Functions

Source

crawlinsta.login

`login(username: str, password: str) → bool` [source]

Login to instagram with given credential, and return True if success, else False.

Parameters:
- **username** (*str*) – username for login.
- **password** (*str*) – corresponding password for login.

Returns: True, if login successes; otherwise False.

Return type: bool

crawlinsta.collecting

, n: int

`collect_comments_of_media(media_code: str) → Json` [source]

Collect comments of a given media.

Strategy: click the media (a button with link */p/<media_code>/*), a list of comments shows in a popup. A request triggered by the clicking was sent to the path */api/v1/media/<media_id>/comments/?can_support_threading=true&permalink_enabled=false* The comments are paginated, to load more comments, have to use the mouse to click a button to load more comments.

Parameters: media_code (*str*) – media short code for generating access url.

Returns: all comments of the given media in json format.

Return type: Json

+

n (*int*) – maximum number of comments, which should be collected

`collect_followers_of_user(username: str) → Json` [source]

+n

Collect followers of the given user.

Strategy: click the username, get to the main page of the user, then click followers, a list of followers of the user shows in a popup. A request triggered by the clicking was sent to the path */api/v1/friendships/<user_id>/followers/?count=12&search_surface=follow_list_page*.

Parameters: **username** (*str*) – name of the user. +n

Returns: all visible followers' user information of the given user in json format.

Return type: Json

1) ~~Rename „collect_comments_of_media“ to „collect_comments“~~ Done

2) ~~Here and in all other places in the documentation, replace the word „media“ with „post“ (because media refers to an image or video an post)~~ Done

3) ~~Replace media_code with post_id~~ Done

4) I don‘t understand the URL here. Is this coming from instagram? Why is the word API in this URL? Do you use the Instagram API?
The URL here is just to help me to keep track of the request path, such that I can write the code easily later to extract the corresponding response body.

5) ~~Include a parameter „n“ which refers to the maximum number of comments. The collect_comment function should scroll down to the page until no more comments can be loaded or the maximum number of comments n is reached. Per default, n is set to 100.~~
~~(! Also check, whether Instagram has any limits, eg. It will always only show up to 1000 comments!)~~ Done

1) ~~Rename „collect_comments_of_user“ to „collect_followers“~~ Done

5) ~~Include a parameter „n“ which refers to the maximum number of followers (Like the n in the collect_comment function). Per default, n is set to 100~~ Done

collect_following~~_of_user~~(*username: str*) → Json [source]

+n

Collect following of the given user.

Strategy: click the username, get to the main page of the user, then click following, a list of following of the user shows in a popup. A request triggered by the clicking was sent to the path */api/v1/friendships/373735170/following/?count=12*.

+n

Parameters: **username** (*str*) – name of the user.
Returns: all visible followings' user information of the given user in json format.
Return type: Json

collect_hashtag_top_medias(*hashtag: str*) → Json [source]

pers

Collect top medias of a given hashtag.

Strategy: click the search button from the left navigation bar, and input the hashtag in the searchbar, and select the desired hashtag from searching results. This action will trigger a request sent to the path */api/v1/tags/web_info/?tag_name=<tag name>*, as result the top medias are showed.

Parameters: **hashtag** (*str*) – hashtag.
Returns: Hashtag information in a json format.
Return type: Json

+ pers (*boolean*) – Indicating whether results should be personalized or not

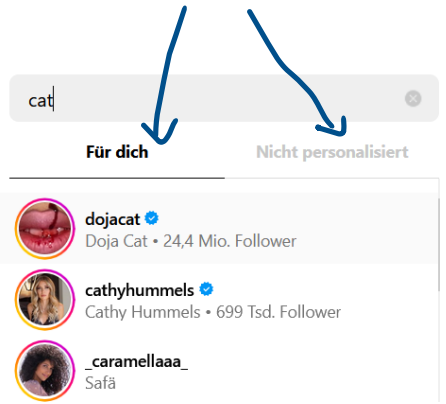
~~1) Rename „collect_followings_of_user“ to „collect_followings“~~ Done

~~5) Include a parameter „n“ which refers to the maximum number of followings (Like the n in the collect_comment function). Per default, n is set to 100~~ Done

~~1) Rename media with post~~ Done

2) When I open the search bar, I have two options, namely showing personalized or unpersonalized results. Which one are you using here?
Maybe we can include a parameter to „pers“ and if this parameter is true, we will show personalized results, else unpersonalized

This function collect_hashtag_top_posts collects the displayed posts under a hashtag. It doesn't do the searching.
What you described here is exactly this function search_with_keyword does, it search the given keyword in the search bar and return the search results. I added this parameter "per" to the function search_with_keyword.



`collect_likers_of_media(media_code: str) → Json` [source]

Collect the users, who likes a given media.

Strategy: click the likes (a button with link */p/<media_code>/liked_by/*), a list of likers shows in a popup. A request triggered by the clicking was sent to the path */api/v1/media/<media_id>/likers/*

Parameters: media_code (*str*) – media short code for generating access url.
Returns: all likers' user information of the given media in json format.
Return type: Json

1) ~~Replace the word „media" with „post"~~ Done
2) ~~Replace media_code with post_id~~ Done
3) ~~Include a parameter „n" which refers to the maximum number of likers (Like the n in the collect_comment function). Per default, n is set to 100~~ Done

`collect_medias_of_user(username: str) → Json` [source]

Collect medias of the given user.

Strategy: click the username, get to the main page of the user, a few medias are displayed in the user's main page. The request was sent to the path */api/v1/feed/user/<user_name>/username /?count=6* to get the medias and a next_max_id for loading other medias. To load the other medias, move the mouse to the bottom of the page, it will trigger another request sent to */api/v1/feed/user/<user_id>/?count=12&max_id=<next_max_id>* to have more medias.

Parameters: **username** (*str*) – name of the user.
Returns: all visible followers' user information of the given user in json format.
Return type: Json

Done

1) ~~Replace the word „media" with „post"~~
2) ~~Include a parameter „n" which refers to the maximum number of posts (Like the n in the collect_comment function). Per default, n is set to 100~~ Done

`collect_user_info(username: str) → Json` [source]

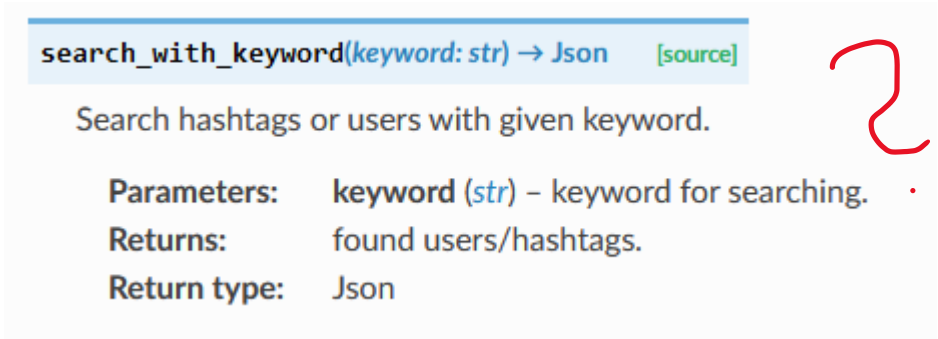
Collect user information through username, including user_id, username, profile_pic_url, biography, media_count, follower_count, following_count, medias.

Strategy: click the username, get to the main page of the user, the user information is returned in the response body.

Alternatively, put the mouse over the user's name in home page, the user information will show in a popup. A request triggered by the mouseover was sent to the path */api/v1/users/<user_id> /info/* or */api/v1/users/web_profile_info/?username=<user_name>*

Parameters: **username** (*str*) – name of the user.
Returns: user information in json format.
Return type: Json

1) ~~Replace the word „media" with „post"~~ Done
2) ~~By medias, you mean that all posts of a user will be collected with this function? I dont think that we need to collect all posts here, because we already have the collect_posts_of_user function~~ Done



How do you do this?
Whats the difference to the collect_hashtags_top_medias function?

I described this above.

Missing functions

It would be good to have in addition the following functions:

- Collect_reels_of_user (user_id) Done
- Collect_user_tagged_posts (user_id): Collect all posts in which user was tagged Done
- Collect_posts_by_music_id (music_id): Search for all posts containing a music_id Done
- get_friendship_status (user1, user2) : get the relationship between any two users (dont know whether this is possible) Done
- Download_media (media_url): Download the image/video based on a media_url or a profile_picture_url Done

The function get_friendship_status can only get the information about whether userA is in userB's following/follower list. The complete friendship status can be extracted only during collecting comments, because this part information is provided by instagram while sending comments of a post in response.

Parameters

pydantic model **UserInfo** [source]

User information contains *id*, *username*, *full name*, *biography* and son on .

▸ Show JSON schema

Fields:

- `biography (str)`
- `follower_count (int)`
- `following_count (int)`
- `following_tag_count (int)`
- `fullname (str)`
- `id (str | None)`
- `is_private (bool | None)`
- `is_verified (bool)`
- `media_count (int)`
- `profile_pic_url (str)`
- `username (str)`
- `usertags_count (int)`

Replace „media“ with „post“

Done

field **biography**: *str* *[Required]*

A shot description of the user account.

~~short~~

field **follower_count**: *int* *[Required]*

Number of the followers.

field **following_count**: *int* *[Required]*

Number of the following.

field **following_tag_count**: *int* *[Required]*

Number of the tags, which followed by the user.

~~which are followed~~ Done

field **fullname**: *str* *[Required]*

It's limited to 30 characters and must contain only letters, numbers, periods, underscores and spaces.

field **id**: *str* | *None* = *None*

unique identifier if an user. It's 9 digits in string format.

field **is_private**: *bool* | *None* = *None*

Indicates whether the user account is private or public.

field **is_verified**: *bool* = *False*

Indicates whether the user account is verified as business/official account or not.

field **media_count**: *int* *[Required]*

Number of the medias of the user.

field **profile_pic_url**: *str* *[Required]*

Url of the profile picture for downloading. The generated link is usually available only for couple hours.

field **username**: *str* *[Required]*

It's limited to 30 characters and must contain only letters in lowercase, numbers, periods, and underscores. ~~Is this unique?~~ It's unique and I enhanced its documentation.

field **usertags_count**: *int* *[Required]*

Number of the times, that the user was tagged in medias.

pydantic model **Users** [source]

An aggregation class to have the field *users* for storing a list of instances of *UserInfo*.

▸ Show JSON schema

Fields:

- `users (List[crawlinsta.schemas.UserInfo])`

field **users**: *List[UserInfo]* *[Required]*

A list of user information.

pydantic model **FriendshipStatus** [source]

Describe the relationship between the liker and the media owner.

▸ Show JSON schema

Fields:

- `blocking (bool)`
- `followed_by (bool)`
- `following (bool)`
- `incoming_request (bool)`
- `is_bestie (bool)`
- `is_blocking_reel (bool)`
- `is_eligible_to_subscribe (bool)`
- `is_feed_favorite (bool)`
- `is_muting_reel (bool)`
- `is_private (bool)`
- `is_restricted (bool)`
- `muting (bool)`
- `outgoing_request (bool)`
- `subscribed (bool)`

field **blocking**: *bool* *[Required]*

Is the liker blocked by the media owner? Blocked users' likes are not showed.

field **followed_by**: *bool* *[Required]*

Does the media owner follow the liker?

field **following**: *bool* *[Required]*

Does the liker follow the media owner?

field **incoming_request**: *bool* *[Required]*

Does the media owner send a follower request to the liker?

~~Has …. sent~~

field **is_bestie**: *bool* *[Required]*

Is the liker bestie of the media owner?

~~Add short explanation how people can get besties on Instagram~~ Done

field **is_blocking_reel**: *bool* *[Required]*

Is the liker blocking reel?

I don't understand what this means
Is the liker blocking reel to show in his personal feed?
I enhance it in the documentation

field **is_eligible_to_subscribe**: *bool* *[Required]*

Is the liker eligible to subscribe the media owner?

~~Add short explanation how people can get eligible to subscribe~~ Done

field **is_feed_favorite**: *bool* *[Required]*

Is the feed of the media owner favorite of the liker?

field **is_muting_reel**: *bool* *[Required]*

Is the liker muting reel?

field **is_private**: *bool* *[Required]*

is a private friend. For a private friend, the liker are not displayed in likes

~~Add short explanation how people can get private friends. Does this mean, the likers profile is set to private?~~ Done

field **is_restricted**: *bool* *[Required]*

field **muting**: *bool* *[Required]*

~~Explanation missing~~ Done

Is the media owner muted by the liker?

field **outgoing_request**: *bool* *[Required]*

Does the liker send a follower request to the media owner?

Has …. sent Done

field **subscribed**: *bool* *[Required]*

Does the liker subscribe the media owner?

Has …. subscribed Done

pydantic model **Liker** [source]

Liker, who likes a media. It contains the basic user information of *id*, *username*, *full name* etc, and additionally a *friendship_status* field to describe the relationship between the liker and the media owner.

▸ Show JSON schema

Fields:

- `friendship_status (crawlinsta.schemas.FriendshipStatus)`
- `fullname (str)`
- `id (str | None)`
- `is_private (bool | None)`
- `is_verified (bool)`
- `profile_pic_url (str)`
- `username (str)`

field **friendship_status**: *FriendshipStatus* *[Required]*

~~Explanation missing. Add something like „See FriendshipStatus“~~ Done

field **fullname**: *str* *[Required]*

It's limited to 30 characters and must contain only letters, numbers, periods, underscores and spaces.

field **id**: *str | None* = *None*

unique identifier if an user. It's 9 digits in string format.

field **is_private**: *bool | None* = *None*

Indicates whether the user account is private or public.

field **is_verified**: *bool* = *False*

Indicates whether the user account is verified as business/official account or not.

field **profile_pic_url**: *str* *[Required]*

Url of the profile picture for downloading. The generated link is usually available only for couple hours.

field **username**: *str* *[Required]*

It's limited to 30 characters and must contain only letters in lowercase, numbers, periods, and underscores.

pydantic model **Likers** [source]

An aggregation class to have the field *likers* for storing a list of instances of *Liker*.

▸ Show JSON schema

Fields:

- `likers (List[crawlinsta.schemas.Liker])`

field **likers**: *List[Liker]* *[Required]*

~~Explanation missing. Add something like „A list of Liker“~~ Done

pydantic model **Comment** [source]

Relevant information about a comment, including *id*, *user_id*, *media_id type* etc.

▸ Show JSON schema

Fields:

- `child_comments (List[crawlinsta.schemas.Comment])`
- `comment_like_count (int)`
- `created_at_utc (int)`
- `has_translation (bool)`
- `id (str)`
- `is_liked_by_media_owner (bool)`
- `is_ranked_comment (bool)`
- `media_id (str)`
- `share_enabled (bool)`
- `status (str)`
- `text (str)`
- `user_id (str)`

field **child_comments**: *List[Comment]* *[Required]*

Comments for this comment.

field **comment_like_count**: *int* *[Required]*

Number of people liked the comment.

who

field **created_at_utc**: *int* *[Required]*

Timestamp of when the comment was made.

field **has_translation**: *bool* *[Required]*

Does the comment have a translation

How does this work? Does the posting user or instagram generate the translation? If it has a translation, do we return the original or the translated text in the text field?

Added more information in documentation.

field **id**: *str* *[Required]*

Unique identifier of the comment. It consists of 17 digits.

field **is_liked_by_media_owner**: *bool* *[Required]*

Is the comment liked by the media owner.

field **is_ranked_comment**: *bool* *[Required]*

Is the comment a ranked one?

~~Add short explanation how a comment get ranked , e.g. By the user or by instagram? Explain which are the consequences of a comment being ranked (will it be shown on the main website for many users?)~~ Done

field **media_id**: *str* *[Required]*

Id reference to the user, who made the comment.

field **share_enabled**: *bool* *[Required]*

Is the comment enabled for sharing.

~~Add short explanation how a comment gets enabled for sharing, e.g. By the user or by instagram?~~

Done

field **status**: *str* *[Required]*

Status of the comment, Active or Inactive.

~~Add short explanation how a comment gets active/inactive~~

Done

field **text**: *str* *[Required]*

Comment content in free text format.

Is there any more details, which we should know (e.g. max number of characters, encoding, does it include mentions, emojis, hashtags,…)

Added more information in documentation.

field **user_id**: *str* *[Required]*

Id reference to the user, who made the comment.

pydantic model **Comments** [source]

An aggregation of comments.

▸ Show JSON schema

Fields:

- `comments (List[crawlinsta.schemas.Comment])`

field **comments**: *List[Comment]* *[Required]*

A list of comments.

pydantic model **Usertag** [source]

Usertag information, mainly about who is tagged at which position at what time in video and how long the tagged user appears.

▸ Show JSON schema

Fields:

- `duration_in_video_in_sec (float | None)`
- `fullname (str)`
- `id (str | None)`
- `is_private (bool | None)`
- `is_verified (bool)`
- `position (List[float])`
- `profile_pic_url (str)`
- `start_time_in_video_in_sec (float | None)`
- `user (crawlinsta.schemas.UserBasicInfo)`
- `username (str)`

field **duration_in_video_in_sec**: *float | None = None*

Duration in the video in seconds, when the tagged user shows up

field **fullname**: *str [Required]*

It's limited to 30 characters and must contain only letters, numbers, periods, underscores and spaces.

field **id**: *str | None = None*

unique identifier if an user. It's 9 digits in string format.

field **is_private**: *bool | None = None*

Indicates whether the user account is private or public.

field **is_verified**: *bool = False*

Indicates whether the user account is verified as business/official account or not.

field **position**: *List[float] [Required]*

A list of two floats, which is used to identify the position of the tagged user in the media.

field **profile_pic_url**: *str [Required]*

Url of the profile picture for downloading. The generated link is usually available only for couple hours.

field **start_time_in_video_in_sec**: *float | None = None*

Start time in video in seconds when the tagged user shows up

field **user**: *UserBasicInfo [Required]*

User who is tagged in the media.

field **username**: *str [Required]*

It's limited to 30 characters and must contain only letters in lowercase, numbers, periods, and underscores.

pydantic model **Location** [source]

Location information, contains location name, city, longitude, latitude and address etc.

▸ Show JSON schema

Fields:

- address (str)
- city (str)
- id (str)
- lat (float)
- lng (float)
- name (str)
- short_name (str)

Specify where the location is coming from.
Is it only given by the user or does Instagram automatically attach the location if GPS is enabled?

added more information in the documentation

field **address**: *str* *[Required]*

Address of the location. Typically is empty.

field **city**: *str* *[Required]*

to which city the location belongs

field **id**: *str* *[Required]*

Unique identifier of the location.

field **lat**: *float* *[Required]*

Latitude of the location.

field **lng**: *float* *[Required]*

Longitude of the location.

field **name**: *str* *[Required]*

Full name of the location.

field **short_name**: *str* *[Required]*

Short name of the location

pydantic model **Caption** [source]

Caption of the media.

▸ Show JSON schema

Fields:

- content_type (str)
- created_at_utc (int)
- id (str)
- text (str)

field **content_type**: *str* *[Required]*

Content type of the Caption.

Add short explanation which content types do we have here

I removed it, since all the time the content_type is “comment”

field **created_at_utc**: *int* *[Required]*

Timestamp when the caption was created. In unix epoch time.

field **id**: *str* *[Required]*

Unique identifier of the caption.

field **text**: *str* *[Required]*

Text content of the caption.

See Comment Text

added more information in the documentation

pydantic model `Media` [source]

Media information about when it was created, who is tagged in it, its caption, location width, height etc.

▸ Show JSON schema

Fields:

- `accessibility_caption (str)`
- `caption (crawlinsta.schemas.Caption)`
- `code (str)`
- `comment_count (int)`
- `has_shared_to_fb (bool)`
- `id (str)`
- `like_count (int)`
- `location (crawlinsta.schemas.Location | None)`
- `media_type (str)`
- `original_height (int)`
- `original_width (int)`
- `taken_at (int)`
- `url (str)`
- `usertags (List[crawlinsta.schemas.Usertag])`

Replace Media with post

done

field **accessibility_caption**: *str* *[Required]*

Accessibility caption in text format.

Add short explanation, what is an Accessibility_caption?

Done

field **caption**: *Caption* *[Required]*

Caption of the port. post

field **code**: *str* *[Required]*

Short code of the media url. Can be used in form of www.instagram.com/<code> to access the media.

field **comment_count**: *int* *[Required]*

Count of comments.

field **has_shared_to_fb**: *bool* *[Required]*

Is the media shared to facebook?

field **id**: *str* *[Required]*

Unique identifier of the media.

field **like_count**: *int* *[Required]*

Count of likes.

field **location**: *Location* | *None* *[Required]*

Location of the media, most of the time it's not given.

field **media_type**: *str* *[Required]*

media type: Photo, Video, IGTV, Reel, Album.

field **original_height**: *int* *[Required]*

Original height of the media, can be used to get the downloading url of the media.

How?

field **original_width**: *int* *[Required]*

Original width of the media, can be used to get the downloading url of the media.

field **taken_at**: *int* *[Required]*

When the media was created, in unix epoch time.

field **url**: *str* *[Required]*

Download URL of the media, which is only accessible in fea hours.

Please find out for how long exactly

The url is at least available in 24 hours, I'm still testing it.

field **usertags**: *List[Usertag]* *[Required]*

Usertags appear in the media.

Here the word media
Is ok, because it refers
to the media (image, video,...)
within the post

pydantic model **Hashtag** [source]

Hashtag information

▸ Show JSON schema

Fields:

- `id ()`
- `is_trending (bool)`
- `media_count ()`
- `medias ()`
- `name ()`
- `profile_pic_url (str)`
- `related_tags (List[str] | None)`
- `social_context (str)`
- `social_context_profile_links (List[str])`
- `subtitle (str)`

field **id**: *str* *[Required]*

Unique identifier of the hashtag.

field **is_trending**: *bool* *[Required]*

Is it a trending hashtag?

field **media_count**: *int* *[Required]*

Count of the medias with the hashtag.

field **medias**: *List[Media]* *[Required]*

A list of medias.

Are these all posts to the given hashtag?
Or does instagram limit here the number of posts?

It will only show a limited number of posts here, less than 30 posts. added it in docu.

field **name**: *str* *[Required]*

name of the hashtag.

field **profile_pic_url**: *str* *[Required]*

Profile picture url.

Of which profile? The profile who has used the hashtag?

Added more information about this field in docu

field **related_tags**: *List[str] | None* = *None*

related tags in a list.

Add a short explanation, how do tags get related? Does Instagram calulcate this?
How? Based on Coocurrence of Posts or likes or…?

done

field **social_context**: *str* = ''

Social context of the hashtag.

Unclear what social context means. Please add some explanation

field **social_context_profile_links**: *List[str]* = *[]*

Social context profile links.

I removed them, since I couldn't find any useful information about these two fields in instagram

field **subtitle**: *str* = ''

subtitle of the hastag.

pydantic model **SearchingResult** [source]

Searching result contains found hashtags and/or users.

▸ Show JSON schema

Fields:

- `hashtags (List[crawlinsta.schemas.SearchingResultHashtag])`
- `users (List[crawlinsta.schemas.SearchingResultUser])`

field **hashtags**: *List[SearchingResultHashtag]* = *[]*

Found hashtags matched to the keywords.

field **users**: *List[SearchingResultUser]* = *[]*

Found users matched to the keywords.

Some more questions

- Can we get the user mention objects and hashtag objects of a post or (caption)?

  Unfortunately the mentions/hashtags in captions are not captured directly.
  But we could get them with the help of some processing functions.

- Do you have mentions/hashtags in comments captions? If yes, can we collect these as well?

  Unfortunately the mentions/hashtags in commentsare not captured directly.
  But we can have the option to collect them through
  collecting all the commentsthen process them to find out all mentions/hashtags.

- Comment structure: Do I understand this correctly: If we have a comment A which answers to comment B which answers to comment C, then the structure would look like this?

  C: {C_id, C_text, [B: {B_id, B_text, [A: A_id, A_text,…],…}] ,…}

  Yes, it's correct