

---

# **IreneUtility**

***Release V1.04.0***

**MujyKun**

**Jun 29, 2021**



## CONTENTS:

<b>1</b>	<b>Utility Client</b>	<b>1</b>
<b>2</b>	<b>Base</b>	<b>3</b>
<b>3</b>	<b>Models</b>	<b>5</b>
3.1	__init__(models) . . . . .	5
3.2	Album . . . . .	5
3.3	BiasGame . . . . .	5
3.4	BlackJackGame . . . . .	6
3.5	File . . . . .	7
3.6	GachaValues . . . . .	7
3.7	Game . . . . .	7
3.8	Group . . . . .	7
3.9	GuessingGame . . . . .	7
3.10	Idol . . . . .	8
3.11	IdolCard . . . . .	8
3.12	Keys . . . . .	8
3.13	PlayingCard . . . . .	10
3.14	User . . . . .	10
<b>4</b>	<b>Util</b>	<b>13</b>
4.1	u_biasgame . . . . .	13
4.2	u_blackjack . . . . .	13
4.3	u_cache . . . . .	14
4.4	u_customcommands . . . . .	16
4.5	u_database . . . . .	16
4.6	u_datadog . . . . .	16
4.7	u_exceptions . . . . .	16
4.8	u_gacha . . . . .	17
4.9	u_groupmembers . . . . .	17
4.10	u_guessinggame . . . . .	20
4.11	u_lastfm . . . . .	20
4.12	u_local_cache . . . . .	21
4.13	u_logger . . . . .	21
4.14	u_miscellaneous . . . . .	22
4.15	u_moderator . . . . .	23
4.16	u_patreon . . . . .	24
4.17	u_reminder . . . . .	24
4.18	u_selfassignroles . . . . .	25
4.19	u_twitch . . . . .	26

4.20	u_twitter	26
4.21	u_weverse	27
<b>5</b>	<b>S_SQL</b>	<b>29</b>
5.1	__init__(s_sql)	29
5.2	db_structure	29
5.3	s_biasgame	29
5.4	s_blackjack	29
5.5	s_cache	30
5.6	s_general	30
5.7	s_groupmembers	31
5.8	s_guessinggame	32
5.9	s_lastfm	32
5.10	s_levels	32
5.11	s_logging	32
5.12	s_miscellaneous	33
5.13	s_moderator	33
5.14	s_patreon	33
5.15	s_reminder	33
5.16	s_selfassignroles	34
5.17	s_session	34
5.18	s_twitch	34
5.19	s_twitter	35
5.20	s_user	35
5.21	s_weverse	35
<b>6</b>	<b>Indices and tables</b>	<b>37</b>
	<b>Python Module Index</b>	<b>39</b>
	<b>Index</b>	<b>41</b>

---

CHAPTER  
**ONE**

---

**UTILITY CLIENT**



## BASE

```
class IreneUtility.Base.Base(utility_object)
```

Base Class that will hold the utility object for a subclass. Meant to be a parent class.





## MODELS

### 3.1 `__init__` (models)

**class** IreneUtility.models.**BaseUtil**

Serves as an extension to avoid Circular imports while maintaining the main Utility object across the models that need to utilize certain methods.

Note that we have this as a class so we do not need to constantly import if ex is None. This object makes it much easier to maintain.

**base\_util** - The primary instance of *BaseUtil* that contains the primary IreneUtility.Utility.Utility Instance.

### 3.2 Album

**class** IreneUtility.models.**Album**(*album\_name, idol\_cards, rap\_score=0, dance\_score=0, vocal\_score=0, popularity=0, income\_rate=0, album\_active\_time=None*)

**async** **calculate\_dance\_score()**

Returns the total dance skills of all idols in the album

**async** **calculate\_rap\_score()**

Returns the total rap skills of all idols in the album

**async** **calculate\_vocal\_score()**

returns the total vocal skills of all idols in the album

**async** **skill\_completion\_multiplier()**

Provides a small bonus for having idols that fulfill all of the different categories.

### 3.3 BiasGame

**class** IreneUtility.models.**BiasGame**(\*args, *bracket\_size=8, gender='all'*)

**async** **check\_message**(*message, first\_idol, second\_idol*)

Check the reactions of the message and process results

**async** **end\_game()**

End the game

**async generate\_brackets()**

Generates the brackets and the idols going against each other

**async process\_game()**

Process bias guessing game by sending messages and new questions until the game should end.

**async run\_current\_bracket()**

Generate a new question for the bias game.

## 3.4 BlackJackGame

**class** IreneUtility.models.BlackJackGame(\*args, first\_player, first\_player\_bet)

BlackJack Game for two users.

**async announce\_winner()**

Announce the winner of the game.

**async calculate\_score**(cards: List[IreneUtility.models.playingcard.PlayingCard]) → int

Calculate the score of a player.

**Parameters** cards – List of PlayingCards the user has in their deck.

**Returns** Score of the player

**async check\_message()**

Check incoming messages in the text channel and determines if the player wants to hit or stand.

**async check\_standing**(first\_player=True)

Check if a player is standing.

**Parameters** first\_player – True if it is the first player that wants to stand. Otherwise its the second player.

**Returns** True if the user is standing.

**async choose\_random\_card()** → IreneUtility.models.playingcard.PlayingCard

Chooses a random card that is available in the deck.

**async deal\_with\_bets()**

Properly deal with bets and appropriately remove/add the bets from the players balances.

**async determine\_winner()** → IreneUtility.models.user.User

Determine the winner of the blackjack game.

**async end\_game()**

End the blackjack game.

**async finalize\_game()**

Finalize the game by dealing with bets, announcing the winner, and officially ending the game.

**async hit**(first\_player=True)

Let a player hit

**Parameters** first\_player – True if it is the first player that wants to hit. Otherwise its the second player.

**async process\_game()**

Start the blackjack game.

**async stand**(first\_player=True)

Let a player stand

**Parameters** **first\_player** – True if it is the first player that wants to stand. Otherwise its the second player.

## 3.5 File

**class** IreneUtility.models.**File**(*file\_location, image\_url*)

Represents an OS file.

**async** **send\_file**(*channel: discord.channel.TextChannel, message=None, url=False*)

### Parameters

- **channel** – Discord Text Channel to send the file to.
- **message** – Message followed with the image.
- **url** – True/False if the url should be posted instead.

## 3.6 GachaValues

**class** IreneUtility.models.**GachaValues**

## 3.7 Game

**class** IreneUtility.models.**Game**(*utility\_obj, ctx*)

**async** **end\_game**()

Ends a guessing game.

**async** **process\_game**()

Starts processing the game.

## 3.8 Group

**class** IreneUtility.models.**Group**(*\*\*kwargs*)

A group of idols/celebrities

## 3.9 GuessingGame

**class** IreneUtility.models.**GuessingGame**(*\*args, max\_rounds=20, timeout=20, gender='all', difficulty='medium', game\_mode='idol'*)

**async** **check\_message**()

Check incoming messages in the text channel and determine if it is correct.

**async** **create\_acceptable\_answers**()

Create acceptable answers.

```
async create_idol_pool()  
    Create the game's idol pool.  
  
async create_new_question()  
    Create a new question and send it to the channel.  
  
async credit_user(user_id)  
    Increment a user's score  
  
async display_winners()  
    Displays the winners and their scores.  
  
async end_game()  
    Ends a guessing game.  
  
async print_answer(question_skipped=False, dead_link=False)  
    Prints the current round's answer.  
  
async process_game()  
    Ignores errors and continuously makes new questions until the game should end.  
  
async update_scores()  
    Updates all player scores
```

## 3.10 Idol

```
class IreneUtility.models.Idol(**kwargs)  
    Represents an Idol/Celebrity.
```

## 3.11 IdolCard

```
class IreneUtility.models.IdolCard(idol, card_owner: discord.user.User, issue_number=1, rap_skill=0,  
                                   vocal_skill=0, dance_skill=0, rarity='common')
```

## 3.12 Keys

```
class IreneUtility.models.Keys(**kwargs)  
  
    ACCESS_SECRET: str  
        Spotify  
  
    X_RapidAPI_headers: dict  
        Tenor  
  
    api_port: str  
        Bot Site  
  
    client: commands.AutoShardedBot  
        Reactions/Emojis Turned into Unicode Strings  
  
    client_session: AioHTTPClient  
        Wolfram
```

**async\_connect\_to\_db()**  
Create a pool to the postgres database using asyncpg

**datadog\_app\_key: str**  
BlackJack

**db\_conn: asyncpg.pool.Pool**  
Papago/Translator

**discord\_boats: discord\_boats\_client**  
Database Connection

**get\_kwarg(kwarg\_name)**  
Get a kwarg

**idol\_photo\_location: str**  
Twitch API

**kwargs**  
Bot Tokens

**last\_fm\_headers: dict**  
Patreon

**lyric\_client: lyrics\_client**  
//github.com/MujyKun/Weverse  
**Type** Weverse - https

**next\_emoji**  
Twitter

**oxford\_app\_key: str**  
Urban Dictionary

**patreon\_super\_role\_id: int**  
AioHTTP

**playing\_card\_location: str**  
Bot API

**spotify\_client\_secret: str**  
Oxford

**tenor\_key: str**  
Top.gg

**test\_client\_token: str**  
General

**top\_gg: DBLClient**  
Discord Boats

**translate\_private\_key**  
LastFM

**twitch\_client\_secret: str**  
//github.com/DataDog/datadogpy  
**Type** DataDog - https

**weverse\_image\_folder: str**  
GroupMembers Directories

```
wolfram_app_id: str
//github.com/KSoft-Si/ksoftapi.py

Type Lyrics API - https
```

### 3.13 PlayingCard

```
class IreneUtility.models.PlayingCard(*args, card_id, card_name, value)
    Represents a custom playing card.
```

### 3.14 User

```
class IreneUtility.models.User(user_id: int)
    Represents a discord user.

    async ensure_level()
        Ensure the user has a row in the levels table.

    async get_daily_amount()
        Get the amount the user should receive daily.

    async static get_needed_for_level(level: int, column_name: str)
        Returns money/experience needed for a certain level.

    async get_profile_xp()
        Get the user's profile xp.

    async get_rob_amount(money)
        The amount to rob a specific person based on their rob level.

        Parameters money – (The amount of money the person getting robbed has)

    async get_rob_percentage()
        Get the percentage of being able to rob. (Every 1 is 5%)

    async get_shortened_balance() → str
        Shorten an amount of money to its value places.

    async register_currency()
        Registers the user to the currency system.

    async set_language(language)
        Sets the user's language.

    async set_level(level, command_name)
        Set the level of a command/feature.

    async set_profile_xp(xp_amount)
        Set the user's profile xp.

    async try_to_rob_user(user) → bool
        Attempt to rob a user.

        Parameters user – User to rob

        Returns True if the user successfully robbed.
```

**async update\_balance**(*balance: Optional[int] = None, add: Optional[int] = None, remove: Optional[int] = None*)

Set balance of user in db and object.

**Parameters**

- **balance** – The amount to set the balance to.
- **add** – The amount to add to the current balance.
- **remove** – The amount to remove from the current balance.

**async update\_level\_in\_db**(*column\_name, level*)

Update the level for the user.





## 4.1 u\_biasgame

**class** IreneUtility.util.u\_biasgame.BiasGame(\*args)

**async create\_bias\_game\_image**(first\_idol\_id, second\_idol\_id)  
Uses thread pool to create bias game image to prevent IO blocking.

**merge\_images**(first\_idol\_id, second\_idol\_id)  
Merge Idol Images if the merge doesn't exist already.

## 4.2 u\_blackjack

**class** IreneUtility.util.u\_blackjack.BlackJack(\*args)

**async find\_game**(user) → *IreneUtility.models.blackjackgame.BlackJackGame*  
Find a blackjack game that a user is in.

**Parameters** **user** – A Utility User object, Context, or User ID

**Returns** BlackJack Game

**async generate\_playing\_cards**()  
Generate custom playing cards with the background as an idol avatar.

**merge\_images**(card\_file\_name, idol\_file\_name, unique\_id)  
Merges a template card with an idol avatar.

**Parameters**

- **card\_file\_name** – A Card's File name & type without the directory.
- **idol\_file\_name** – An Idol's File name & type without the directory.
- **unique\_id** – The unique row id in the database table that will be the merged file name.

**remove\_all\_card\_files**()  
Remove all card files from OS.

## 4.3 u\_cache

**class** IreneUtility.util.u\_cache.Cache(\*args)

**static** apply\_bold\_to\_braces(text: str) → str

Applies bold markdown in between braces.

**async** create\_bot\_bans()

Create the cache for banned users from the bot.

**async** create\_bot\_command\_cache()

Create custom command cache

**async** create\_cache(on\_boot\_up=True)

Create the general cache on startup

**async** create\_command\_counter()

Updates Cache for command counter and sessions

**async** create\_currency\_cache()

Create cache for currency

**async** create\_dead\_link\_cache()

Creates Dead Link Cache

**async** create\_disabled\_games\_cache()

Creates a list of channels with disabled games.

**async** create\_gg\_filter\_cache()

Create filtering of guessing game cache.

**async** create\_group\_cache()

Create Group Objects and store them as cache

**async** create\_groups()

Set cache for group photo count

**async** create\_guessing\_game\_cache()

Create cache for guessing game scores

**async** create\_guild\_cache()

Update the DB Guild Cache. Useful for updating info for API.

**async** create\_idol\_cache()

Create Idol Objects and store them as cache.

**async** create\_idols()

Set cache for idol photo count

**async** create\_image\_cache()

Creates Image objects and stores them in local cache.

Note that usage of these images is unnecessary as a call to the API would be more efficient. Therefore, IreneBot will not be using the image objects directly.

**async** create\_language\_cache()

Create cache for user languages.

**async** create\_levels\_cache()

Create the cache for user levels.

**async create\_logging\_channels()**  
Create the cache for logged servers and channels.

**async create\_mod\_mail()**  
Create the cache for existing mod mail

**async create\_n\_word\_counter()**  
Update NWord Cache

**async create\_patreons()**  
Create the cache for Patrons.

**async create\_playing\_cards()**  
Crache cache for playing cards.

**async create\_reminder\_cache()**  
Create cache for reminders

**async create\_restricted\_channel\_cache()**  
Create restricted idol channel cache

**async create\_self\_assignable\_role\_cache()**  
Create cache for self assignable roles

**async create\_send\_idol\_photo\_cache()**  
Creates the list of idols that needs to be sent to a text channel after t time.

**async create\_server\_prefixes()**  
Create the cache for server prefixes.

**async create\_temp\_channels()**  
Create the cache for temp channels.

**async create\_timezone\_cache()**  
Create cache for timezones

**async create\_twitch\_cache()**  
Create cache for twitch followings

**async create\_unscramble\_game\_cache()**  
Create cache for unscramble game scores

**async create\_user\_notifications()**  
Set the cache for user phrases

**async create\_welcome\_message\_cache()**  
Create the cache for welcome messages.

**async create\_weverse\_channel\_cache()**  
Create cache for channels that are following a community on weverse.

**async get\_session\_id()**  
Force get the session id, this will also set total used and the session id.

**async load\_language\_packs()**  
Create cache for language packs.

**async process\_cache\_time(*method, name, \*args, \*\*kwargs*)**  
Process the cache time.

**async process\_session()**  
Sets the new session id, total used, and time format for distinguishing days.

**async request\_support\_server\_members()**

Request the support server to be chunked and be put in cache.

We need this so that we can accurately determine if a user is in the support server or not.

**async request\_twitter\_channel()**

Fetch twitter channel and store it in cache.

## 4.4 u\_customcommands

**class IreneUtility.util.u\_customcommands.CustomCommands(\*args)**

## 4.5 u\_database

**class IreneUtility.util.u\_database.DataBase(\*args)**

**async get\_db\_connection()**

Retrieve Database Connection

## 4.6 u\_datadog

**class IreneUtility.util.u\_datadog.DataDog(\*args)**

**get\_metric\_info()**

Retrieves metric info to send to datadog.

**initialize\_data\_dog()**

Initialize The DataDog Class for metrics.

**send\_metric(metric\_name, value)**

Send a metric value to DataDog.

**send\_metrics()**

Send metrics for all stats.

## 4.7 u\_exceptions

**exception IreneUtility.util.u\_exceptions.ImproperFormat**

Invalid Format was given.

**exception IreneUtility.util.u\_exceptions.InvalidParamsPassed(msg)**

Raised when IDs are invalid for an add/remove/set method.

**exception IreneUtility.util.u\_exceptions.Limit**

A limit was reached.

**exception IreneUtility.util.u\_exceptions.MaxAttempts(msg)**

Essentially StopIteration, but created for logging to file & console upon raising error.

**exception** IreneUtility.util.u\_exceptions.NoKeyFound(*msg*)

Raised when no keys were found.

**exception** IreneUtility.util.u\_exceptions.NoTimeZone

No Timezone was found.

**exception** IreneUtility.util.u\_exceptions.Pass

A hack exception. This exception was meant to occur in order to jump to a part in code. Indicates something went right instead of wrong.

**exception** IreneUtility.util.u\_exceptions.ShouldNotBeHere(*msg*)

Raised when safe-guarded code is created. If this exception is raised, the code reached a point it should not have

**exception** IreneUtility.util.u\_exceptions.TooLarge

The input was too long.

## 4.8 u\_gacha

**class** IreneUtility.util.u\_gacha.Gacha(\**args*)

**async static** get\_all\_skill\_scores(*idol\_skill\_type*, *card\_rarity*)

Returns the rap, dance, and vocal scores of an idol

**async static** random\_album\_popularity()

Returns a truncated normal random popularity between 0 and 1 that follows the PDF  $f(y) = \exp(-a * (y - c)^2)$  where  $a$  is the curvature and  $c$  is the bell center. This is a truncated normal distribution. The random variable transformation  $g(x) : x \rightarrow y$  needs to be used where  $x$  is a uniform distribution and  $y$  is the  $f(x)$  distribution.  $g(x) = Fy^{-1}(F(x))$  where  $Fx = x$  and  $Fy = \text{erf}(\sqrt{a} * (y - c))$  which are the corresponding CDFs of  $x$  and  $y$ . Solving we find that  $g(x) = \text{erfinv}(x) / \sqrt{a} + c$ .

**async** random\_skill\_score(*card\_rarity*)

Return a random skill score for rap/dance/vocal for the gacha card between 1 and 99 dependent on the rarity of the card.

## 4.9 u\_groupmembers

**class** IreneUtility.util.u\_groupmembers.GroupMembers(\**args*)

**async** check\_channel\_sending\_photos(*channel\_id*)

Checks a text channel ID to see if it is restricted from having idol photos sent.

**async** check\_group\_and\_idol(*message\_content*, *server\_id=None*)

returns specific idols being called from a reference to a group ex: redvelvet irene

**async** check\_idol\_post\_reactions(*message*, *user\_msg*, *idol*, *link*, *guessing\_game=False*)

Check the reactions on an idol post or guessing game.

**async** check\_server\_sending\_photos(*server\_id*)

Checks a server to see if it has a specific channel to send idol photos to

**async static** check\_to\_add\_alias\_to\_list(*alias*, *name*, *mode=0*)

Check whether to add an alias to a list. Compares a name with an existing alias.

**async** choose\_random\_member(*members=None*, *groups=None*)

Choose a random member object from a member or group list given.

**async delete\_channel\_from\_send\_idol**(*text\_channel*)

Deletes a channel permanently from the send idol cache.

**Parameters** **text\_channel** – (discord.TextChannel or int) Key to pop from the cache. The client should know which it is.

**async delete\_restricted\_channel\_from\_cache**(*channel\_id, send\_all*)

Deletes restricted channel from cache.

**async static format\_card\_fields**(*obj, card\_formats*)

Formats all relevant card fields to be displayed

**async static get\_all\_groups**()

Get all groups.

**async get\_all\_images\_count**()

Get the amount of images the bot has.

**async get\_channel\_sending\_photos**(*server\_id*)

Returns a text channel from a server that requires idol photos to be sent to a specific text channel.

**async get\_db\_aliases**(*object\_id, group=False*)

Get the aliases of an idol or group from the database.

**async get\_db\_groups\_from\_member**(*member\_id*)

Return all the group ids an idol is in from the database.

**async get\_db\_idol\_called**(*member\_id*)

Get the amount of times an idol has been called from the database.

**async get\_db\_members\_in\_group**(*group\_id*)

Get the members in a specific group from the database.

**async get\_google\_drive\_link**(*api\_url*)

Get the google drive link based on the api's image url.

**async get\_group**(*group\_id*) → *IreneUtility.models.group.Group*

Get a group by the group id.

**async get\_group\_names\_as\_string**(*idol*)

Get the group names split by a | .

**async get\_group\_where\_group\_matches\_name**(*name, mode=0, server\_id=None*)

Get group ids for a specific name.

**async get\_idol\_by\_image\_id**(*image\_id*)

Get an idol object by the unique image id.

**Returns** Idol Object or NoneType

**async get\_idol\_post\_embed**(*group\_id, idol, photo\_link, user\_id=None, guild\_id=None, guessing\_game=False, scores=None*)

The embed for an idol post.

**async get\_idol\_where\_member\_matches\_name**(*name, mode=0, server\_id=None*)

Get idol object if the name matches an idol

**async get\_member**(*idol\_id*) → *IreneUtility.models.idol.Idol*

Get a member by the idol id.

**async get\_member\_names\_as\_string**(*group*)

Get the member names split by a | .

**async get\_random\_idol()**

Get a random idol with at least 1 photo.

**async idol\_post(channel, idol, \*\*kwargs)**

The main process for managing the errors behind an api call for an idol's photo.

#### Parameters

- **channel** – (discord.Channel) Channel that the embed/image should be posted to.
- **idol** – (IreneUtility Idol object) Idol that will be posted.
- **photo\_link** – Image that will be embedded.
- **group\_id** – Group ID the idol may be posted with. (used for if a group photo was called instead).
- **special\_message** – Any message that should be applied to the post.
- **user\_id** – User ID that is calling the photo.
- **guessing\_game** – (bool) Whether the method is called from a guessing game.
- **scores** – (dict) Any scores that may come with a game. In a format of {user id : score}
- **msg\_timeout** – Amount of time before deleting a message.

**log\_idol\_command(message)**

Log an idol photo that was called.

**async manage\_send\_idol\_photo(text\_channel, idol\_id, limit=None)**

Adds/Removes/Updates idol ids based on the text channel that will be used to send idol photos after t time.

#### Parameters

- **text\_channel** – discord.TextChannel or text channel id for the idol photo to be sent to
- **idol\_id** – idol id to add or remove.
- **limit** – (int) the limit for what the text channel can have. If this is exceeded, u\_exceptions.Limit will be raised.

**Returns** False if text channel input was incorrect. 'insert' if the idol id was inserted. 'remove' if the idol id was removed. 'delete' if the channel was completely removed from the table.

**async process\_names(ctx, page\_number\_or\_group, mode)**

Structures the input for idol names commands and sends information to transfer the names to the channels.

**async remove\_global\_alias(obj, alias)**

Remove a global idol/group alias

**async remove\_local\_alias(obj, alias, server\_id)**

Remove a server idol/group alias

**async request\_image\_post(message, idol, channel)**

Checks if the user can post an image, then posts it.

**async send\_names(ctx, mode, user\_page\_number=1, group\_ids=None)**

Send the names of all idols in an embed with many pages.

**async send\_vote\_message(message)**

Send the vote message to a user.

**async set\_as\_group\_photo(link)**

Set a photo as a group photo.

**async set\_embed\_card\_info**(*obj, group=False, server\_id=None*)

Sets General Information about a Group or Idol.

**async set\_embed\_with\_aliases**(*name, server\_id=None*)

Create an embed with the aliases of the names of groups or idols sent in

**async set\_embed\_with\_all\_aliases**(*mode, server\_id=None*)

Send the names of all aliases in an embed with many pages.

**async set\_global\_alias**(*obj, alias*)

Set an idol/group alias for the bot.

**async set\_local\_alias**(*obj, alias, server\_id*)

Set an idol/group alias for a server

**async update\_member\_count**(*idol*)

Update the amount of times an idol has been called.

## 4.10 u\_guessinggame

**class** IreneUtility.util.u\_guessinggame.**GuessingGame**(\*args)

**async create\_user\_in\_guessing\_game**(*user\_id*)

Inserts a user into the guessing game db with no scores. This allows for updating scores easier.

**async filter\_add\_group**(*user, group*)

Adds a filtered group to a user.

**async filter\_auto\_add\_remove\_group**(*user\_or\_id, group\_or\_id*)

Automatically Add/Remove a group from a user's filtered group list based on the current list.

:returns False if group was removed. :returns True if group was added. :exception self.ex.exceptions.InvalidParamsPassed if invalid group id.

**async filter\_remove\_group**(*user, group*)

Remove a filtered group from a user.

**async get\_guessing\_game\_top\_ten**(*difficulty, members=None*)

Get the top ten of a certain guessing game difficulty

**async toggle\_filter**(*user\_id*)

Enables/Disables the group filter for the guessing game on a user.

**async update\_user\_guessing\_game\_score**(*difficulty, user\_id, score*)

Update a user's guessing game score.

## 4.11 u\_lastfm

**class** IreneUtility.util.u\_lastfm.**LastFM**(\*args)

**create\_fm\_payload**(*method, user=None, limit=None, time\_period=None*)

Creates the payload to be sent to Last FM

**async get\_fm\_response**(*method, user=None, limit=None, time\_period=None*)

Receives the response from Last FM



**async get\_fm\_username**(*user\_id*)  
Gets Last FM username from the DB.

**async set\_fm\_username**(*user\_id*, *username*)  
Sets Last FM username to the DB.

## 4.12 u\_local\_cache

**class** IreneUtility.util.u\_local\_cache.**Cache**(\*args)

**commands\_used**  
{ server\_id : {  
    send\_all: 0 or 1, logging\_channel: channel\_id, channels: [channel\_id, channel\_id] },  
    ... }

**list\_of\_logged\_channels**  
Welcome Messages { server\_id : {  
    channel\_id: channel\_id, message: text, enabled: 0 or 1 }  
}

**server\_prefixes**  
reset timer for idol photos (keeps track of command usage) {  
    reset\_time: date userid: [commands\_used, time\_since\_last\_command]  
}

**twitch\_channels\_is\_live**  
User Notifications and Mod Mail are constantly iterated over, therefore we need a synced list apart from the information present in the user objects to stop the bot from being blocked/behind on future commands. These were removed and put into the user objects, but will now be placed back due to this issue.

**welcome\_messages**  
Temp Channels { channel\_id : seconds }

## 4.13 u\_logger

IreneUtility.util.u\_logger.**console**(*message*, *method=None*, *event\_loop=None*)  
Prints message to console and adds to logging.

### Parameters

- **message** – The message that will be printed out and logged.
- **method** – The function/method that called this function.
- **event\_loop** – An existing event loop.

IreneUtility.util.u\_logger.**get\_class**(*method*)  
Returns the class that belongs to the method. :param method: The method that needs to be checked.

REF -> <https://stackoverflow.com/a/25959545/13159093>

IreneUtility.util.u\_logger.**logfile**(*message*)  
Logs a message to the info file.

**Parameters** **message** – The message that will be logged.

`IreneUtility.util.u_logger.manage_log(body_msg, log_type, method=None, event_loop=None)`  
Process the type of logging it is and writes to file.

**Parameters**

- **body\_msg** – (str) Line that should be appended to the file.
- **log\_type** – (str) The end of the file name that differentiates the type of logging it is.
- **method** – The function/method that called this function.
- **event\_loop** – An existing event loop.

`IreneUtility.util.u_logger.useless(message, method=None)`

Logs Try-Except-Passes. This will put the exceptions into a log file specifically for cases with no exception needed.

**Parameters**

- **message** – The message that will be logged.
- **method** – The function/method that called this function.

**async** `IreneUtility.util.u_logger.write_to_file(location, body_msg)`  
Write a line to a file.

**Parameters**

- **location** – (str) File Location
- **body\_msg** – (str) Line that should be appended to the file.

## 4.14 u\_miscellaneous

**class** `IreneUtility.util.u_miscellaneous.Miscellaneous(*args)`

**async** `add_command_count(command_name)`

Add 1 to the specific command count and to the count of the current minute.

**async** `add_session_count()`

Adds one to the current session count for commands used and for the total used.

**async** `ban_user_from_bot(user_id)`

Bans a user from using the bot.

**async** `check_for_bot_mentions(message)`

Returns true if the message is only a bot mention and nothing else.

**async** `check_for_nword(message)`

Processes new messages that contains the N word.

**async** `check_if_bot_banned(user_id)`

Check if the user can use the bot.

**async** **static** `check_if_moderator(ctx)`

Check if a user is a moderator on a server

**async** `check_if_temp_channel(channel_id)`

Check if a channel is a temp channel

**async check\_message\_is\_command**(*message*, *is\_command\_name=False*)  
Check if a message is a command.

**static check\_message\_not\_empty**(*message*)  
Check if a message has content.

**check\_nword**(*message\_content*)  
Check if a message contains the NWord.

**async delete\_temp\_messages**(*message*)  
Delete messages that are temp channels

**async disable\_interaction**(*server\_id*, *interaction*)  
Disable an interaction (to a specific server)

**async enable\_interaction**(*server\_id*, *interaction*)  
Reenable an interaction that was disabled by a server

**get\_channel\_count**()  
Returns the channel count from all the guilds the bot is connected to.

**async static get\_cooldown\_time**(*time*)  
Turn command cooldown of seconds into hours, minutes, and seconds.

**async get\_disabled\_server\_interactions**(*server\_id*)  
Get a server's disabled interactions.

**static get\_int\_index**(*number*, *index*)  
Retrieves the specific index of an integer. Ex: Calling index 3 for integer 12345 will return 123.

**async get\_language\_code**(*input\_language*)  
Returns a language code that is compatible with the papago framework.

**get\_server\_count**()  
Returns the guild count the bot is connected to.

**get\_text\_channel\_count**()  
Returns the text channel count from all the guilds the bot is connected to.

**get\_user\_count**()  
Get the amount of users that the bot is watching over.

**get\_voice\_channel\_count**()  
Returns the voice channel count from all the guilds the bot is connected to.

**async send\_ban\_message**(*channel*)  
A message to send for a user that is banned from the bot.

**async unban\_user\_from\_bot**(*user\_id*)  
UnBans a user from the bot.

## 4.15 u\_moderator

```
class IreneUtility.util.u_moderator.Moderator(*args)
```

**async add\_welcome\_message\_server**(*channel\_id*, *guild\_id*, *message*, *enabled*)  
Adds a new welcome message server.

**async check\_welcome\_message\_enabled**(*server\_id*)  
Check if a welcome message server is enabled.

**async toggle\_games**(*channel\_id: int*) → bool  
Toggles game usage in a text channel.  
Will return True if channel has games enabled.

**async update\_welcome\_message\_channel**(*server\_id, channel\_id*)  
Update the welcome message channel.

**async update\_welcome\_message\_enabled**(*server\_id, enabled*)  
Update a welcome message server's enabled status

## 4.16 u\_patreon

**class** IreneUtility.util.u\_patreon.Patreon(\*args)

**async add\_to\_patreon**(*user\_id*)  
Add user as a permanent patron.

**async check\_if\_patreon**(*user\_id, super\_patron=False*)  
Check if the user is a patreon. There are two ways to check if a user is a patreon. The first way is getting the members in the Patreon/Super Patreon Role. The second way is a table to check for permanent patreon users that are directly added by the bot owner. – After modifying -> We take it straight from cache now.

**async get\_patreon\_role\_members**(*super\_patron=False*)  
Get the members in the patreon roles.

**async get\_patreon\_users**()  
Get the permanent patron users

**async remove\_from\_patreon**(*user\_id*)  
Remove user from being a permanent patron.

**async reset\_patreon\_cooldown**(*ctx*)  
Checks if the user is a patreon and resets their cooldown.

## 4.17 u\_reminder

**class** IreneUtility.util.u\_reminder.Reminder(\*args)

**async static determine\_time\_type**(*user\_input*)  
Determine if time is relative time or absolute time relative time: remind me to \_\_\_\_\_ in 6 days absolute time: remind me to \_\_\_\_\_ at 6PM

**async static format\_time**(*string\_format, user\_timezone, input\_time: Optional[datetime.datetime] = None*)  
Format time according to the user timezone

**async get\_locale\_time**(*m\_time, user\_timezone=None*)  
Return a string containing locale date format. For now, enforce all weekdays to be en\_US format

**async get\_reminders**(*user\_id*)  
Get the reminders of a user

**async get\_user\_timezone**(*user\_id*)  
Returns the user's timezone

**async process\_absolute\_time\_input**(*time\_input, user\_id*)  
Returns the absolute date time of the input

**async process\_relative\_time\_input**(*time\_input*)  
Returns the relative time of the input in seconds

**async static process\_reminder\_reason**(*user\_input, cutoff\_index*)  
Return the reminder reason that comes before in/at

**async process\_reminder\_time**(*user\_input, type\_index, is\_relative\_time, user\_id*)  
Return the datetime of the reminder depending on the time format

**async static process\_timezone\_input**(*input\_timezone, input\_country\_code=None*)  
Convert timezone abbreviation and country code to standard timezone name

**async remove\_user\_reminder**(*user\_id, reminder\_id*)  
Remove a reminder from the cache and the database.

**async remove\_user\_timezone**(*user\_id*)  
Remove user timezone

**async set\_reminder**(*remind\_reason, remind\_time, user\_id*)  
Add reminder date to cache and db.

**async set\_user\_timezone**(*user\_id, timezone*)  
Set user timezone

## 4.18 u\_selfassignroles

**class** IreneUtility.util.u\_selfassignroles.**SelfAssignRoles**(\*args)

**async add\_self\_role**(*role\_id, role\_name, server\_id*)  
Adds a self-assignable role to a server.

**async check\_for\_self\_assignable\_role**(*message*)  
Main process for processing self-assignable roles.

**async static check\_member\_has\_role**(*member\_roles, role\_id*)  
Check if a member has a role

**async check\_self\_assignable\_channel**(*server\_id, channel*)  
Check if a channel is a self assignable role channel.

**async check\_self\_role\_exists**(*role\_id, role\_name, server\_id*)  
Check if a role exists as a self-assignable role in a server.

**async get\_assignable\_server\_roles**(*server\_id*)  
Get all the self-assignable roles from a server.

**async get\_self\_role**(*message\_content, server\_id*)  
Returns a discord.Object that can be used for adding or removing a role to a member.

**async modify\_channel\_role**(*channel\_id, server\_id*)  
Add or Change a server's self-assignable role channel.

**async process\_member\_roles**(*message, role, role\_name, prefix, author*)  
Adds or removes a (Self-Assignable) role from a member

**async remove\_current\_channel\_role**(*channel\_id, server\_id*)  
Remove the self-assignable role channel if the current channel was previously assigned.

**async remove\_self\_role**(*role\_name, server\_id*)  
Remove a self-assignable role from a server.

## 4.19 u\_twitch

**class** IreneUtility.util.u\_twitch.**Twitch**(\*args)

**async add\_channel**(*twitch\_username, guild\_id*)  
Follows a twitch channel.

**async change\_twitch\_role**(*guild\_id, role\_id*)  
Adds/Changes a twitch role that gets mentioned on twitch updates.

**async check\_channel\_followed**(*twitch\_username, guild\_id*)  
Check if a guild is being followed.

**async check\_guild\_limit**(*guild\_id*)  
check if a guild is allowed to follow more channels

**async delete\_twitch\_role**(*guild\_id*)  
Delete a twitch role mentioned on updates.

**async get\_channels\_followed**(*guild\_id*)  
Get the twitch channels a discord server follows.

**async get\_discord\_channel**(*guild\_id*)  
Get the channel that follow announcements are sent to.

**async remove\_channel**(*twitch\_username, guild\_id*)  
Stop following a twitch channel.

**async reset\_twitch\_token**()  
Get/and reset twitch access token to use on the twitch api.

**async set\_discord\_channel**(*guild\_id, channel\_id*)  
Set the channel for a guild that receives live updates.

## 4.20 u\_twitter

**class** IreneUtility.util.u\_twitter.**Twitter**(\*args)

**get\_random\_idol\_photo**()  
Get a random idol photo existing in the file directory.  
  
This method may block the heartbeat due to OS operation. Should be run separately.

**async upload\_random\_image**()  
Uploads a random (BUT UNIQUE) idol photo to twitter.  
  
**Returns** twitter body message & twitter link to the post.

## 4.21 u\_weverse

**class** IreneUtility.util.u\_weverse.**Weverse**(\*args)

**async add\_weverse\_channel**(channel\_id, community\_name)

Add a channel to get updates for a community

**async add\_weverse\_channel\_to\_cache**(channel\_id, community\_name)

Add a weverse channel to cache.

**async add\_weverse\_role**(channel\_id, community\_name, role\_id)

Add a weverse role to notify.

**async change\_weverse\_comment\_media\_status**(channel\_id, community\_name, t\_disabled,  
updated=False, media=False)

Change a channel's subscription and whether or not they receive updates on comments/comments.

### Parameters

- **channel\_id** – (int) Channel id on discord
- **community\_name** – (str) Community name on Weverse
- **t\_disabled** – (integer) Represents the current status of the disable.
- **updated** – (bool) Whether it needs to be updated.
- **media** – (bool) Whether to disable media or not.

**async change\_weverse\_media\_status**(channel\_id, community\_name, media\_disabled, updated=False)

Change a channel's subscription and whether or not they receive updates on media.

**async check\_weverse\_channel**(channel\_id, community\_name)

Check if a channel is already getting updates for a community

**async delete\_weverse\_channel**(channel\_id, community\_name)

Delete a community from a channel's updates.

**async delete\_weverse\_role**(channel\_id, community\_name)

Remove a weverse role from a server (no longer notifies a role).

**async disable\_type**(ctx, community\_name, media=False)

Disable media/comments on a community and deal with the user messages.

### Parameters

- **ctx** – Context Object
- **community\_name** – Weverse community name
- **media** – Whether the post type is media

**async download\_weverse\_post**(url, file\_name)

Downloads an image url and returns image host url.

If we are to upload from host, it will return the folder location instead.

**async get\_weverse\_channels**(community\_name)

Get all of the channel ids for a specific community name

**async replace\_cache\_role\_id**(channel\_id, community\_name, role\_id)

Replace the server role that gets notified on Weverse Updates.

**async set\_comment\_embed**(*notification, embed\_title*)

Set Comment Embed for Weverse.

**async set\_media\_embed**(*notification, embed\_title*)

Set Media Embed for Weverse.

**async set\_post\_embed**(*notification, embed\_title*)

Set Post Embed for Weverse.

**Returns** Embed and ( a list of file locations OR a string with image urls )



## 5.1 \_\_init\_\_ (s\_sql)

**class** IreneUtility.s\_sql.SqlConnection

Used so that we have a stable reference to our DB Connection. This way we do not need to worry if our connection at any starting point is None as long as it gets set eventually

**self** - The primary instance of *SqlConnection*

## 5.2 db\_structure

**async** IreneUtility.s\_sql.db\_structure.create\_db\_structure()

Creates the db structure based on the existing version sql file.

This prevents having to manually update the db structure across development bots. Before this method is used, there must be a SQL connection whether it is connected to another DB, or creating an empty DB with no schemas just to create the python connection.

SQL File must be in the main directory of the Bot Client.

Includes a blocking File IO Task, but since this is executed on start/run, it really does not matter.

## 5.3 s\_biasgame

## 5.4 s\_blackjack

**async** IreneUtility.s\_sql.s\_blackjack.delete\_playing\_cards()

Delete all custom playing cards from table.

**async** IreneUtility.s\_sql.s\_blackjack.fetch\_playing\_cards()

Fetch playing cards.

**async** IreneUtility.s\_sql.s\_blackjack.generate\_playing\_card(card\_value\_id, bg\_idol\_id) → int

Add a playing card and return the custom id.

Note that we could technically have a file format name of “(card id)\_(idol id).png”, but in the case we would like to add several images for one idol, it would be better that we go by the unique index of the table, even if it does take two more sql queries than needed be.

### Parameters

- **card\_value\_id** – Number from 1 to 52 that represents the custom card.
- **bg\_idol\_id** – Idol ID of the background.

**Returns** Custom ID of card.

## 5.5 s\_cache

**async** IreneUtility.s\_sql.s\_cache.add\_guild(guild)  
Adds a guild to the guild table.

**Parameters** guild – D.py Guild

**async** IreneUtility.s\_sql.s\_cache.remove\_guild(guild)  
Removes a guild from the guild table.

**Parameters** guild – D.py Guild

## 5.6 s\_general

**async** IreneUtility.s\_sql.s\_general.delete\_welcome\_role(guild\_id: int)  
Delete a guild's welcome role.

**async** IreneUtility.s\_sql.s\_general.fetch\_bot\_bans()  
Fetch all bot bans.

**async** IreneUtility.s\_sql.s\_general.fetch\_bot\_statuses()  
Fetch all bot statuses.

**async** IreneUtility.s\_sql.s\_general.fetch\_mod\_mail()  
Fetch mod mail users and channels.

**async** IreneUtility.s\_sql.s\_general.fetch\_n\_word(ordered\_by\_greatest=False)  
Fetch all users N Word count.

**async** IreneUtility.s\_sql.s\_general.fetch\_server\_prefixes()  
Fetch the server prefixes.

**async** IreneUtility.s\_sql.s\_general.fetch\_temp\_channels()  
Fetch all temporary channels

**async** IreneUtility.s\_sql.s\_general.fetch\_welcome\_messages()  
Fetch all welcome messages

**async** IreneUtility.s\_sql.s\_general.fetch\_welcome\_roles()  
Fetch all welcome roles.

**async** IreneUtility.s\_sql.s\_general.insert\_welcome\_role(guild\_id: int, role\_id: int)  
Insert or Update a welcome role.

**async** IreneUtility.s\_sql.s\_general.update\_welcome\_role(guild\_id: int, role\_id: int)  
Update a guild's welcome role.

## 5.7 s\_groupmembers

**async** IreneUtility.s\_sql.s\_groupmembers.delete\_send\_idol\_photo\_channel(*text\_channel\_id: int*)  
Deletes a text channel from receiving photos after t time.

**Parameters** **text\_channel\_id** – ID of the text channel that should no longer receive idol photos.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_aliases(*object\_id, group=False*)  
Fetch all global and server aliases of an idol or group.

**Parameters**

- **object\_id** – An Idol or Group id
- **group** – Whether the object is a group.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_all\_groups()  
Fetch all groups.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_all\_idols()  
Fetch all idols.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_all\_images()  
Fetch all images.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_dead\_links()  
Fetch all dead links.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_members\_in\_group(*group\_id*)  
Fetches the idol ids in a group.

**Parameters** **group\_id** – The group's id

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_restricted\_channels()  
Fetch all restricted idol photo channels.

**async** IreneUtility.s\_sql.s\_groupmembers.fetch\_send\_idol\_photos()  
Fetches the text channels and idols that should be sent to the channel after t time.

**async** IreneUtility.s\_sql.s\_groupmembers.get\_idol\_id\_by\_image\_id(*image\_id: int*)  
Get an idol id from a unique image id.

**Returns** Idol ID or NoneType (if the image id does not exist)

**async** IreneUtility.s\_sql.s\_groupmembers.insert\_send\_idol\_photo(*text\_channel\_id: int, idol\_id: int*)  
Inserts a text channel to receive photos from certain idols.

**Parameters**

- **text\_channel\_id** – ID of the text channel that will receive idol photos from the idol.
- **idol\_id** – The idol id that will have their photos be sent to the text channel.

**async** IreneUtility.s\_sql.s\_groupmembers.update\_send\_idol\_photo(*text\_channel\_id: int, idol\_ids: list*)

Update a text channel's idol list

**Parameters**

- **text\_channel\_id** – ID of the text channel that will receive idol photos from the idol.
- **idol\_ids** – ALL idol ids that should be associated with the text channel

## 5.8 s\_guessinggame

**async** IreneUtility.s\_sql.s\_guessinggame.**fetch\_filter\_enabled**()  
Fetches the users with a guessing game filter enabled.

**async** IreneUtility.s\_sql.s\_guessinggame.**fetch\_filtered\_groups**()  
Fetches all the users and the groups they have filtered.

**async** IreneUtility.s\_sql.s\_guessinggame.**fetch\_gg\_stats**()  
Fetch the user's id, easy, medium, and hard guessing game stats

## 5.9 s\_lastfm

## 5.10 s\_levels

**async** IreneUtility.s\_sql.s\_levels.**create\_level\_row**(*user\_id: int*)  
Create a row in currency.levels for the user.

**Parameters** **user\_id** – Discord User ID

**async** IreneUtility.s\_sql.s\_levels.**fetch\_levels**()  
Fetches all user ids and their rob, daily, beg, and profile level.

**async** IreneUtility.s\_sql.s\_levels.**get\_profile\_xp**(*user\_id: int*)  
Get a user's profile xp.

**Parameters** **user\_id** – Discord User ID

**async** IreneUtility.s\_sql.s\_levels.**level\_row\_exists**(*user\_id: int*)  
Check if a user has a row in the levels table.

**async** IreneUtility.s\_sql.s\_levels.**update\_level**(*user\_id: int, column\_name: str, level: int*)  
Update the level of a user.

**Parameters**

- **user\_id** – Discord User ID
- **column\_name** – Column name of the currency.levels table
- **level** – Level to set column to.

## 5.11 s\_logging

**async** IreneUtility.s\_sql.s\_logging.**fetch\_logged\_channels**(*primay\_key*)  
Fetch the channels of a logged server.

**Parameters** **primay\_key** – The table key of the logged server.

**async** IreneUtility.s\_sql.s\_logging.**fetch\_logged\_servers**()  
Fetch the servers being logged.

## 5.12 s\_miscellaneous

## 5.13 s\_moderator

**async** IreneUtility.s\_sql.s\_moderator.disable\_game\_in\_channel(channel\_id: int)

Disable games in a text channel.

**async** IreneUtility.s\_sql.s\_moderator.enable\_game\_in\_channel(channel\_id: int)

Enable games in a text channel.

**async** IreneUtility.s\_sql.s\_moderator.fetch\_games\_disabled()

Fetch the servers being logged.

## 5.14 s\_patreon

**async** IreneUtility.s\_sql.s\_patreon.add\_patron(user\_id, super\_patron: int)

Add a patron

### Parameters

- **user\_id** – Discord User ID
- **super\_patron** – 1 for the user becoming a super patron, or 0 if they are a normal patron.

**async** IreneUtility.s\_sql.s\_patreon.delete\_patron(user\_id)

Delete a patron.

### Parameters **user\_id** – Discord User ID

**async** IreneUtility.s\_sql.s\_patreon.fetch\_cached\_patrons()

Fetch the cached patrons.

**async** IreneUtility.s\_sql.s\_patreon.update\_patron(user\_id, super\_patron: int)

Updates a patron's status

### Parameters

- **user\_id** – Discord User ID
- **super\_patron** – 1 for the user becoming a super patron, or 0 if they are a normal patron.

## 5.15 s\_reminder

**async** IreneUtility.s\_sql.s\_reminder.fetch\_reminders()

Fetch all reminders. (id, user id, reason, timestamp)

## 5.16 s\_selfassignroles

**async** IreneUtility.s\_sql.s\_selfassignroles.fetch\_all\_self\_assign\_channels()  
Fetch all channel ids and server ids for self assignable roles.

**async** IreneUtility.s\_sql.s\_selfassignroles.fetch\_all\_self\_assign\_roles()  
Fetch all role ids, role names, and server ids for self assignable roles.

## 5.17 s\_session

**async** IreneUtility.s\_sql.s\_session.add\_new\_session(*total\_used*, *session\_commands*, *date*)  
Insert a new session.

### Parameters

- **total\_used** – Total commands used for all sessions.
- **session\_commands** – The amount of commands to give the session (usually 0)
- **date** – Usually datetime.date.today()

**async** IreneUtility.s\_sql.s\_session.fetch\_command(*session\_id*)  
Fetch the command name and its usage amount for a certain session.

**async** IreneUtility.s\_sql.s\_session.fetch\_session\_id(*date*)  
Fetch a session id with a date.

**Parameters** **date** – Usually datetime.date.today()

**async** IreneUtility.s\_sql.s\_session.fetch\_session\_usage(*date*)  
Fetch the session command usage of a date.

**async** IreneUtility.s\_sql.s\_session.fetch\_total\_session\_usage()  
Fetches the total amount of commands used.

## 5.18 s\_twitch

**async** IreneUtility.s\_sql.s\_twitch.check\_twitch\_already\_posted(*twitch\_username*, *channel\_id*) → bool  
Check if a twitch channel being live was already posted to a text cahnnel.

### Parameters

- **twitch\_username** – Twitch username
- **channel\_id** – Text Channel ID

**Returns** True if the live announcement was already posted.

**async** IreneUtility.s\_sql.s\_twitch.delete\_twitch\_posted(*twitch\_username*)  
Delete the text channels from a table that have received messages for a twitch username. :param twitch\_username: Twitch username

**async** IreneUtility.s\_sql.s\_twitch.fetch\_twitch\_guilds()  
Fetch all guild ids, channel ids, and role ids

**async** IreneUtility.s\_sql.s\_twitch.fetch\_twitch\_notifications()  
Fetch all twitch username and guild ids that announcements should be sent to.

**async** IreneUtility.s\_sql.s\_twitch.set\_twitch\_posted(*twitch\_username, channel\_id*)  
 Set a discord text channel to have already been sent a message from twitch announcements. :param twitch\_username: Twitch username :param channel\_id: Text Channel ID

## 5.19 s\_twitter

**async** IreneUtility.s\_sql.s\_twitter.check\_photo\_uploaded(*image\_id*)  
 Checks if a photo was already uploaded.

**Parameters** *image\_id* – The unique image id.

**Returns** Count of the image id in the table (should be 0 or 1)

**async** IreneUtility.s\_sql.s\_twitter.insert\_photo\_uploaded(*image\_id, media\_id*)  
 Insert an image so that we can keep track of twitter media uploads.

**Parameters**

- *image\_id* – Unique Image ID
- *media\_id* – Twitter Media ID

## 5.20 s\_user

**async** IreneUtility.s\_sql.s\_user.delete\_user\_language(*user\_id: int*)  
 Deletes the user from the language table. Default is en\_us [Which should not exist in the language table]

**Parameters** *user\_id* –

**async** IreneUtility.s\_sql.s\_user.fetch\_languages()  
 Fetches all user ids and their language preference.

**async** IreneUtility.s\_sql.s\_user.fetch\_timezones()  
 Fetch all timezones. (user id, timezone)

**async** IreneUtility.s\_sql.s\_user.set\_user\_language(*user\_id: int, language: str*)  
 Set the user's client language.

**Parameters**

- *user\_id* –
- *language* –

## 5.21 s\_weverse

**async** IreneUtility.s\_sql.s\_weverse.fetch\_weverse()  
 Fetch all weverse subscriptions.





## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### i

IreneUtility.models, 5  
IreneUtility.s\_sql, 29  
IreneUtility.s\_sql.db\_structure, 29  
IreneUtility.s\_sql.s\_biasgame, 29  
IreneUtility.s\_sql.s\_blackjack, 29  
IreneUtility.s\_sql.s\_cache, 30  
IreneUtility.s\_sql.s\_general, 30  
IreneUtility.s\_sql.s\_groupmembers, 31  
IreneUtility.s\_sql.s\_guessinggame, 32  
IreneUtility.s\_sql.s\_lastfm, 32  
IreneUtility.s\_sql.s\_levels, 32  
IreneUtility.s\_sql.s\_logging, 32  
IreneUtility.s\_sql.s\_miscellaneous, 33  
IreneUtility.s\_sql.s\_moderator, 33  
IreneUtility.s\_sql.s\_patreon, 33  
IreneUtility.s\_sql.s\_reminder, 33  
IreneUtility.s\_sql.s\_selfassignroles, 34  
IreneUtility.s\_sql.s\_session, 34  
IreneUtility.s\_sql.s\_twitch, 34  
IreneUtility.s\_sql.s\_twitter, 35  
IreneUtility.s\_sql.s\_user, 35  
IreneUtility.s\_sql.s\_weverse, 35  
IreneUtility.util.u\_biasgame, 13  
IreneUtility.util.u\_blackjack, 13  
IreneUtility.util.u\_cache, 14  
IreneUtility.util.u\_customcommands, 16  
IreneUtility.util.u\_database, 16  
IreneUtility.util.u\_datadog, 16  
IreneUtility.util.u\_exceptions, 16  
IreneUtility.util.u\_gacha, 17  
IreneUtility.util.u\_groupmembers, 17  
IreneUtility.util.u\_guessinggame, 20  
IreneUtility.util.u\_lastfm, 20  
IreneUtility.util.u\_local\_cache, 21  
IreneUtility.util.u\_logger, 21  
IreneUtility.util.u\_miscellaneous, 22  
IreneUtility.util.u\_moderator, 23  
IreneUtility.util.u\_patreon, 24  
IreneUtility.util.u\_reminder, 24  
IreneUtility.util.u\_selfassignroles, 25  
IreneUtility.util.u\_twitch, 26  
IreneUtility.util.u\_twitter, 26  
IreneUtility.util.u\_weverse, 27



## A

ACCESS\_SECRET (*IreneUtility.models.Keys* attribute), 8  
 add\_channel() (*IreneUtility.util.u\_twitch.Twitch* method), 26  
 add\_command\_count() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22  
 add\_guild() (in module *IreneUtility.s\_sql.s\_cache*), 30  
 add\_new\_session() (in module *IreneUtility.s\_sql.s\_session*), 34  
 add\_patron() (in module *IreneUtility.s\_sql.s\_patreon*), 33  
 add\_self\_role() (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
 add\_session\_count() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22  
 add\_to\_patreon() (*IreneUtility.util.u\_patreon.Patreon* method), 24  
 add\_welcome\_message\_server() (*IreneUtility.util.u\_moderator.Moderator* method), 23  
 add\_weverse\_channel() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 add\_weverse\_channel\_to\_cache() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 add\_weverse\_role() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 Album (class in *IreneUtility.models*), 5  
 announce\_winner() (*IreneUtility.models.BlackJackGame* method), 6  
 api\_port (*IreneUtility.models.Keys* attribute), 8  
 apply\_bold\_to\_braces() (*IreneUtility.util.u\_cache.Cache* static method), 14

## B

ban\_user\_from\_bot() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22  
 Base (class in *IreneUtility.Base*), 3  
 BaseUtil (class in *IreneUtility.models*), 5

BiasGame (class in *IreneUtility.models*), 5  
 BiasGame (class in *IreneUtility.util.u\_biasgame*), 13  
 BlackJack (class in *IreneUtility.util.u\_blackjack*), 13  
 BlackJackGame (class in *IreneUtility.models*), 6

## C

Cache (class in *IreneUtility.util.u\_cache*), 14  
 Cache (class in *IreneUtility.util.u\_local\_cache*), 21  
 calculate\_dance\_score() (*IreneUtility.models.Album* method), 5  
 calculate\_rap\_score() (*IreneUtility.models.Album* method), 5  
 calculate\_score() (*IreneUtility.models.BlackJackGame* method), 6  
 calculate\_vocal\_score() (*IreneUtility.models.Album* method), 5  
 change\_twitch\_role() (*IreneUtility.util.u\_twitch.Twitch* method), 26  
 change\_weverse\_comment\_media\_status() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 change\_weverse\_media\_status() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 check\_channel\_followed() (*IreneUtility.util.u\_twitch.Twitch* method), 26  
 check\_channel\_sending\_photos() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17  
 check\_for\_bot\_mentions() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22  
 check\_for\_nword() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22  
 check\_for\_self\_assignable\_role() (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
 check\_group\_and\_idol() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17  
 check\_guild\_limit() (*IreneUtility.util.u\_twitch.Twitch* method), 26

`check_idol_post_reactions()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17

`check_if_bot_banned()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22

`check_if_moderator()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* static method), 22

`check_if_patreon()` (*IreneUtility.util.u\_patreon.Patreon* method), 24

`check_if_temp_channel()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22

`check_member_has_role()` (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* static method), 25

`check_message()` (*IreneUtility.models.BiasGame* method), 5

`check_message()` (*IreneUtility.models.BlackJackGame* method), 6

`check_message()` (*IreneUtility.models.GuessingGame* method), 7

`check_message_is_command()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 22

`check_message_not_empty()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* static method), 23

`check_nword()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23

`check_photo_uploaded()` (in module *IreneUtility.s\_sql.s\_twitter*), 35

`check_self_assignable_channel()` (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25

`check_self_role_exists()` (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25

`check_server_sending_photos()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17

`check_standing()` (*IreneUtility.models.BlackJackGame* method), 6

`check_to_add_alias_to_list()` (*IreneUtility.util.u\_groupmembers.GroupMembers* static method), 17

`check_twitch_already_posted()` (in module *IreneUtility.s\_sql.s\_twitch*), 34

`check_welcome_message_enabled()` (*IreneUtility.util.u\_moderator.Moderator* method), 23

`check_weverse_channel()` (*IreneUtility.util.u\_weverse.Weverse* method), 27

`choose_random_card()` (*IreneUtility.models.BlackJackGame* method), 6

`choose_random_member()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17

`client` (*IreneUtility.models.Keys* attribute), 8

`client_session` (*IreneUtility.models.Keys* attribute), 8

`commands_used` (*IreneUtility.util.u\_local\_cache.Cache* attribute), 21

`connect_to_db()` (*IreneUtility.models.Keys* method), 8

`console()` (in module *IreneUtility.util.u\_logger*), 21

`create_acceptable_answers()` (*IreneUtility.models.GuessingGame* method), 7

`create_bias_game_image()` (*IreneUtility.util.u\_biasgame.BiasGame* method), 13

`create_bot_bans()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_bot_command_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_command_counter()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_currency_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_db_structure()` (in module *IreneUtility.s\_sql.db\_structure*), 29

`create_dead_link_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_disabled_games_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_fm_payload()` (*IreneUtility.util.u\_lastfm.LastFM* method), 20

`create_gg_filter_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_group_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_groups()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_guessing_game_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_guild_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_idol_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_idol_pool()` (*IreneUtility.models.GuessingGame* method), 8

`create_idols()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_image_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

`create_language_cache()` (*IreneUtility.util.u\_cache.Cache* method), 14

create\_level\_row() (in module *IreneUtility.s\_sql.s\_levels*), 32  
 create\_levels\_cache() (*IreneUtility.util.u\_cache.Cache* method), 14  
 create\_logging\_channels() (*IreneUtility.util.u\_cache.Cache* method), 14  
 create\_mod\_mail() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_n\_word\_counter() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_new\_question() (*IreneUtility.models.GuessingGame* method), 8  
 create\_patreons() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_playing\_cards() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_reminder\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_restricted\_channel\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_self\_assignable\_role\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_send\_idol\_photo\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_server\_prefixes() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_temp\_channels() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_timezone\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_twitch\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_unscramble\_game\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_user\_in\_guessing\_game() (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
 create\_user\_notifications() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_welcome\_message\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 create\_weverse\_channel\_cache() (*IreneUtility.util.u\_cache.Cache* method), 15  
 credit\_user() (*IreneUtility.models.GuessingGame* method), 8  
 CustomCommands (class in *IreneUtility.util.u\_customcommands*), 16

deal\_with\_bets() (*IreneUtility.models.BlackJackGame* method), 6  
 delete\_channel\_from\_send\_idol() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 17  
 delete\_patron() (in module *IreneUtility.s\_sql.s\_patreon*), 33  
 delete\_playing\_cards() (in module *IreneUtility.s\_sql.s\_blackjack*), 29  
 delete\_restricted\_channel\_from\_cache() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
 delete\_send\_idol\_photo\_channel() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
 delete\_temp\_messages() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
 delete\_twitch\_posted() (in module *IreneUtility.s\_sql.s\_twitch*), 34  
 delete\_twitch\_role() (*IreneUtility.util.u\_twitch.Twitch* method), 26  
 delete\_user\_language() (in module *IreneUtility.s\_sql.s\_user*), 35  
 delete\_welcome\_role() (in module *IreneUtility.s\_sql.s\_general*), 30  
 delete\_weverse\_channel() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 delete\_weverse\_role() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 determine\_time\_type() (*IreneUtility.util.u\_reminder.Reminder* static method), 24  
 determine\_winner() (*IreneUtility.models.BlackJackGame* method), 6  
 disable\_game\_in\_channel() (in module *IreneUtility.s\_sql.s\_moderator*), 33  
 disable\_interaction() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
 disable\_type() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
 discord\_boats (*IreneUtility.models.Keys* attribute), 9  
 display\_winners() (*IreneUtility.models.GuessingGame* method), 8  
 download\_weverse\_post() (*IreneUtility.util.u\_weverse.Weverse* method), 27

## D

DataBase (class in *IreneUtility.util.u\_database*), 16  
 DataDog (class in *IreneUtility.util.u\_datadog*), 16  
 datadog\_app\_key (*IreneUtility.models.Keys* attribute), 9  
 db\_conn (*IreneUtility.models.Keys* attribute), 9

## E

enable\_game\_in\_channel() (in module *IreneUtility.s\_sql.s\_moderator*), 33  
 enable\_interaction() (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
 end\_game() (*IreneUtility.models.BiasGame* method), 5

end\_game() (*IreneUtility.models.BlackJackGame* method), 6  
end\_game() (*IreneUtility.models.Game* method), 7  
end\_game() (*IreneUtility.models.GuessingGame* method), 8  
ensure\_level() (*IreneUtility.models.User* method), 10

## F

fetch\_aliases() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_all\_groups() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_all\_idols() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_all\_images() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_all\_self\_assign\_channels() (in module *IreneUtility.s\_sql.s\_selfassignroles*), 34  
fetch\_all\_self\_assign\_roles() (in module *IreneUtility.s\_sql.s\_selfassignroles*), 34  
fetch\_bot\_bans() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_bot\_statuses() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_cached\_patrons() (in module *IreneUtility.s\_sql.s\_patreon*), 33  
fetch\_command() (in module *IreneUtility.s\_sql.s\_session*), 34  
fetch\_dead\_links() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_filter\_enabled() (in module *IreneUtility.s\_sql.s\_guessinggame*), 32  
fetch\_filtered\_groups() (in module *IreneUtility.s\_sql.s\_guessinggame*), 32  
fetch\_games\_disabled() (in module *IreneUtility.s\_sql.s\_moderator*), 33  
fetch\_gg\_stats() (in module *IreneUtility.s\_sql.s\_guessinggame*), 32  
fetch\_languages() (in module *IreneUtility.s\_sql.s\_user*), 35  
fetch\_levels() (in module *IreneUtility.s\_sql.s\_levels*), 32  
fetch\_logged\_channels() (in module *IreneUtility.s\_sql.s\_logging*), 32  
fetch\_logged\_servers() (in module *IreneUtility.s\_sql.s\_logging*), 32  
fetch\_members\_in\_group() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_mod\_mail() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_n\_word() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_playing\_cards() (in module *IreneUtility.s\_sql.s\_blackjack*), 29

fetch\_reminders() (in module *IreneUtility.s\_sql.s\_reminder*), 33  
fetch\_restricted\_channels() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_send\_idol\_photos() (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
fetch\_server\_prefixes() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_session\_id() (in module *IreneUtility.s\_sql.s\_session*), 34  
fetch\_session\_usage() (in module *IreneUtility.s\_sql.s\_session*), 34  
fetch\_temp\_channels() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_timezones() (in module *IreneUtility.s\_sql.s\_user*), 35  
fetch\_total\_session\_usage() (in module *IreneUtility.s\_sql.s\_session*), 34  
fetch\_twitch\_guilds() (in module *IreneUtility.s\_sql.s\_twitch*), 34  
fetch\_twitch\_notifications() (in module *IreneUtility.s\_sql.s\_twitch*), 34  
fetch\_welcome\_messages() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_welcome\_roles() (in module *IreneUtility.s\_sql.s\_general*), 30  
fetch\_weverse() (in module *IreneUtility.s\_sql.s\_weverse*), 35  
File (class in *IreneUtility.models*), 7  
filter\_add\_group() (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
filter\_auto\_add\_remove\_group() (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
filter\_remove\_group() (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
finalize\_game() (*IreneUtility.models.BlackJackGame* method), 6  
find\_game() (*IreneUtility.util.u\_blackjack.BlackJack* method), 13  
format\_card\_fields() (*IreneUtility.util.u\_groupmembers.GroupMembers* static method), 18  
format\_time() (*IreneUtility.util.u\_reminder.Reminder* static method), 24

## G

Gacha (class in *IreneUtility.util.u\_gacha*), 17  
GachaValues (class in *IreneUtility.models*), 7  
Game (class in *IreneUtility.models*), 7  
generate\_brackets() (*IreneUtility.models.BiasGame* method), 5



`generate_playing_card()` (in module *IreneUtility.s\_sql.s\_blackjack*), 29  
`generate_playing_cards()` (*IreneUtility.util.u\_blackjack.BlackJack* method), 13  
`get_all_groups()` (*IreneUtility.util.u\_groupmembers.GroupMembers* static method), 18  
`get_all_images_count()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_all_skill_scores()` (*IreneUtility.util.u\_gacha.Gacha* static method), 17  
`get_assignable_server_roles()` (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
`get_channel_count()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
`get_channel_sending_photos()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_channels_followed()` (*IreneUtility.util.u\_twitch.Twitch* method), 26  
`get_class()` (in module *IreneUtility.util.u\_logger*), 21  
`get_cooldown_time()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* static method), 23  
`get_daily_amount()` (*IreneUtility.models.User* method), 10  
`get_db_aliases()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_db_connection()` (*IreneUtility.util.u\_database.DataBase* method), 16  
`get_db_groups_from_member()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_db_idol_called()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_db_members_in_group()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_disabled_server_interactions()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
`get_discord_channel()` (*IreneUtility.util.u\_twitch.Twitch* method), 26  
`get_fm_response()` (*IreneUtility.util.u\_lastfm.LastFM* method), 20  
`get_fm_username()` (*IreneUtility.util.u\_lastfm.LastFM* method), 20  
`get_google_drive_link()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_group()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_group_names_as_string()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_group_where_group_matches_name()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_guessing_game_top_ten()` (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
`get_idol_by_image_id()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_idol_id_by_image_id()` (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
`get_idol_post_embed()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_idol_where_member_matches_name()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_int_index()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* static method), 23  
`get_kwarg()` (*IreneUtility.models.Keys* method), 9  
`get_language_code()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous* method), 23  
`get_locale_time()` (*IreneUtility.util.u\_reminder.Reminder* method), 24  
`get_member()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_member_names_as_string()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18  
`get_metric_info()` (*IreneUtility.util.u\_datadog.DataDog* method), 16  
`get_needed_for_level()` (*IreneUtility.models.User* static method), 10  
`get_patreon_role_members()` (*IreneUtility.util.u\_patreon.Patreon* method), 24  
`get_patreon_users()` (*IreneUtility.util.u\_patreon.Patreon* method), 24  
`get_profile_xp()` (in module *IreneUtility.s\_sql.s\_levels*), 32  
`get_profile_xp()` (*IreneUtility.models.User* method), 10  
`get_random_idol()` (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 18

`get_random_idol_photo()` (*IreneUtility.util.u\_twitter.Twitter method*), 26  
`get_reminders()` (*IreneUtility.util.u\_reminder.Reminder method*), 24  
`get_rob_amount()` (*IreneUtility.models.User method*), 10  
`get_rob_percentage()` (*IreneUtility.models.User method*), 10  
`get_self_role()` (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles method*), 25  
`get_server_count()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23  
`get_session_id()` (*IreneUtility.util.u\_cache.Cache method*), 15  
`get_shortened_balance()` (*IreneUtility.models.User method*), 10  
`get_text_channel_count()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23  
`get_user_count()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23  
`get_user_timezone()` (*IreneUtility.util.u\_reminder.Reminder method*), 24  
`get_voice_channel_count()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23  
`get_weverse_channels()` (*IreneUtility.util.u\_weverse.Weverse method*), 27  
`Group` (class in *IreneUtility.models*), 7  
`GroupMembers` (class in *IreneUtility.util.u\_groupmembers*), 17  
`GuessingGame` (class in *IreneUtility.models*), 7  
`GuessingGame` (class in *IreneUtility.util.u\_guessinggame*), 20  

## H

`hit()` (*IreneUtility.models.BlackJackGame method*), 6  

## I

`Idol` (class in *IreneUtility.models*), 8  
`idol_photo_location` (*IreneUtility.models.Keys attribute*), 9  
`idol_post()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 19  
`IdolCard` (class in *IreneUtility.models*), 8  
`ImproperFormat`, 16  
`initialize_data_dog()` (*IreneUtility.util.u\_datadog.DataDog method*), 16  
`insert_photo_uploaded()` (in module *IreneUtility.s\_sql.s\_twitter*), 35  
`insert_send_idol_photo()` (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
`insert_welcome_role()` (in module *IreneUtility.s\_sql.s\_general*), 30  
`InvalidParamsPassed`, 16  
`IreneUtility.models` module, 5  
`IreneUtility.s_sql` module, 29  
`IreneUtility.s_sql.db_structure` module, 29  
`IreneUtility.s_sql.s_biasgame` module, 29  
`IreneUtility.s_sql.s_blackjack` module, 29  
`IreneUtility.s_sql.s_cache` module, 30  
`IreneUtility.s_sql.s_general` module, 30  
`IreneUtility.s_sql.s_groupmembers` module, 31  
`IreneUtility.s_sql.s_guessinggame` module, 32  
`IreneUtility.s_sql.s_lastfm` module, 32  
`IreneUtility.s_sql.s_levels` module, 32  
`IreneUtility.s_sql.s_logging` module, 32  
`IreneUtility.s_sql.s_miscellaneous` module, 33  
`IreneUtility.s_sql.s_moderator` module, 33  
`IreneUtility.s_sql.s_patreon` module, 33  
`IreneUtility.s_sql.s_reminder` module, 33  
`IreneUtility.s_sql.s_selfassignroles` module, 34  
`IreneUtility.s_sql.s_session` module, 34  
`IreneUtility.s_sql.s_twitch` module, 34  
`IreneUtility.s_sql.s_twitter` module, 35  
`IreneUtility.s_sql.s_user` module, 35  
`IreneUtility.s_sql.s_weverse` module, 35  
`IreneUtility.util.u_biasgame` module, 13  
`IreneUtility.util.u_blackjack` module, 13  
`IreneUtility.util.u_cache`

module, 14  
 IreneUtility.util.u\_customcommands  
   module, 16  
 IreneUtility.util.u\_database  
   module, 16  
 IreneUtility.util.u\_datadog  
   module, 16  
 IreneUtility.util.u\_exceptions  
   module, 16  
 IreneUtility.util.u\_gacha  
   module, 17  
 IreneUtility.util.u\_groupmembers  
   module, 17  
 IreneUtility.util.u\_guessinggame  
   module, 20  
 IreneUtility.util.u\_lastfm  
   module, 20  
 IreneUtility.util.u\_local\_cache  
   module, 21  
 IreneUtility.util.u\_logger  
   module, 21  
 IreneUtility.util.u\_miscellaneous  
   module, 22  
 IreneUtility.util.u\_moderator  
   module, 23  
 IreneUtility.util.u\_patreon  
   module, 24  
 IreneUtility.util.u\_reminder  
   module, 24  
 IreneUtility.util.u\_selfassignroles  
   module, 25  
 IreneUtility.util.u\_twitch  
   module, 26  
 IreneUtility.util.u\_twitter  
   module, 26  
 IreneUtility.util.u\_weverse  
   module, 27

## K

Keys (class in IreneUtility.models), 8  
 kwargs (IreneUtility.models.Keys attribute), 9

## L

last\_fm\_headers (IreneUtility.models.Keys attribute), 9  
 LastFM (class in IreneUtility.util.u\_lastfm), 20  
 level\_row\_exists() (in module IreneUtility.s\_sql.s\_levels), 32  
 Limit, 16  
 list\_of\_logged\_channels (IreneUtility.util.u\_local\_cache.Cache attribute), 21  
 load\_language\_packs() (IreneUtility.util.u\_cache.Cache method), 15

log\_idol\_command() (IreneUtility.util.u\_groupmembers.GroupMembers method), 19  
 logfile() (in module IreneUtility.util.u\_logger), 21  
 lyric\_client (IreneUtility.models.Keys attribute), 9

## M

manage\_log() (in module IreneUtility.util.u\_logger), 22  
 manage\_send\_idol\_photo() (IreneUtility.util.u\_groupmembers.GroupMembers method), 19  
 MaxAttempts, 16  
 merge\_images() (IreneUtility.util.u\_biasgame.BiasGame method), 13  
 merge\_images() (IreneUtility.util.u\_blackjack.BlackJack method), 13  
 Miscellaneous (class in IreneUtility.util.u\_miscellaneous), 22  
 Moderator (class in IreneUtility.util.u\_moderator), 23  
 modify\_channel\_role() (IreneUtility.util.u\_selfassignroles.SelfAssignRoles method), 25

## module

IreneUtility.models, 5  
 IreneUtility.s\_sql, 29  
 IreneUtility.s\_sql.db\_structure, 29  
 IreneUtility.s\_sql.s\_biasgame, 29  
 IreneUtility.s\_sql.s\_blackjack, 29  
 IreneUtility.s\_sql.s\_cache, 30  
 IreneUtility.s\_sql.s\_general, 30  
 IreneUtility.s\_sql.s\_groupmembers, 31  
 IreneUtility.s\_sql.s\_guessinggame, 32  
 IreneUtility.s\_sql.s\_lastfm, 32  
 IreneUtility.s\_sql.s\_levels, 32  
 IreneUtility.s\_sql.s\_logging, 32  
 IreneUtility.s\_sql.s\_miscellaneous, 33  
 IreneUtility.s\_sql.s\_moderator, 33  
 IreneUtility.s\_sql.s\_patreon, 33  
 IreneUtility.s\_sql.s\_reminder, 33  
 IreneUtility.s\_sql.s\_selfassignroles, 34  
 IreneUtility.s\_sql.s\_session, 34  
 IreneUtility.s\_sql.s\_twitch, 34  
 IreneUtility.s\_sql.s\_twitter, 35  
 IreneUtility.s\_sql.s\_user, 35  
 IreneUtility.s\_sql.s\_weverse, 35  
 IreneUtility.util.u\_biasgame, 13  
 IreneUtility.util.u\_blackjack, 13  
 IreneUtility.util.u\_cache, 14  
 IreneUtility.util.u\_customcommands, 16  
 IreneUtility.util.u\_database, 16  
 IreneUtility.util.u\_datadog, 16  
 IreneUtility.util.u\_exceptions, 16  
 IreneUtility.util.u\_gacha, 17  
 IreneUtility.util.u\_groupmembers, 17

`IreneUtility.util.u_guessinggame`, 20  
`IreneUtility.util.u_lastfm`, 20  
`IreneUtility.util.u_local_cache`, 21  
`IreneUtility.util.u_logger`, 21  
`IreneUtility.util.u_miscellaneous`, 22  
`IreneUtility.util.u_moderator`, 23  
`IreneUtility.util.u_patreon`, 24  
`IreneUtility.util.u_reminder`, 24  
`IreneUtility.util.u_selfassignroles`, 25  
`IreneUtility.util.u_twitch`, 26  
`IreneUtility.util.u_twitter`, 26  
`IreneUtility.util.u_weverse`, 27

## N

`next_emoji` (*IreneUtility.models.Keys* attribute), 9  
`NoKeyFound`, 16  
`NoTimeZone`, 17

## O

`oxford_app_key` (*IreneUtility.models.Keys* attribute), 9

## P

`Pass`, 17  
`Patreon` (class in *IreneUtility.util.u\_patreon*), 24  
`patreon_super_role_id` (*IreneUtility.models.Keys* attribute), 9  
`playing_card_location` (*IreneUtility.models.Keys* attribute), 9  
`PlayingCard` (class in *IreneUtility.models*), 10  
`print_answer`() (*IreneUtility.models.GuessingGame* method), 8  
`process_absolute_time_input`() (*IreneUtility.util.u\_reminder.Reminder* method), 24  
`process_cache_time`() (*IreneUtility.util.u\_cache.Cache* method), 15  
`process_game`() (*IreneUtility.models.BiasGame* method), 6  
`process_game`() (*IreneUtility.models.BlackJackGame* method), 6  
`process_game`() (*IreneUtility.models.Game* method), 7  
`process_game`() (*IreneUtility.models.GuessingGame* method), 8  
`process_member_roles`() (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
`process_names`() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 19  
`process_relative_time_input`() (*IreneUtility.util.u\_reminder.Reminder* method), 25  
`process_reminder_reason`() (*IreneUtility.util.u\_reminder.Reminder* static method), 25

`process_reminder_time`() (*IreneUtility.util.u\_reminder.Reminder* method), 25  
`process_session`() (*IreneUtility.util.u\_cache.Cache* method), 15  
`process_timezone_input`() (*IreneUtility.util.u\_reminder.Reminder* static method), 25

## R

`random_album_popularity`() (*IreneUtility.util.u\_gacha.Gacha* static method), 17  
`random_skill_score`() (*IreneUtility.util.u\_gacha.Gacha* method), 17  
`register_currency`() (*IreneUtility.models.User* method), 10  
`Reminder` (class in *IreneUtility.util.u\_reminder*), 24  
`remove_all_card_files`() (*IreneUtility.util.u\_blackjack.BlackJack* method), 13  
`remove_channel`() (*IreneUtility.util.u\_twitch.Twitch* method), 26  
`remove_current_channel_role`() (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
`remove_from_patreon`() (*IreneUtility.util.u\_patreon.Patreon* method), 24  
`remove_global_alias`() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 19  
`remove_guild`() (in module *IreneUtility.s\_sql.s\_cache*), 30  
`remove_local_alias`() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 19  
`remove_self_role`() (*IreneUtility.util.u\_selfassignroles.SelfAssignRoles* method), 25  
`remove_user_reminder`() (*IreneUtility.util.u\_reminder.Reminder* method), 25  
`remove_user_timezone`() (*IreneUtility.util.u\_reminder.Reminder* method), 25  
`replace_cache_role_id`() (*IreneUtility.util.u\_weverse.Weverse* method), 27  
`request_image_post`() (*IreneUtility.util.u\_groupmembers.GroupMembers* method), 19  
`request_support_server_members`() (*IreneUtility.util.u\_cache.Cache* method), 15  
`request_twitter_channel`() (*IreneUtility.util.u\_cache.Cache* method), 16  
`reset_patreon_cooldown`() (*IreneUtility.util.u\_patreon.Patreon* method), 24  
`reset_twitch_token`() (*IreneUtility.util.u\_twitch.Twitch* method), 26

`run_current_bracket()` (*IreneUtility.models.BiasGame method*), 6

## S

`SelfAssignRoles` (class in *IreneUtility.util.u\_selfassignroles*), 25

`send_ban_message()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23

`send_file()` (*IreneUtility.models.File method*), 7

`send_metric()` (*IreneUtility.util.u\_datadog.DataDog method*), 16

`send_metrics()` (*IreneUtility.util.u\_datadog.DataDog method*), 16

`send_names()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 19

`send_vote_message()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 19

`server_prefixes` (*IreneUtility.util.u\_local\_cache.Cache attribute*), 21

`set_as_group_photo()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 19

`set_comment_embed()` (*IreneUtility.util.u\_weverse.Weverse method*), 27

`set_discord_channel()` (*IreneUtility.util.u\_twitch.Twitch method*), 26

`set_embed_card_info()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 19

`set_embed_with_aliases()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 20

`set_embed_with_all_aliases()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 20

`set_fm_username()` (*IreneUtility.util.u\_lastfm.LastFM method*), 21

`set_global_alias()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 20

`set_language()` (*IreneUtility.models.User method*), 10

`set_level()` (*IreneUtility.models.User method*), 10

`set_local_alias()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 20

`set_media_embed()` (*IreneUtility.util.u\_weverse.Weverse method*), 28

`set_post_embed()` (*IreneUtility.util.u\_weverse.Weverse method*), 28

`set_profile_xp()` (*IreneUtility.models.User method*), 10

`set_reminder()` (*IreneUtility.util.u\_reminder.Reminder method*), 25

`set_twitch_posted()` (in module *IreneUtility.s\_sql.s\_twitch*), 34

`set_user_language()` (in module *IreneUtility.s\_sql.s\_user*), 35

`set_user_timezone()` (*IreneUtility.util.u\_reminder.Reminder method*), 25

`ShouldNotBeHere`, 17

`skill_completion_multiplier()` (*IreneUtility.models.Album method*), 5

`spotify_client_secret` (*IreneUtility.models.Keys attribute*), 9

`SqlConnection` (class in *IreneUtility.s\_sql*), 29

`stand()` (*IreneUtility.models.BlackJackGame method*), 6

## T

`tenor_key` (*IreneUtility.models.Keys attribute*), 9

`test_client_token` (*IreneUtility.models.Keys attribute*), 9

`toggle_filter()` (*IreneUtility.util.u\_guessinggame.GuessingGame method*), 20

`toggle_games()` (*IreneUtility.util.u\_moderator.Moderator method*), 23

`TooLarge`, 17

`top_gg` (*IreneUtility.models.Keys attribute*), 9

`translate_private_key` (*IreneUtility.models.Keys attribute*), 9

`try_to_rob_user()` (*IreneUtility.models.User method*), 10

`Twitch` (class in *IreneUtility.util.u\_twitch*), 26

`twitch_channels_is_live` (*IreneUtility.util.u\_local\_cache.Cache attribute*), 21

`twitch_client_secret` (*IreneUtility.models.Keys attribute*), 9

`Twitter` (class in *IreneUtility.util.u\_twitter*), 26

## U

`unban_user_from_bot()` (*IreneUtility.util.u\_miscellaneous.Miscellaneous method*), 23

`update_balance()` (*IreneUtility.models.User method*), 10

`update_level()` (in module *IreneUtility.s\_sql.s\_levels*), 32

`update_level_in_db()` (*IreneUtility.models.User method*), 11

`update_member_count()` (*IreneUtility.util.u\_groupmembers.GroupMembers method*), 20

`update_patron()` (in module *IreneUtility.s\_sql.s\_patreon*), 33

`update_scores()` (*IreneUtility.models.GuessingGame* method), 8  
`update_send_idol_photo()` (in module *IreneUtility.s\_sql.s\_groupmembers*), 31  
`update_user_guessing_game_score()` (*IreneUtility.util.u\_guessinggame.GuessingGame* method), 20  
`update_welcome_message_channel()` (*IreneUtility.util.u\_moderator.Moderator* method), 24  
`update_welcome_message_enabled()` (*IreneUtility.util.u\_moderator.Moderator* method), 24  
`update_welcome_role()` (in module *IreneUtility.s\_sql.s\_general*), 30  
`upload_random_image()` (*IreneUtility.util.u\_twitter.Twitter* method), 26  
`useless()` (in module *IreneUtility.util.u\_logger*), 22  
`User` (class in *IreneUtility.models*), 10

## W

`welcome_messages` (*IreneUtility.util.u\_local\_cache.Cache* attribute), 21  
`Weverse` (class in *IreneUtility.util.u\_weverse*), 27  
`weverse_image_folder` (*IreneUtility.models.Keys* attribute), 9  
`wolfram_app_id` (*IreneUtility.models.Keys* attribute), 9  
`write_to_file()` (in module *IreneUtility.util.u\_logger*), 22

## X

`X_RapidAPI_headers` (*IreneUtility.models.Keys* attribute), 8