully, we still need to check the application is running as expected. In the yaml file the web service in the container is mapped to localhost. So you can open your web browser at this point and see if you can connect to the serivce. It should be runnning on http://127.0.0.1:8000/lendingbot.html.
7. If you don't see anything when connecting to that you can check the logs of the container with `docker-compose logs`. You should get some useful information from there. You may need to change some config vaules.
8. When you change the config values you need to restart the container, this can be done with `docker-compose stop` and then after changing configs, `docker-compose up -d`. You should notice it's significantly quicker than the first run now.
9. The last command to note is `docker-compose ps` this will give infomation on all running instances and the ports that are mapped. This can be useful if you plan on running multiple bots, or you just want to know if it's running.

CHAPTER 2

Configuration

Configuring the bot can be as simple as select the exchange to use and copy-pasting your API key and Secret.

New features are required to be backwards compatible with previous versions of the .cfg but it is still recommended that you update your config immediately after updating to take advantage of new features.

To begin, copy `default.cfg.example` to `default.cfg`. Now you can edit your settings.

Exchange selection, API key and Secret

Select the exchange to use in attribute `exchange`. Supported are `Poloniex` and `Bitfinex`. Default is Poloniex.

`exchange = Poloniex` or `exchange = Bitfinex`

Create a **NEW** API key and Secret from Poloniex or Bitfinex and paste them into the respective slots in the config.

```
apikey = XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX
secret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx...
```

Warning: Do not share your API key nor secret with anyone whom you do not trust with all your Poloniex funds.

Note: If you use an API key that has been used by any other application, it will likely fail for one application or the other. This is because the API requires a nonce.

Poloniex

Your Poloniex API key is all capital letters and numbers in groups of 8, separated by hyphens. Your secret is 128 lowercase letters and numbers.

HIGHLY Recommended:

- Disable the "Enable Trading" checkbox. The bot does not need it to operate normally.

- Enable IP filter to only the IP address the bot will be running from.

Bitfinex

Your Bitfinex API key and secret are both 43 letters and numbers.

HIGHLY Recommended:

- The lending bot needs only READ permission to "Account History", "Marging Funding", "Wallets" and WRITE permission to "Margin Funding" and "Wallets". Deselect all other on key generation, especially to "Withdraw".

Exchange Sections

There is a section for each exchange to configure exchange specific attributes.

- `all_currencies` List of all supported currencies for funding. The list have to change only when the exchange adds a new supported currency or removes one.

Timing

- `sleeptimeactive` is how long the bot will "rest" (in seconds) between running while the bot has loan offers waiting to be filled.
  - Default value: 60 seconds
  - Allowed range: 1 to 3600 seconds
  - If the bot finishes a cycle and has no open lend orders left to manage, it will change to inactive mode.

Note: Just because 1 second is a permitted sleeptime does not mean it is a good idea.

- `sleeptimeinactive` is how long the bot will "rest" (in seconds) between running while the bot has nothing to do.
  - Default value: 300 seconds (5 minutes)
  - Allowed range: 1 to 3600 seconds
  - If the bot finishes a cycle and has lend orders to manage, it will change to active mode.
- `timeout` is how long the bot waits for a response of a request
  - Default value: 30 seconds
  - Allowed range: 1 to 180 seconds

Min and Max Rates

- `mindailyrate` is the minimum rate (in percent) that the bot will allow offer loans at.
  - Default value: 0.005 percent
  - Allowed range: 0.0031 to 5 percent
  - 0.0031% every day for a year, works out around 1%. This is less than most bank accounts and is considered not worth while.

  - The current default value is a optimistic but very viable for the more high volume currencies. Not viable for lending DOGE, for example.

- `maxdailyrate` is the maximum rate (in percent) that the bot will allow lends to open.
  - Default value: 5 percent
  - Allowed range: 0.0031 to 5 percent
  - 2% is the default value offered by the exchange, but there is little reason not to set it higher if you feel optimistic.

Spreading your Lends

If `spreadlend = 1` and `gapbottom = 0`, it will behave as simple lending bot lending at lowest possible offer.

- `spreadlend` is the amount (as an integer) of separate loans the bot will split your balance into across the order book.
  - Default value: 3
  - Allowed range: 1 to 20 (1 is the same as disabling)
  - The loans are distributed evenly between gapbottom and gaptop.
  - This allows the bot to benefit from spikes in lending rate but can result in loan fragmentation (not really a bad thing since the bot has to deal with it.)
- `gapMode` is the "mode" you would like your gaps to be calculated in.
  - Default value: Relative
  - Allowed values: Relative, RawBTC, Raw
  - The values are case insensitive.
  - The purpose of spreading your lends is to skip dust offers in the lendbook, and also to take advantage of any spikes that occur.
  - **Mode descriptions:**
    - **`Relative` - `Gapbottom` and `Gaptop` will be relative to your balance for each coin individually.**
      - This is relative to your total lending balance, both loaned and unloaned.
      - `gapbottom` and `gaptop` will be in percents of your balance. (A setting of 100 will equal 100%)
      - Example: You have 1BTC. If `gapbottom = 100` then you will skip 100% of your balance of dust offers, thus skipping 1BTC into the lendbook. If `gaptop = 200` then you will continue into the lendbook until you reach 200% of your balance, thus 2BTC. Then, if `spreadlend = 5`, you will make 5 equal volume loans over that gap.
    - **`RawBTC` - `Gapbottom` and `Gaptop` will be in a raw BTC value, converted to each coin.**
      - Recommended when using one-size-fits-all settings.
      - `gapbottom` and `gaptop` will be in BTC. (A setting of 3 will equal 3 BTC)
      - Example: If `gapbottom = 1` and you are currently lending ETH, the bot will check the current exchange rate, say 1BTC = 10ETH. Then the bot will skip 10ETH of dust offers at the bottom of the lendbook before lending. If `gaptop = 10`, then using the same exchange

rate 10BTC will be 100ETH. The bot will then continue 100ETH into the loanbook before stopping. Then, if `spreadlend = 5`, you will make 5 equal volume loans over that gap.

* **`Raw - Gapbottom` and `Gaptop` will be in a raw value of the coin being lent.**
  - Recommended when used with coin-specific settings.
  - `gapbottom` and `gaptop` will be in value of the coin. (A setting of 3 will equal 3 BTC, 3 ETH, 3 DOGE, or whatever coin is being lent.)
  - Example: If `gapbottom = 1` and you are currently lending ETH, the bot will skip 1ETH of dust offers at the bottom of the lendbook before lending. If `gaptop = 10`, the bot will then continue 10ETH into the loanbook before stopping. Then, if `spreadlend = 5`, you will make 5 equal volume loans over that gap.

- `gapbottom` is the lower setting for your `gapMode` values, and will be where you start to lend.
  - Default value: 10 percent
  - Allowed range: 0 to <arbitrary large number>
  - 10% gapbottom is recommended to skip past dust at the bottom of the lending book, but if you have a VERY high volume this will cause issues as you stray to far away from the most competitive bid.
- `gaptop` is the upper setting for your `gapMode` values, and will be where you finish spreading your lends.
  - Default value: 200 percent
  - Allowed range: 0 to <arbitrary large number>
  - This value should be adjusted based on your coin volume to avoid going astronomically far away from a realistic rate.

Variable loan Length

These values allow you to lock in a better rate for a longer period of time, as per your configuration.

- `xdaythreshold` is the rate (in percent) where the bot will begin attempting to lend for a longer period of time.
  - Default value: 0.2 percent
  - Allowed range: 0 to 5 percent
- `xdays` is the length(in days) of any loan whose rate exceeds the set xdaythreshold.
  - Default value: 60 days
  - Allowed range: 2 to 60 days

Auto-transfer from Exchange Balance

If you regularly transfer funds into your Poloniex account but don't enjoy having to log in yourself and transfer them to the lending balance, this feature is for you.

- `transferableCurrencies` is a list of currencies you would like to be transferred.
  - Default value: Commented out
  - Format: `CURRENCY_TICKER,STR,BTC,BTS,CLAM,DOGE,DASH,LTC,MAID,XMR,XRP,ETH,FCT,ALL,ACTIVE`

  - Commenting it out will disable the feature.
  - Entering `ACTIVE` within the list will transfer any currencies that are found in your lending account, as well as any other currencies alongside it. Example: `ACTIVE, BTC, CLAM` will do BTC, CLAM, and any coins you are already lending.
  - Entering `ALL` will simply transfer all coins available to lending.
  - Do not worry about duplicates when using `ACTIVE`, they are handled.
  - Coins will be transferred every time the bot runs (60 seconds by default) so if you intend to trade or withdrawal it is recommended to turn off the bot or disable this feature.

Unimportant settings

Very few situations require you to change these settings.

- `minloansize` is the minimum size that a bot will make a loan at.
  - Default value: 0.01 of a coin
  - Allowed range: 0.01 and up.
  - If you dislike loan fragmentation, then this will make the minimum for each loan larger.
  - Automatically adjusts to at least meet the minimum of each coin.
- `KeepStuckOrders` If True, keeps orders that are "stuck" in the market instead of canceling them.
  - Default value: True
  - Allowed values: True or False
  - A "Stuck" order occurs when it partially fills and leaves the coins balance total (total = open orders + let in balance) below your `minloansize` and so the bot would not be able to lend it again if it was canceled.
  - When disabled, stuck orders will be canceled and held in balance until enough orders expire to allow it to lend again.
- `hideCoins` If True, will not lend any of a coin if its market low is below the set `mindailyrate`.
  - Default value: True
  - Allowed values: True or False. Commented defaults to True
  - This hides your coins from appearing in walls.
  - Allows you to catch a higher rate if it spikes past your `mindailyrate`.
  - Not necessarily recommended if used with `analyseCurrencies` with an aggressive `lendingStyle`, as the bot may miss short-lived rate spikes.
  - If you are using the `analyseCurrencies` option, you will likely see a lot of `Not lending BTC due to rate below 0.9631%` type messages in the logs. This is normal.
- `endDate` Bot will try to make sure all your loans are done by this date so you can withdraw or do whatever you need.
  - Default value: Disabled
  - Uncomment to enable.
  - Format: `YEAR,MONTH,DAY`

Max to be lent

This feature group allows you to only lend a certain percentage of your total holding in a coin, until the lending rate suprasses a certain threshhold. Then it will lend at max capacity.

- `maxtolend` is a raw number of how much you will lend of each coin whose lending rate is below `maxtolendrate`.
  - Default value: Disabled
  - Allowed range: 0 (disabled) or `minloansize` and up
  - If set to 0, same as if commented.
  - If disabled, will check if `maxpercenttolend` is enabled and use that if it is enabled.
  - Setting this overwrites `maxpercenttolend`
  - This is a global setting for the raw value of coin that will be lent if the coins lending value is under `maxtolendrate`
  - Has no effect if current rate is higher than `maxtolendrate`
  - If the remainder (after subtracting `maxtolend`) in a coin's balance is less than `minloansize`, then the remainder will be lent anyway. Otherwise, the coins would go to waste since you can't lend under `minloansize`
- `maxpercenttolend` is a percentage of how much you will lend of each coin whose lending rate is below `maxtolendrate`
  - Default value: Disabled
  - Allowed range: 0 (disabled) to 100 percent
  - If set to 0, same as if commented.
  - If disabled in addition to `maxtolend`, entire feature will be disabled.
  - This percentage is calculated per-coin, and is the percentage of the balance that will be lent if the coin's current rate is less than `maxtolendrate`
  - Has no effect if current rate is higher than `maxtolendrate`
  - If the remainder (after subtracting `maxpercenttolend`'s value) in a coin's balance is less than `minloansize`, then the remainder will be lent anyway. Otherwise, the coins would go to waste since you can't lend under `minloansize`
- `maxtolendrate` is the rate threshold when all coins are lent.
  - Default value: Disabled
  - Allowed range: 0 (disabled) or `mindailyrate` to 5 percent
  - Setting this to 0 with a limit in place causes the limit to always be active.
  - When an indiviaual coin's lending rate passes this threshold, all of the coin will be lent instead of the limits `maxtolend` or `maxpercenttolend`

Market Analysis

This feature allows you to record a currency's market and have the bot see trends. With this data, we can compute a recommended minimum lending rate per currency to avoid lending at times when the rate dips.

- `analyseCurrencies` is the list of currencies to analyse.
  - Format: `CURRENCY_TICKER,STR,BTC,BTS,CLAM,DOGE,DASH,LTC,MAID,XMR,XRP,ETH,FCT,ALL,ACTIVE`
  - Commenting it out will disable the entire feature.
  - Entering `ACTIVE` within the list will analyse any currencies that are found in your lending account, as well as any other currencies alongside it. Example: `ACTIVE, BTC, CLAM` will do BTC, CLAM, and any coins you are already lending.
  - Entering `ALL` will simply analyse all coins on the lending market, whether or not you are using them.
  - Do not worry about duplicates when using `ACTIVE`, they are handled.
- `analyseMaxAge` is the maximum duration to store market data.
  - Default value: 30 days
  - Allowed range: 1-365 days
- `analyseUpdateInterval` is how often (asynchronous to the bot) to record each market's data.
  - Default value: 60 seconds
  - Allowed range: 10-3600 seconds

Note: Storage usage caused by the above two settings can be calculated by: `<amountOfCurrencies> * 30 * analyseMaxAge * (86,400 / analyseUpdateInterval)` bytes. Default settings with `ALL` currencies enabled will result in using `15.552 MegaBytes` maximum.

- `lendingStyle` lets you choose the percentile of each currency's market to lend at.
  - Default value: 75
  - Allowed range: 1-99
  - Recommendations: Conservative = 50, Moderate = 75, Aggressive = 90, Very Aggressive = 99
  - This is a percentile, so choosing 75 would mean that your minimum will be the value that the market is above 25% of the recorded time.
  - This will stop the bot from lending during a large dip in rate, but will still allow you to take advantage of any spikes in rate.

Config per Coin

This can be configured in one of two ways.

Coincfg dictionary

- `coincfg` is in the form of a dictionary and allows for advanced, per-coin options.
  - Default value: Commented out, uncomment to enable.
  - Format: `["COINTICKER:MINLENDRATE:ENABLED?:MAXTOLEND:MAXPERCENTTOLEND:MAXTOLENDRATE", "CLAM:0.6:1:0:.75:.1",...]`
  - COINTICKER refers to the ticker of the coin, ex. BTC, CLAM, MAID, DOGE.

- MINLENDRATE is that coins minimum lending rate, overrides the global setting. Follows the limits of `minlendrate`
- ENABLED? refers to a value of `0` if the coin is disabled and will no longer lend. Any positive integer will enable lending for the coin.
- MAXTOLEND, MAXPERCENTTOLEND, and MAXTOLENDRATE refer to their respective settings above, but are unique to the specified coin specifically.
- There can be as many different coins as you want in coincfg, but each coin may only appear once.

Separate coin sections

This is an alternative layout for the coin config mentioned above. It provides the ability to change the minloansize per coin, but is otherwise identical in functionality. To use this configuration, make sure to comment out the line where coincfg is defined, then add a section for each coin you wish to configure.

> **Warning:** These sections should come at the end of the file, after the other options for the bot.

Configuration should look like this:

```
[BTC]
minloansize = 0.01
mindailyrate = 0.1
maxactiveamount = 1
maxtolend = 0
maxpercenttolend = 0
maxtolendrate = 0
gapmode = raw
gapbottom = 10
gaptop = 20
```

Advanced logging and Web Display

- `jsonfile` is the location where the bot will log to a .json file instead of into console.
  - Default value: Commented out, uncomment to enable.
  - Format: `www/botlog.json`
  - This is the location relative to the running instance of the bot where it will store the .json file. The default location or a path inside the `customWebServerTemplate` folder is recommended if using the webserver functionality.
- `jsonlogsize` is the amount of lines the botlog will keep before deleting the oldest event.
  - Default value: Commented out, uncomment to enable.
  - Format: `200`
  - Reasons to lower this include: you are conscious of bandwidth when hosting your webserver, you prefer (slightly) faster loading times and less RAM usage of bot.
- `startWebServer` if true, this enables a webserver on the www/ folder.
  - Default value: Commented out, uncomment to enable.
  - The server page can be accessed locally, at `http://localhost:8000/lendingbot.html` by default.

  - Forces `jsonfile` to be set using `www/botlog.json` (unless otherwise configured)
  - You must close bot with a keyboard interrupt (CTRL-C on Windows) to properly shutdown the server and release the socket, otherwise you may have to wait several minutes for it to release itself.
- `customWebServerAddress` is the IP address that the webserver can be found at.
  - Advanced users only.
  - Default value: 0.0.0.0 Uncomment to change
  - Format: `IP`
  - Setting the ip to `127.0.0.1` will ONLY allow the webpage to be accessed at localhost (`127.0.0.1`)
  - Setting the ip to `0.0.0.0` will allow the webpage to be accessed at localhost (`127.0.0.1`) as well as at the computer's LAN IP address within the local network. This option is the most versatile, and is default.
  - Setting the ip to `192.168.0.<LAN IP>` will ONLY allow the webpage to be access at the computer's LAN IP address within the local network (And not through localhost.) It is recommended to be sure the device has a static local IP.
  - You must know what you are doing when changing the IP address to anything other than the three suggested configurations above.
- `customWebServerPort` is the IP port that the webserver can be found at
  - Advanced users only.
  - Default value: 8000 Uncomment to change
  - Format: `PORT`
  - Do not set the port to a reserved port or you will receive an error when running the bot or attempting to connect (depending on HOW reserved a port is.)
  - When you like to run more than one bot on same host (e.g. the first to lend on Poloniex and another one to lend on Bitfinex) different port numbers have to defined. (e.g 8000 in Poloniex's config and 8001 in Bitfinex's config file)
- `customWebServerTemplate` is the location the bot will use for WebServer HTML GUI template.
  - Default value: www, uncomment to enable.
  - Format: `PATH`
  - This is the location relative to the running HTML GUI instance used by the bot. Be sure the `jsonfile` belongs to this folder.
- `outputCurrency` this is the ticker of the coin which you would like the website to report your summary earnings in.
  - Default value: BTC
  - Acceptable values: BTC, USDT, Any coin with a direct Poloniex BTC trading pair (ex. DOGE, MAID, ETH), Currencies that have a BTC exchange rate on blockchain.info (i.e. EUR, USD)
  - Will be a close estimate, due to unexpected market fluctuations, trade fees, and other unforseeable factors.
- `label` is a custom name of the bot, that will be displayed in html page.
  - Default value: Lending Bot
  - Allowed values: Any literal string

Plugins

Plugins allow extending Bot functionality with extra features. To enable/disable a plugin add/remove it to the `plugins` list config option under the [BOT] section, example:

```
plugins = Plugin1, Plugin2, etc...
```

AccountStats Plugin

The AccountStats plugin fetches all your loan history and provides statistics based on it. Current implementation sends a earnings summary Notification (see Notifications sections) every 24hr.

To enable the plugin add `AccountStats` to the `plugins` config options, example:

```
plugins = AccountStats
```

There is an optional setting to change how frequently this plugin reports. By default, once per day. Example:

```
[ACCOUNTSTATS]
ReportInterval = 1800
```

Be aware that first initialization might take longer as the bot will fetch all the history.

lendingbot.html options

You can pass options to statistics page by adding them to URL. Eg, `http://localhost:8000/lendingbot.html?option1=value&option2=0`

- `effrate` controls how effective loan rate is calculated. Yearly rates are calculated based on effective rate, so this option affects them as well. Last used mode remembered by browser, so you do not have to specify this option every time. By default, effective loan rate is calculated considering lent precentage (from total available coins) and poloniex 15% fee.
  - Allowed values: `lentperc`, `onlyfee`.
  - Default value: `lentperc`.
  - `onlyfee` calculates effective rate without considering lent coin percentage.
- `displayUnit` controls BTC's unit output.
  - Allowed values: `BTC`, `mBTC`, `Bits`, `Satoshi`
  - Default value: `BTC`
  - This setting will change all display of Bitcoin to that unit. Ex. 1 BTC -> 1000 mBTC.
- `earningsInOutputCurrency` define which earnings are shown in the output currency.
  - Allowed values: `all`, `summary`
  - Default value: `all`

Notifications

The bot supports sending notifications for serveral different events on several different platforms. To enable notifications, you must first have a section in your config called `[notifications]`, inside which you should enable at least one of the following events and also at least one notification platfom. The list of events you can notify about are:

Notification events

- `notify_new_loans`
  - Sends a notification each time a loan offer is filled.
- `notify_tx_coins`
  - This will send a notification if any coins are transferred from your exchange account, to your lending account. You must have `transferableCurrencies` enabled for this to work. Then you should set `notify_tx_coins = True`.
- `notify_xday_threshold`
  - This will send a notification every time a loan is created that is above your `xdaythreshold` config value. To enable you should set `notify_xday_threshold = True`.
- `notify_summary_minutes`
  - This will send a summary of the current loans you have every X minutes. This is similar to the information you get in the log line when running the bot, or the line a the top of the web page. To enable this add `notify_summary_minutes = 120`. This will send you a notification every 2 hours (120 minutes).
- `notify_caught_exception`
  - This is more useful for developers and people wanting to help out by raising issues on github. This will send a notification every time there is an exception thrown in the bot that we don't handle. To enable add `notify_caught_exception = True`.

Once you have decided which notifications you want to recive, you can then go about configuring platforms to send them on. Currently the bot supports:

Email notifications

This is probably the easiest to configure, though there can still be issues with gmail where you need to enable a few things. You can find out more about that here if you're having problems. If you don't wish to use gmail search google for the smtp settings of your email provider. To enable email you should configure the following:

```
email = True
email_login_address = me@gmail.com
email_login_password = secretPassword
email_smtp_server = smtp.gmail.com
email_smtp_port = 465
email_smtp_starttls = False
email_to_addresses = me@gmail.com,you@gmail.com
```

Slack notifications

Before you can post to slack you need to create an API token, to do this visit this page. Once you have a token you can then configure the bot as so:

```
slack = True
slack_token = xoxp-46351793751-46348393136-47931965411-a8757952e4
slack_channels = #cryptocurrency,@someUser
```

To post in a channel prefix with # and to post a dm to a user prefix with @. You can send to as many channels or users as you want.

Telegram notifications

Quickstart To have telegram notifications you need to get a bot id from the BotFather. You can do that here. Once you have a bot id you need to get your Chat ID or create a channel and invite the bot so it can chat there. Once you have all this in place you configure it like so:

```
telegram = True
telegram_bot_id = 281421543:AGGB1TqP7XqhxhT7VOty0Aml8DV_R6kimHw
telegram_chat_ids = 123456789,@cryptocurrency
```

Detailed Messages are sent to the telegram bot API using HTTPS requests. You can read more about it here.

Telegram Bots are special accounts that do not require an additional phone number to set up, they do however need a unique authentication token. This is the token we need to get and add to the lendingbot's default.cfg. They are normally in the format `123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11`.

When we say we are creating a new telegram bot, all it means is that we are creating an account for the lendingbot to send message through. To create a bot and get a token, we must request it from the BotFather. This is telegram's tool for creating new bots.

These are the steps to carry out:

1. Install the telegram desktop client from their site. Then set it up with your phone number and login.
2. Start a conversation with The BotFather. When you click the link it should open up in the telegram desktop client.
3. Once you have a conversation started type `/newbot`, you'll then be asked what to call the bot and it's username. The name of your bot is displayed in contact details and elsewhere. The Username is a short name, to be used in mentions and telegram.me links. When complete you'll receive a token.
4. You can check everything is working OK by going to https://api.telegram.org/bot*YOURTOKEN*/getme, for example my test one is https://api.telegram.org/bot288427377:AAGB1TqL7XqhxhT7VOxu8Ams8DV_J6kimHw/getme. If that's all working then move on to the next step.
5. Now we need somewhere to send the messages, if you want to send a message to yourself, you first need your Chat ID. The easiest way I've found to get this is to send the bot a message from your desktop client and then use the getupdates method. So search for the bot in the desktop client's search bar and start a conversation. Then in your browser go to https://api.telegram.org/bot*YOURTOKEN*/getupdates. You should see a few lines of text, the one we're interested in looks like `"chat":{"id":123456789,"first_name":"Michael","last_name":"Robinson","type":"private"}`. The number after ID is your chat ID.
6. Again, just to check everything is working, lets send ourselves a message. You can do this by putting this in your browser https://api.telegram.org/bot*YOURTOKEN*/sendmessage?text=TEST%20BOT&chat_id=*YOUR_CHAT_ID* You should see a message in your desktop client. If so you have the right ID and we can move on.
7. The last step to get it working is just adding the two values to your default.cfg file and turning on `telegram = True`. You should set `telegram_bot_id` to the token you got from the BotFather,

and set the `telegram_chat_ids` to a comma separated list of people you want to send messages to.

8. **(optional) If you'd like a specific channel for the bot to send messages you can follow these steps.**

   (a) Open the desktop client and create a new channel

   (b) Start a conversation with the BotFather and type /setjoingroups, then follow the questsions he asks.

   (c) Click on the message we sent earlier from the bot, then click on the bot's name in the conversation. You should see 'Add To Group'. Click this and add it to the new group you created.

   (d) Now you should be able to add the `@nameOfChannel` to your `default.cfg` file and post all the updates there too. Make sure the list is comma separated and you have the '@' infront of the channel name. This is only done for names, not Chat IDs.

Pushbullet notifications

To enable Pushbullet notifications, you first need to create an API key and then discover your device ID.

Visit your Account Settings and click 'Create Access Token'. Add this to the config file as shown below.

You then need to visit this documentation page and run the example curl command for listing your devices (be sure to substitute your API token as created in the previous step). Copy the value listed for 'iden' into the config file as shown below.:

```
pushbullet = True
pushbullet_token = l.2mDDvy4RRdzcQN9LEWSy22amS7u3LJZ1
pushbullet_deviceid = ujpah72o0sjAoRtnM0jb
```

IRC notifications

IRC is very easy to configure, if you are already interested in using it you'll understand what each of the options are.

The main thing to note is that you need to have the python module 'irc' installed. You can git it from pip like so:

```
pip install irc
```

Once you have that installed you have access to the following options for configuration:

```
irc = True
irc_host = irc.freenode.net
irc_port = 6667
irc_nick = LendingBot
irc_ident = ledningbot
irc_realname = Poloniex lending bot
irc_target = #bitbotfactory
```

If you want to send a message directly to a user rather than a channel, you can specify it in the irc_target without the preceeding '#'. There is currently only support for one channel or user, but we can add more if there's any interest for it.

CHAPTER 3

Contributing

How to format Python Code

If you want to make a successful pull request, here are some suggestions.

Recommended IDE: PyCharm

PEP8

Poloniex lending bot follows PEP8 styling guidelines to maximize code readability and maintenance.

To help out users and automate much of the process, the Codacy Continuous Integration bot will comment on pull requests to alert you to any changes you need to make. Codacy has many inspections it does, which may extend past PEP8 conventions. We recommend you follow its suggestions as much as possible.

To make following PEP8 as painless as possible, we strongly recommend using an Integrated Development Environment that features PEP8 suggestions, such as PyCharm.

Indent Style

You may have your own preference, it does not matter because *spaces and tabs do not mix.*

Poloniexlendingbot uses *spaces* to conform with PEP8 standards. Please use an IDE that can help you with this.

Commenting Code

Many coders learned to code without commenting their logic. That works if you are the only person working on the project, but quickly becomes a problem when it is your job to decipher what someone else was thinking when coding.

You will probably be relieved to read that code comments are not mandatory, because code comments are an apology.

Only comment your code if you need to leave a note. (We won't judge you for it.)

Variable or Option Naming

Whenever you create a variable or configuration option, follow PEP8 standards, they are as follows:

> Do not make global single-letter variable names, those do not help anybody. Using them within a function for a few lines in context is okay, but calling it much later requires it be given a proper name.
>
> Functions are named like `create_new_offer()` and variables are named similarly, like `amount_of_lends`.

Line Length

To make it simple to review code in a diff viewer (and several other reasons) line length is limited to 128 characters in Python code.

Python allows plenty of features for one line to be split into multiple lines, those are permitted.

Configuration Options

New configuration options should be placed near similar options (see categories on the configuration page) and require a short description above the actual setting.

If a setting is added that changes functionality, it is required that you add handling for having the option commented out.

How to use the Configuration module:

- If your change is in a new module, you need to init it to import the Config object. Create a function init(<args>) that set the args to global variables within the module. With this, pass Config from the main of the bot.
- Use `option = Config.get(CATEGORY, OPTION, DEFAULT_VALUE, LOWER_LIMIT, UPPER_LIMIT)` to get the option from the Config. Only do this in your init()
- CATEGORY: The category of the config file it goes under. Currently there is only 'API' and 'BOT'
- OPTION: Case-sensitive name of the option you are pulling from the bot.
- DEFAULT_VALUE: Default: False. This is the value that .get() will return if no value is set (option is commented). If set to "None": the bot will not allow it to be left blank ever. Optional.
- LOWER_LIMIT: Default: False. The lower float value that the option can be set to. If OPTION's value is lesser than this, the bot will alert them and exit. Optional. Only use for numerical options.
- UPPER_LIMIT: Default: False. The upper float value that the option can be set to. If OPTION's value is greater than this, the bot will alert them and exit. Optional. Only use for numerical options.

`Config.has_option(CATEGORY, OPTION)` will always return a boolean for whether the option exists or not. If the option is commented it will return False.

Making Documentation

It is important to keep proper documentation of configuration options, to make it as clear as possible for the user.

Building Docs

If you want to be able to build the html files of the documentation, you need to have Sphinx installed. You can install this with `pip install sphinx`. From there, run `make html` in the docs directory. These instructions can also be found in the included README.

Writing Docs

Just follow the lead of the rest of the docs.

- Configurations need a default, allowed values, effect, etc. in a format similar to the other options.
- Installation instructions should be similar to a followable list.

Javascript

Codacy will offer suggestions for fixes to standardize/fix the code. Do not worry about having too many commits in your PR.

Lendingbot.js is already quite messy, so following Codacy's suggestions is highly encouraged.