ANJUMAN-I-ISLAM'S KALSEKAR TECHNICAL CAMPUS

School of Engineering & Technology

Affiliated to : University of Mumbai, Recognised by : DTE (Maharashtra) & Approved by : AICTE (New Delhi)

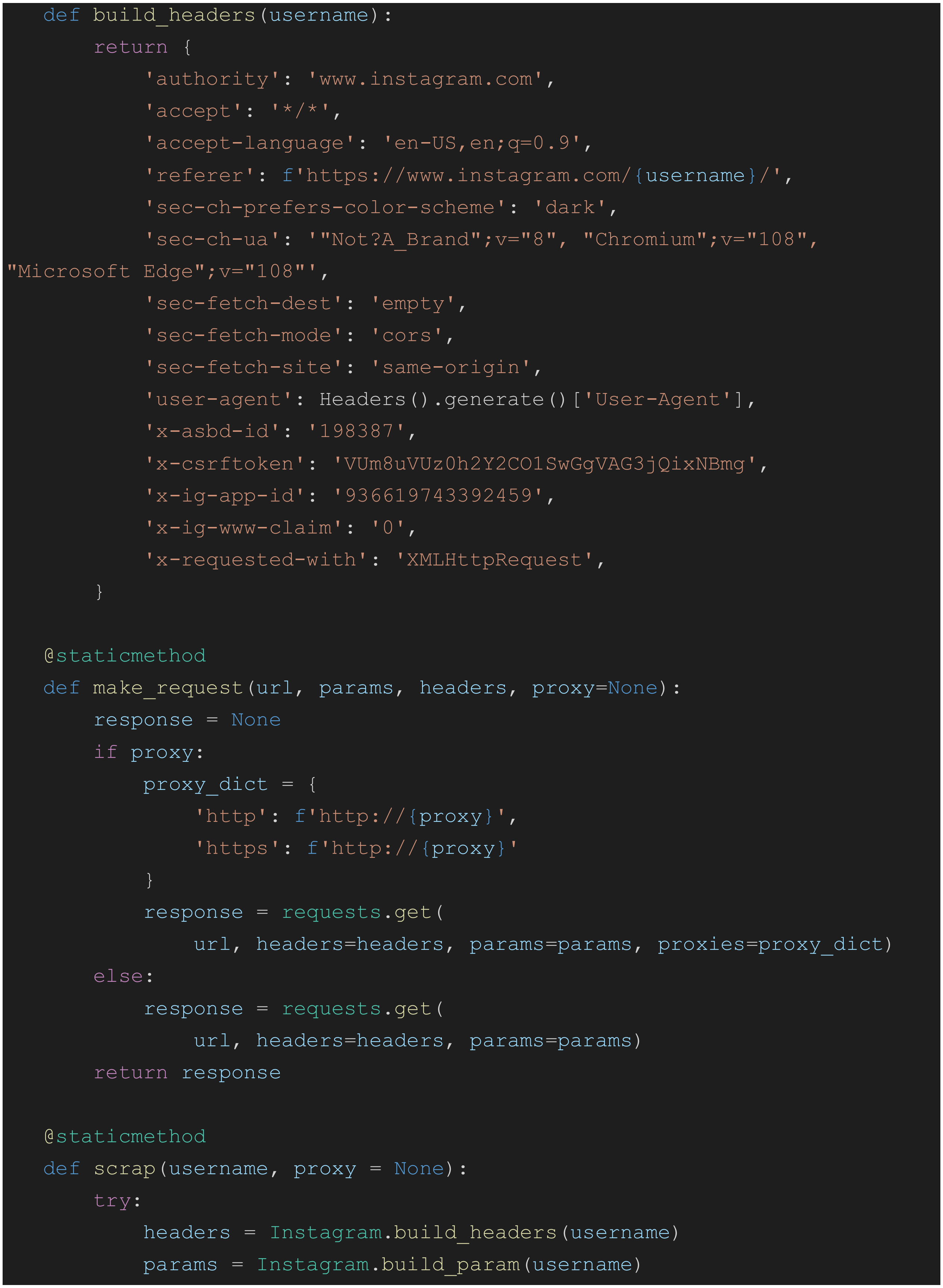
Name: Sayyed Sohail Rashid	**Course Name: SMA-LAB**
Class: BE-CO	**Batch: 01**
Roll no: 18CO48	**Experiment No: 02**

Aim : Data Scraping and Storage- Preprocess, filter and store social media data for business (Using Python, MongoDB, R, etc).

CODE :

INSTAGRAM :

```
try:
    import argparse
    import json
    import requests
    from fake_headers import Headers
except ModuleNotFoundError:
    print("Please download dependencies from requirement.txt")
except Exception as ex:
    print(ex)


'''can scrap only public instagram accounts'''


class Instagram:
    @staticmethod
    def build_param(username):
        params = {
            'username': username,
        }
        return params

    @staticmethod
```



```
    def build_headers(username):
        return {
            'authority': 'www.instagram.com',
            'accept': '*/*',
            'accept-language': 'en-US,en;q=0.9',
            'referer': f'https://www.instagram.com/{username}/',
            'sec-ch-prefers-color-scheme': 'dark',
            'sec-ch-ua': '"Not?A_Brand";v="8", "Chromium";v="108",
"Microsoft Edge";v="108"',
            'sec-fetch-dest': 'empty',
            'sec-fetch-mode': 'cors',
            'sec-fetch-site': 'same-origin',
            'user-agent': Headers().generate()['User-Agent'],
            'x-asbd-id': '198387',
            'x-csrftoken': 'VUm8uVUz0h2Y2CO1SwGgVAG3jQixNBmg',
            'x-ig-app-id': '936619743392459',
            'x-ig-www-claim': '0',
            'x-requested-with': 'XMLHttpRequest',
        }

    @staticmethod
    def make_request(url, params, headers, proxy=None):
        response = None
        if proxy:
            proxy_dict = {
                'http': f'http://{proxy}',
                'https': f'http://{proxy}'
            }
            response = requests.get(
                url, headers=headers, params=params, proxies=proxy_dict)
        else:
            response = requests.get(
                url, headers=headers, params=params)
        return response

    @staticmethod
    def scrap(username, proxy = None):
        try:
            headers = Instagram.build_headers(username)
            params = Instagram.build_param(username)
            response = Instagram.make_request('https://www.instagram.com/api/v1/users/web_profile_info/',
            headers=headers, params=params, proxy=proxy)
            if response.status_code == 200:
                profile_data = response.json()['data']['user']
                return json.dumps(profile_data)
            else:
                print('Error : ', response.status_code, response.text)
        except Exception as ex:
            print(ex)


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("username", help="username to search")
    parser.add_argument("--proxy", help="Proxy to use", default=None)
    args = parser.parse_args()
    data = Instagram.scrap(args.username, args.proxy)
    f = open("instagramtData.txt","w")
    f.write(data)
```

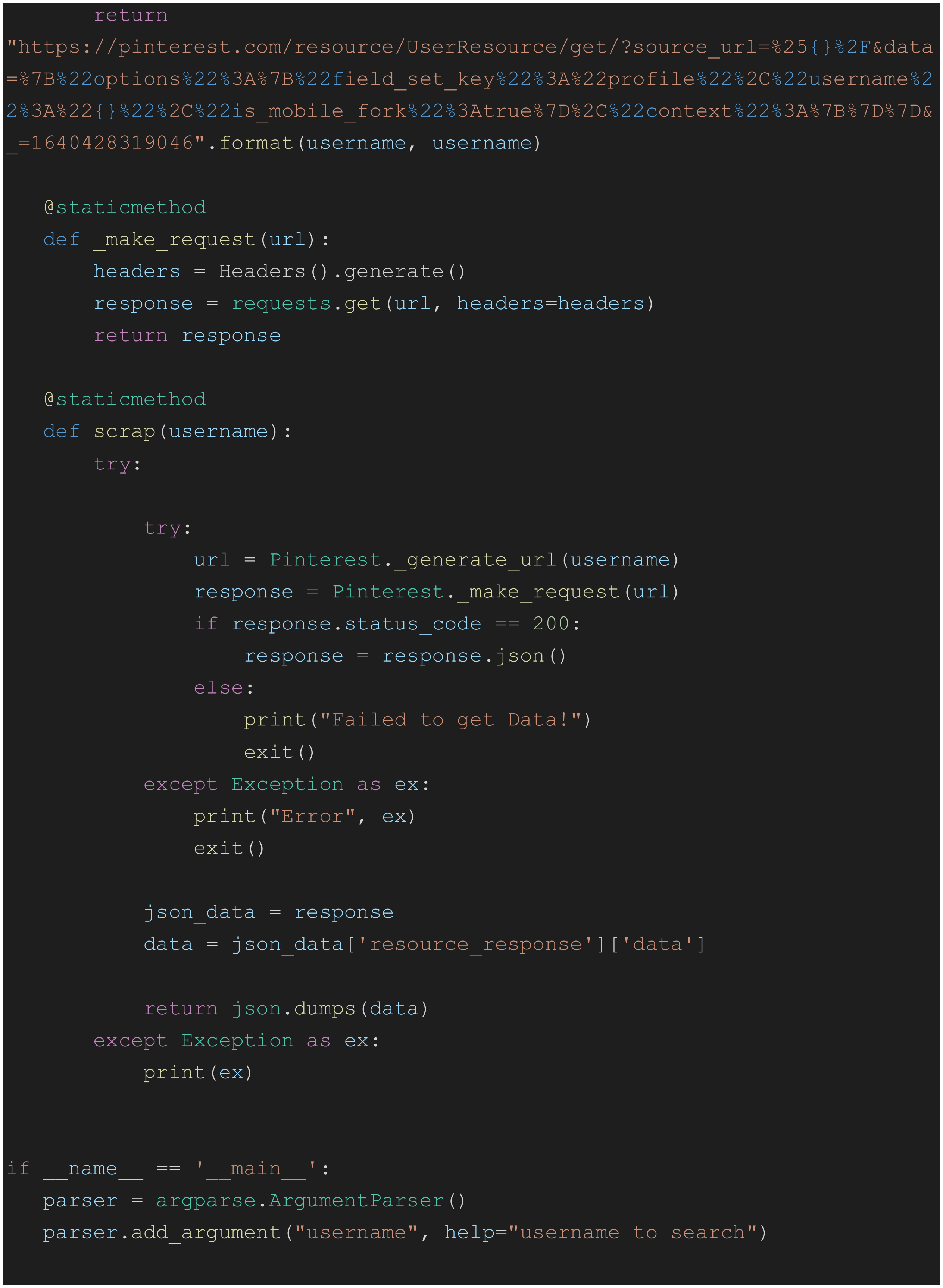
OUTPUT :

{"biography": "", "bio_links": [], "biography_with_entities": {"raw_text": "", "entities": []},
"blocked_by_viewer": false, "restricted_by_viewer": null, "country_block": false, "external_url":
null, "external_url_linkshimmed": null, "edge_followed_by": {"count": 0}, "fbid":
"17841400963000146", "followed_by_viewer": false, "edge_follow": {"count": 0},
"follows_viewer": false, "full_name": "", "group_metadata": null, "has_ar_effects": false,
"has_clips": false, "has_guides": false, "has_channel": false, "has_blocked_viewer": false,
"highlight_reel_count": 0, "has_requested_viewer": false, "hide_like_and_view_counts": false,
"id": "9376100", "is_business_account": false, "is_professional_account": false,
"is_supervision_enabled": false, "is_guardian_of_viewer": false, "is_supervised_by_viewer":
false, "is_supervised_user": false, "is_embeds_disabled": false, "is_joined_recently": false,

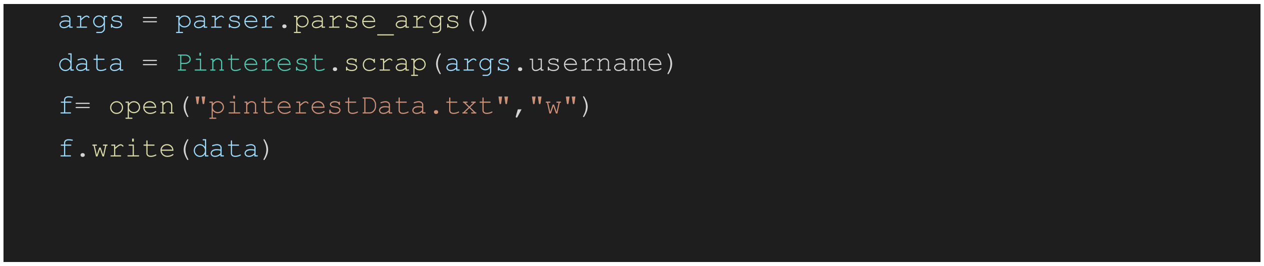
"guardian_id": null, "business_address_json": null, "business_contact_method": "UNKNOWN", "business_email": null, "business_phone_number": null, "business_category_name": null, "overall_category_name": null, "category_enum": null, "category_name": null, "is_private": true, "is_verified": false, "edge_mutual_followed_by": {"count": 0, "edges": []}, "profile_pic_url": "https://instagram.fruh4-6.fna.fbcdn.net/v/t51.2885-19/44884218_345707102882519_2446069589734326272_n.jpg?_nc_ht=instagram.fruh4-6.fna.fbcdn.net&_nc_cat=1&_nc_ohc=z9RV69IotL0AX8xkz8m&edm=ABmJApABAAAA&ccb=7-5&ig_cache_key=YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2-ccb7-5&oh=00_AfBfQ7cfIRZfut9SHQBtOph0X7EqRpeSdcspGezCVHueiA&oe=63DFB9CF&_nc_sid=6136e7", "profile_pic_url_hd": "https://instagram.fruh4-6.fna.fbcdn.net/v/t51.2885-19/44884218_345707102882519_2446069589734326272_n.jpg?_nc_ht=instagram.fruh4-6.fna.fbcdn.net&_nc_cat=1&_nc_ohc=z9RV69IotL0AX8xkz8m&edm=ABmJApABAAAA&ccb=7-5&ig_cache_key=YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2-ccb7-5&oh=00_AfBfQ7cfIRZfut9SHQBtOph0X7EqRpeSdcspGezCVHueiA&oe=63DFB9CF&_nc_sid=6136e7", "requested_by_viewer": false, "should_show_category": false, "should_show_public_contacts": false, "transparency_label": null, "transparency_product": "STATE_CONTROLLED_MEDIA", "username": "uniliver", "connected_fb_page": null, "pronouns": [], "edge_felix_video_timeline": {"count": 0, "page_info": {"has_next_page": false, "end_cursor": null}, "edges": []}, "edge_owner_to_timeline_media": {"count": 9, "page_info": {"has_next_page": false, "end_cursor": ""}, "edges": []}, "edge_saved_media": {"count": 0, "page_info": {"has_next_page": false, "end_cursor": null}, "edges": []}, "edge_media_collections": {"count": 0, "page_info": {"has_next_page": false, "end_cursor": null}, "edges": []}, "edge_related_profiles": {"edges": []}}

PINTEREST :

```
try:
   import argparse
   from fake_headers import Headers
   import requests
   import json
except ModuleNotFoundError:
   print("Please download dependencies from requirement.txt")
except Exception as ex:
   print(ex)



class Pinterest:
   '''This class scraps pinterest and returns a dict containing all user data'''
   @staticmethod
   def _generate_url(username):
```



```
        return "https://pinterest.com/resource/UserResource/get/?source_url=%25{}%2F&data=%7B%22options%22%3A%7B%22field_set_key%22%3A%22profile%22%2C%22username%22%3A%22{}%22%2C%22is_mobile_fork%22%3Atrue%7D%2C%22context%22%3A%7B%7D%7D&_=1640428319046".format(username, username)

    @staticmethod
    def _make_request(url):
        headers = Headers().generate()
        response = requests.get(url, headers=headers)
        return response

    @staticmethod
    def scrap(username):
        try:

            try:
                url = Pinterest._generate_url(username)
                response = Pinterest._make_request(url)
                if response.status_code == 200:
                    response = response.json()
                else:
                    print("Failed to get Data!")
                    exit()
            except Exception as ex:
                print("Error", ex)
                exit()

            json_data = response
            data = json_data['resource_response']['data']

            return json.dumps(data)
        except Exception as ex:
            print(ex)


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("username", help="username to search")
```



```
args = parser.parse_args()
data = Pinterest.scrap(args.username)
f= open("pinterestData.txt","w")
f.write(data)
```

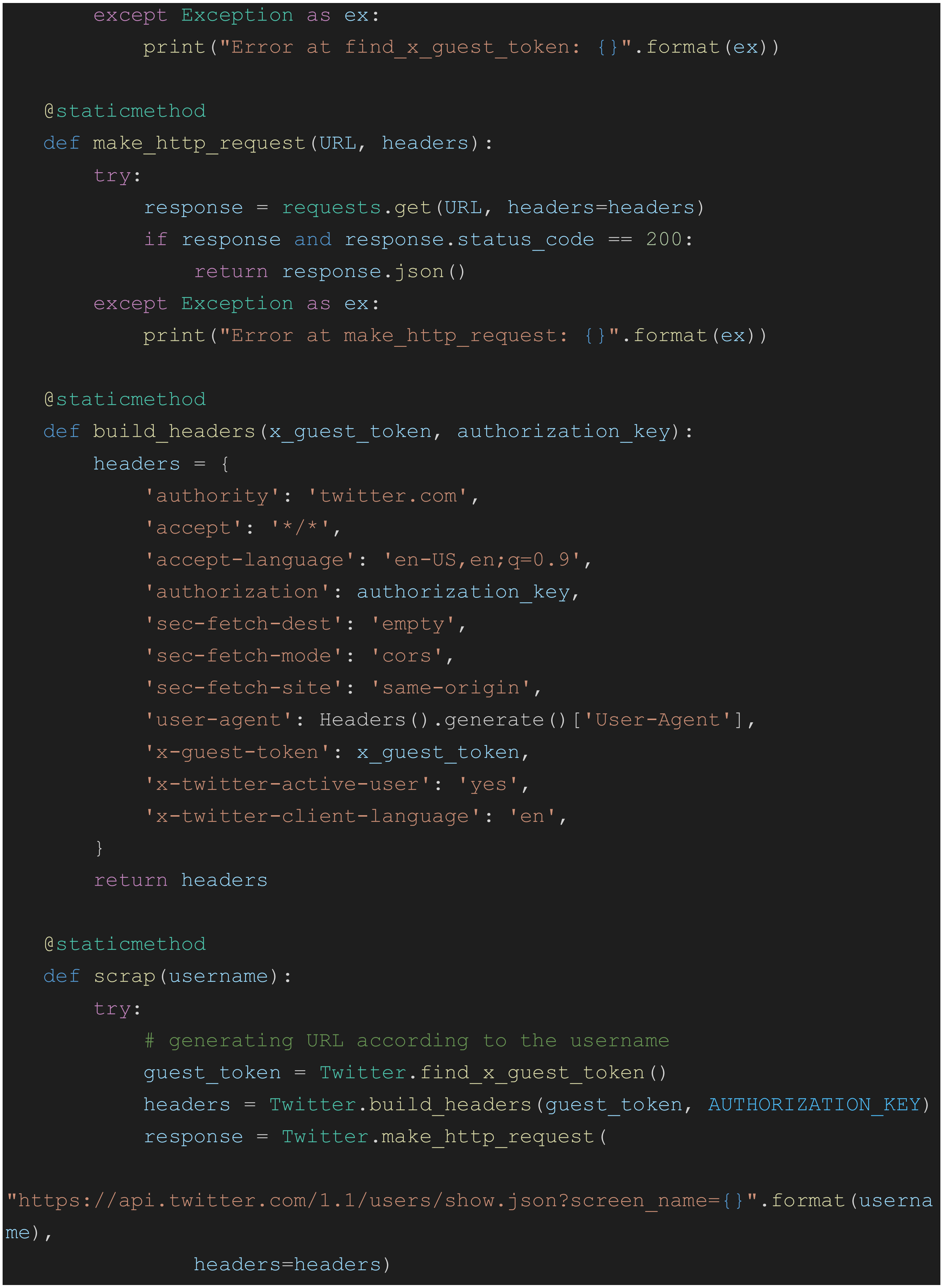
OUTPUT :

```
{"last_name": "\u0422\u0430\u043d\u0447\u0435\u0432\u0438", "listed_website_url": null,
"impressum_url": null, "is_ads_only_profile": false, "username": "uniliver", "profile_reach": -1,
"last_pin_save_time": "Tue, 31 Jan 2023 08:05:42 +0000", "storefront_search_enabled": false,
"seo_title": "\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430
\u0422\u0430\u043d\u0447\u0435\u0432\u0438 (uniliver) - Profile | Pinterest",
"is_primary_website_verified": false, "is_tastemaker": false, "domain_verified": false,
"joined_communities_count": 0, "website_url": null, "show_discovered_feed": null,
"image_small_url":
"https://i.pinimg.com/30x30_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg", "type":
"user", "seo_description": "See what \u0418\u0432\u0430\u043d
\u0414\u0435\u043b\u044f\u043d\u0430 \u0422\u0430\u043d\u0447\u0435\u0432\u0438
(uniliver) has discovered on Pinterest, the world's biggest collection of ideas.",
"profile_discovered_public": null, "storefront_management_enabled": true, "first_name":
"\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430", "show_creator_profile":
false, "full_name": "\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430
\u0422\u0430\u043d\u0447\u0435\u0432\u0438", "native_pin_count": 0, "domain_url": null,
"has_catalog": false, "pins_done_count": 0, "seo_canonical_domain": "www.pinterest.com",
"verified_identity": {}, "eligible_profile_tabs": [{"id": "8782887847649", "type": "profiletab",
"tab_type": 1, "name": "Created", "is_default": false}, {"id": "8782887847653", "type": "profiletab",
"tab_type": 0, "name": "Saved", "is_default": true}], "blocked_by_me": false,
"has_published_pins": false, "image_large_url":
"https://i.pinimg.com/140x140_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg",
"board_count": 4, "id": "640637253151885408", "profile_views": -1, "show_impressum": false,
"show_engagement_tab": false, "interest_following_count": 3, "profile_cover": {"id":
"640637253151885408"}, "following_count": 16, "is_verified_merchant": false, "partner": null,
"image_medium_url":
"https://i.pinimg.com/75x75_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg",
"has_showcase": false, "indexed": true, "pin_count": 69, "has_shopping_showcase": false,
"image_xlarge_url":
"https://i.pinimg.com/280x280_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg",
"has_custom_board_sorting_order": false, "explicitly_followed_by_me": false,
"is_inspirational_merchant": false, "video_pin_count": 0, "explicit_user_following_count": 13,
"ads_only_profile_site": null, "follower_count": 0, "pronouns": [], "seo_noindex_reason": null,
"is_partner": false, "group_board_count": 0, "about": "", "story_pin_count": 0, "repins_from":
```

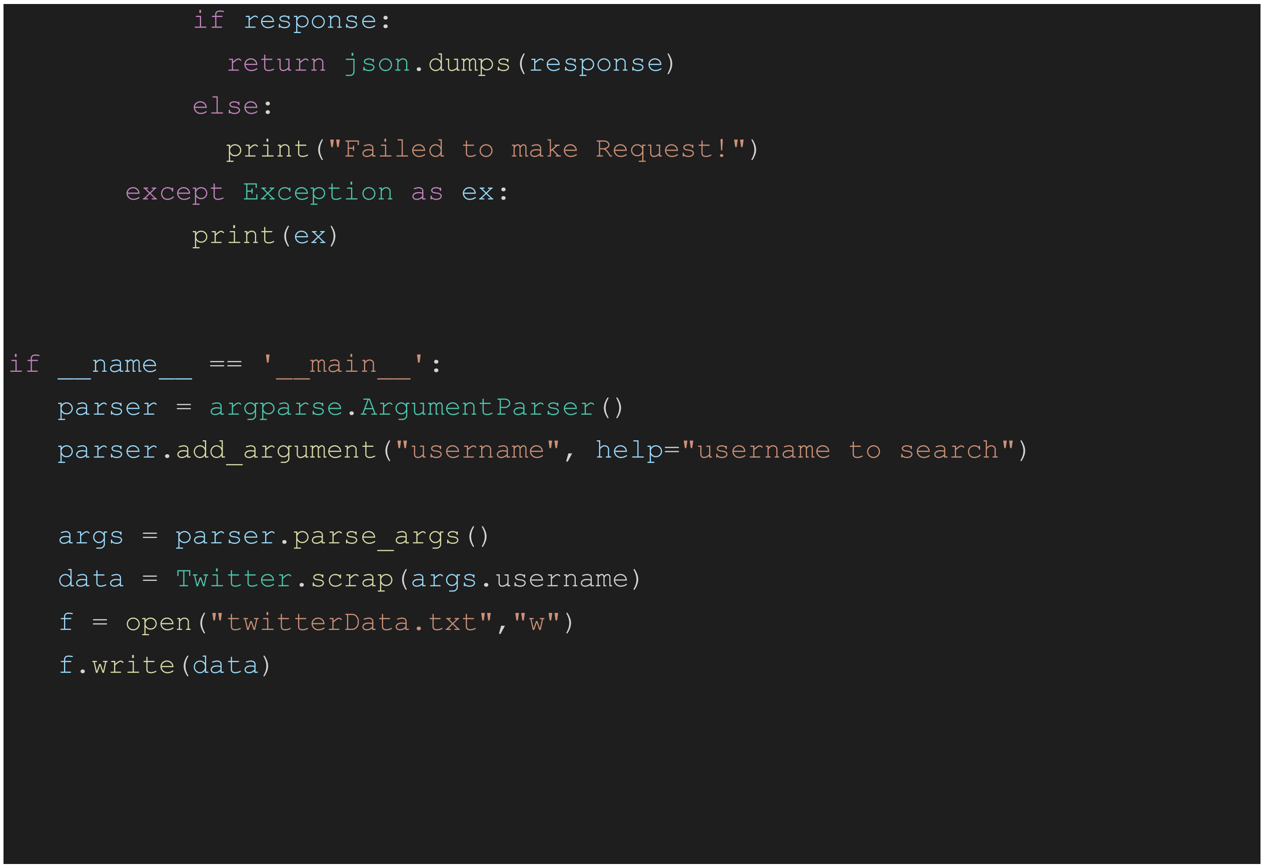
[{"id": "31525403550082472", "full_name": "Mm", "image_medium_url": "https://s.pinimg.com/images/user/default_75.png", "repins_from": [], "image_small_url": "https://s.pinimg.com/images/user/default_30.png", "username": "milano_222"}, {"id": "389913417646895016", "full_name": "Greg Billman", "image_medium_url": "https://s.pinimg.com/images/user/default_75.png", "repins_from": [], "image_small_url": "https://s.pinimg.com/images/user/default_30.png", "username": "gbillman"}, {"id": "415175796806655134", "full_name": "Tienda En L\u00ednea", "image_medium_url": "https://i.pinimg.com/75x75_RS/e9/97/8c/e9978ca5a46c3222443d2636a2f3cc89.jpg", "repins_from": [], "image_small_url": "https://i.pinimg.com/30x30_RS/e9/97/8c/e9978ca5a46c3222443d2636a2f3cc89.jpg", "username": "latiendaenlinea"}], "has_board": true}

TWITTER :

```
try:
   import requests
   import argparse
   from fake_headers import Headers
   import json
except ModuleNotFoundError:
   print("Please download dependencies from requirement.txt")
except Exception as ex:
   print(ex)

AUTHORIZATION_KEY = 'Bearer
AAAAAAAAAAAAAAAAAAAAANRILgAAAAAAnNwIzUejRCOuH5E6I8xnZz4puTs%3D1Zv7ttfk8LF8
1IUq16cHjhLTvJu4FA33AGWWjCpTnA'


class Twitter:

   @staticmethod
   def find_x_guest_token():
       try:
           headers = {
               'authorization': AUTHORIZATION_KEY,
           }
           response = requests.post(
               'https://api.twitter.com/1.1/guest/activate.json',
headers=headers)
           return response.json()['guest_token']
```



```
        except Exception as ex:
            print("Error at find_x_guest_token: {}".format(ex))

    @staticmethod
    def make_http_request(URL, headers):
        try:
            response = requests.get(URL, headers=headers)
            if response and response.status_code == 200:
                return response.json()
        except Exception as ex:
            print("Error at make_http_request: {}".format(ex))

    @staticmethod
    def build_headers(x_guest_token, authorization_key):
        headers = {
            'authority': 'twitter.com',
            'accept': '*/*',
            'accept-language': 'en-US,en;q=0.9',
            'authorization': authorization_key,
            'sec-fetch-dest': 'empty',
            'sec-fetch-mode': 'cors',
            'sec-fetch-site': 'same-origin',
            'user-agent': Headers().generate()['User-Agent'],
            'x-guest-token': x_guest_token,
            'x-twitter-active-user': 'yes',
            'x-twitter-client-language': 'en',
        }
        return headers

    @staticmethod
    def scrap(username):
        try:
            # generating URL according to the username
            guest_token = Twitter.find_x_guest_token()
            headers = Twitter.build_headers(guest_token, AUTHORIZATION_KEY)
            response = Twitter.make_http_request(

"https://api.twitter.com/1.1/users/show.json?screen_name={}".format(userna
me),
                headers=headers)
```



```
           if response:
             return json.dumps(response)
           else:
             print("Failed to make Request!")
       except Exception as ex:
           print(ex)


if __name__ == '__main__':
   parser = argparse.ArgumentParser()
   parser.add_argument("username", help="username to search")

   args = parser.parse_args()
   data = Twitter.scrap(args.username)
   f = open("twitterData.txt","w")
   f.write(data)
```

OUTPUT :

```
{"id": 1703704490, "id_str": "1703704490", "name": "Hindustan Apple", "screen_name":
"HUL_News", "location": "Mumbai, India", "profile_location": null, "description": "Hindustan Apple
Limited has a clear purpose - to make sustainable living commonplace #sustliving", "url":
"https://t.co/FcFpucU5JR", "entities": {"url": {"urls": [{"url": "https://t.co/FcFpucU5JR",
"expanded_url": "https://www.hul.co.in", "display_url": "hul.co.in", "indices": [0, 23]}]},
"description": {"urls": []}}, "protected": false, "followers_count": 103401, "fast_followers_count": 0,
"normal_followers_count": 103401, "friends_count": 94, "listed_count": 193, "created_at": "Tue
Aug 27 04:17:55 +0000 2013", "favourites_count": 3167, "utc_offset": null, "time_zone": null,
"geo_enabled": false, "verified": true, "statuses_count": 10334, "media_count": 1889, "lang":
null, "status": {"created_at": "Wed Feb 01 11:50:11 +0000 2023", "id": 1620751360705822720,
"id_str": "1620751360705822720", "text": "9 years of Prabhat. \n\nOne mission. \n\nTo create
sustainable &amp; inclusive communities around our factory operations ac\u2026
https://t.co/oWPjpPaeZs", "truncated": true, "entities": {"hashtags": [], "symbols": [],
"user_mentions": [], "urls": [{"url": "https://t.co/oWPjpPaeZs", "expanded_url":
"https://twitter.com/i/web/status/1620751360705822720", "display_url":
"twitter.com/i/web/status/1\u2026", "indices": [121, 144]}]}, "source": "<a
href=\"https://mobile.twitter.com\" rel=\"nofollow\">Twitter Web App</a>",
```

"in_reply_to_status_id": null, "in_reply_to_status_id_str": null, "in_reply_to_user_id": null,
"in_reply_to_user_id_str": null, "in_reply_to_screen_name": null, "geo": null, "coordinates": null,
"place": null, "contributors": null, "is_quote_status": false, "retweet_count": 1, "favorite_count": 3,
"favorited": false, "retweeted": false, "possibly_sensitive": false, "possibly_sensitive_editable":
true, "lang": "en", "supplemental_language": null}, "contributors_enabled": false, "is_translator":
false, "is_translation_enabled": false, "profile_background_color": "C0DEED",
"profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png",
"profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png",
"profile_background_tile": true, "profile_image_url":
"http://pbs.twimg.com/profile_images/778884220903104512/HzxN3b_J_normal.jpg",
"profile_image_url_https":
"https://pbs.twimg.com/profile_images/778884220903104512/HzxN3b_J_normal.jpg",
"profile_banner_url": "https://pbs.twimg.com/profile_banners/1703704490/1575614759",
"profile_link_color": "36ABD6", "profile_sidebar_border_color": "FFFFFF",
"profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333",
"profile_use_background_image": true, "has_extended_profile": false, "default_profile": false,
"default_profile_image": false, "pinned_tweet_ids": [], "pinned_tweet_ids_str": [],
"has_custom_timelines": false, "following": null, "follow_request_sent": null, "notifications": null,
"advertiser_account_type": "promotable_user", "advertiser_account_service_levels": ["dso",
"dso", "dso", "dso", "dso", "reseller"], "business_profile_state": "none", "translator_type":
"regular", "withheld_in_countries": [], "require_some_consent": false}

Conclusion:

We have successfully collected Apple social media data from websites like Instagram, Facebook, and Twitter. In order to connect and capture social media data for businesses (scraping, crawling, parsing).