Python Script 7: Scraping tweets using BeautifulSoup (/post/522/python-script-7-scraping-tweets-using-beautifulsoup/)

beautifulsoup (/tag/beautifulsoup/) scraping (/tag/scraping/) 0 6663

Twitter is one of the most popular social networking services used by most prominent people of world. Tweets can be used to perform sentimental analysis (https://scholar.google.co.in/scholar?q=tweet+uses+sentiment+analysis&hl=en&as_sdt=0&as_vis=1&oi=scholart&sa=X&ved=0ahUKEwjN47mhwvfYAhWJv48KHY9bBPIQgQMIJjAA).

In this article we will see how to scrape tweets using BeautifulSoup. We are not using Twitter API as most of the APIs have rate limits.

You can download all the pictures of any Instagram user (https://www.pythoncircle.com/post/447/py_instagram_dl-the-python-package-to-download-all-pictures-of-an-instagram-user/) in just few lines of codes. We converted the script into reusable python package (https://www.pythoncircle.com/post/368/how-to-develop-a-distributable-django-app-to-block-the-crawling-ip-addresses/) to make things easy.

Setup:

Create a virtual environment. If you are not in the habit of working with virtual environments, please stop immediately and read this article on virtual environments (https://www.pythoncircle.com/post/404/virtual-environment-in-python-a-pocket-guide/) first.
Once virtual environment is created and activated, install the dependencies in it.

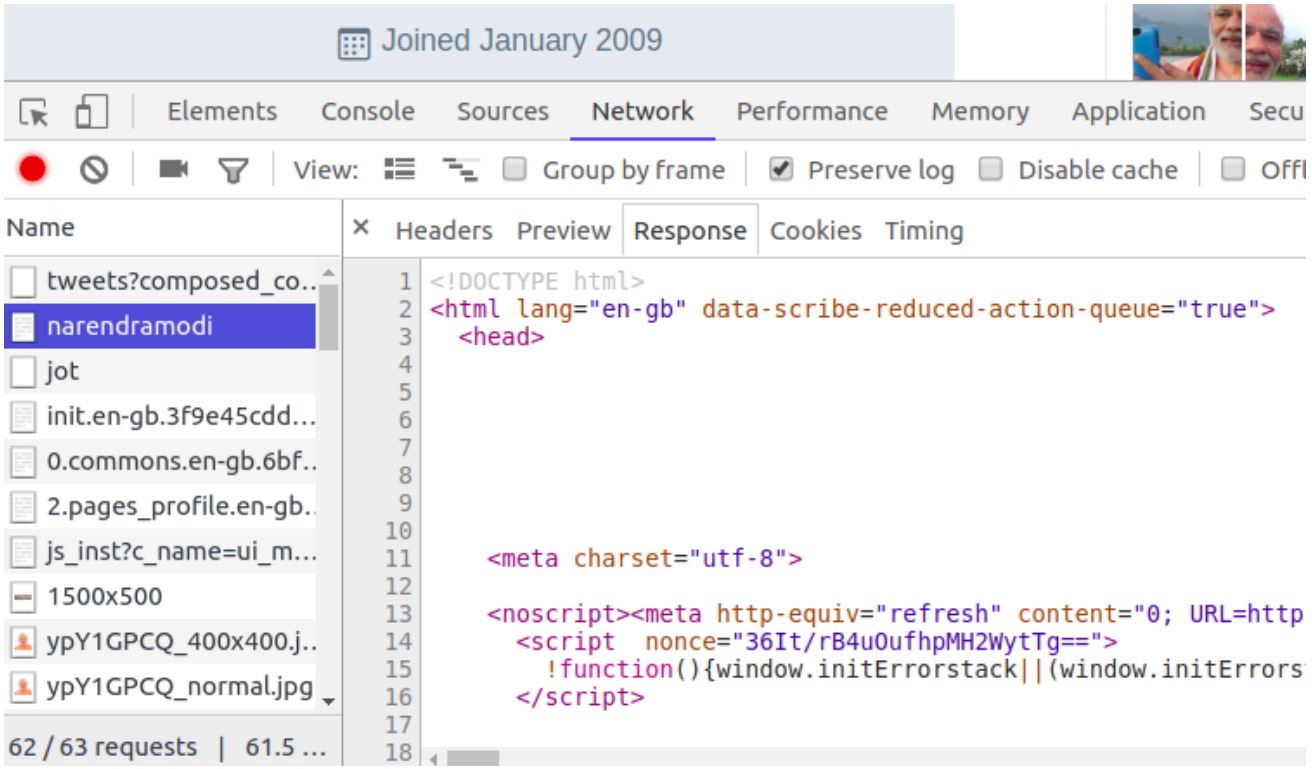
```
pip install beautifulsoup4==4.6.0 bs4==0.0.1 requests==2.18.4
```

Analysing Twitter Web Requests:

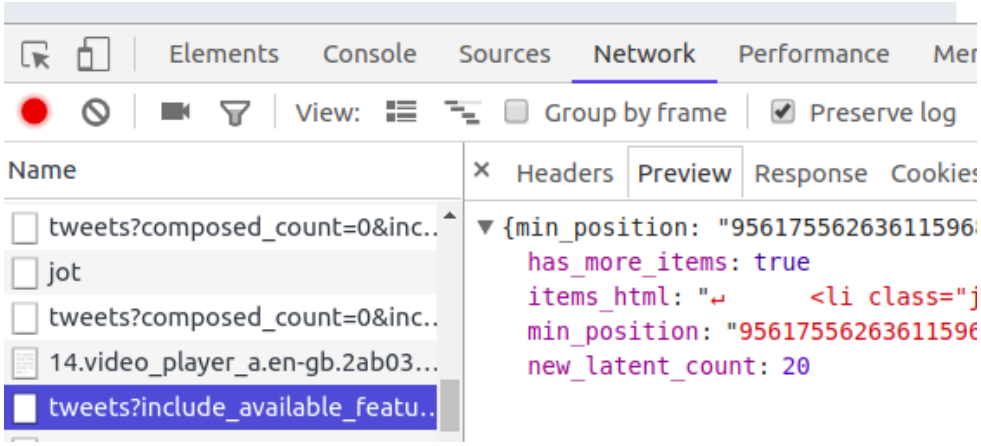
Lets say we want to scrape all the tweets made by Honourable Prime Minister of India, Shri Narendra Modi.
Go to the browser, I am using Chrome, press F12 to open the debugging tool.

Now go the the URL `https://twitter.com/narendramodi`. In the network tab of debugging tool, you will see the response of request made to URL /narendramodi.

Response is an HTML page. We will convert this HTML response into a BeautifulSoup object and will extract the tweets.



If you scroll down the page to load more tweets, you will see more requests being sent where response is not simple HTML but is in JSON format.



Extracting tweets from HTML content:

First inspect the tweet element on web page. You will see that all the tweets are enclosed in `li` HTML tag. Actual tweet text is inside a `p` tag which is the descendent of `li` tag.
We will first get all the `li` tags and then `p` tags from each `li` tag. Text contained in the `p` tag is what we need.

Code to start with:

```
# script to scrape tweets by a twitter user.
# Author - ThePythonDjango.Com
# dependencies - BeautifulSoup, requests

from bs4 import BeautifulSoup
import requests
import sys
import json


def usage():
    msg = """
    Please use the below command to use the script.
    python script_name.py twitter_username
    """
    print(msg)
    sys.exit(1)


def get_username():
    # if username is not passed
    if len(sys.argv) < 2:
        usage()
    username = sys.argv[1].strip().lower()
    if not username:
        usage()

    return username


def start(username = None):
    username = get_username()
    url = "http://www.twitter.com/" + username
    print("\n\nDownloading tweets for " + username)
    response = None
    try:
        response = requests.get(url)
    except Exception as e:
        print(repr(e))
        sys.exit(1)

    if response.status_code != 200:
```

```
        print("Non success status code returned "+str(response.status_code))
        sys.exit(1)

    soup = BeautifulSoup(response.text, 'lxml')

    if soup.find("div", {"class": "errorpage-topbar"}):
        print("\n\n Error: Invalid username.")
        sys.exit(1)

    tweets = get_tweets_data(username, soup)
```

We will start with `start` function. First collect the username from command line and then send the request to twitter page.

If there is no exception and status code returned in response is `200` i.e. success, proceed otherwise exit.

Convert the response text into BeautifulSoup object and see if there is any `div` tag in the HTML with class `errorpage-topbar` . If yes that means the username is invalid. Although this check is not required because in case of invalid username, 404 status is returned which will be checked in `status_code` check condition.

Extract tweet text:

```
def get_this_page_tweets(soup):
    tweets_list = list()
    tweets = soup.find_all("li", {"data-item-type": "tweet"})
    for tweet in tweets:
        tweet_data = None
        try:
            tweet_data = get_tweet_text(tweet)
        except Exception as e:
            continue
            #ignore if there is any loading or tweet error

        if tweet_data:
            tweets_list.append(tweet_data)
            print(".", end="")
            sys.stdout.flush()

    return tweets_list


def get_tweets_data(username, soup):
    tweets_list = list()
    tweets_list.extend(get_this_page_tweets(soup))
```

As discussed, we first find out all `li` tags and then for each element we try to get tweet text out of that `li` tag.

We keep printing a dot on screen every time a tweet is scrapped successfully to show the progress otherwise user may think that script is doing nothing or is hanged.

```
def get_tweet_text(tweet):
    tweet_text_box = tweet.find("p", {"class": "TweetTextSize TweetTextSize--normal js-tweet-text tweet-text"})
    images_in_tweet_tag = tweet_text_box.find_all("a", {"class": "twitter-timeline-link u-hidden"})
    tweet_text = tweet_text_box.text
    for image_in_tweet_tag in images_in_tweet_tag:
        tweet_text = tweet_text.replace(image_in_tweet_tag.text, '')

    return tweet_text
```

We sometimes have images inside tweets, we will discard those images as of now. We do this by getting image tags inside tweets and replacing image text by empty string.

Scrapping more tweets:

So far we were able to get tweets from first page. As we load more pages, when scrolling down, we get JSON response. We need to parse JSON response, which is slightly different.

```
def get_tweets_data(username, soup):
    tweets_list = list()
    tweets_list.extend(get_this_page_tweets(soup))

    next_pointer = soup.find("div", {"class": "stream-container"})["data-min-position"]

    while True:
        next_url = "https://twitter.com/i/profiles/show/" + username + \
                   "/timeline/tweets?include_available_features=1&" \
                   "include_entities=1&max_position=" + next_pointer + "&reset_error_state=false"

        next_response = None
        try:
            next_response = requests.get(next_url)
        except Exception as e:
            # in case there is some issue with request. None encountered so far.
            print(e)
            return tweets_list

        tweets_data = next_response.text
        tweets_obj = json.loads(tweets_data)
        if not tweets_obj["has_more_items"] and not tweets_obj["min_position"]:
            # using two checks here bcz in one case has_more_items was false but there were more items
            print("\nNo more tweets returned")
            break
        next_pointer = tweets_obj["min_position"]
        html = tweets_obj["items_html"]
        soup = BeautifulSoup(html, 'lxml')
        tweets_list.extend(get_this_page_tweets(soup))

    return tweets_list
```

First we check if there are more tweets. If yes then we find the next pointer and create the next URL. Once JSON is received, we take out the `items_html` part and repeat the process of creating soup and fetching tweets. We keep doing this until there are no more tweets to scrap. We know this by looking at the variable `has_more_items` and `min_position` in JSON response.

Complete script:

Now all the functions are completed. Let put them together.
Download the complete script from GitHub (https://github.com/anuragrana/Python-Scripts).

Running the script:

Assuming you have installed dependencies in virtual environment, lets run the script.

```
(scrappingvenv) rana@Nitro:python_scripts$ python tweets_scrapper.py narendramodi


Downloading tweets for narendramodi
.....................................................................................................................
No more tweets returned

Dumping data in file narendramodi_twitter.json
844 tweets dumped.
(scrappingvenv) rana@Nitro:python_scripts$
```

You might introduce some wait between requests if you get any rate limit errors.

Dumping data in file:

You might want to dump the data in text file. I prefer dumping data in JSON format.

```
# dump final result in a json file
def dump_data(username, tweets):
    filename = username+"_twitter.json"
    print("\nDumping data in file " + filename)
    data = dict()
    data["tweets"] = tweets
    with open(filename, 'w') as fh:
        fh.write(json.dumps(data))

    return filename
```

Let us know if you face any issues.

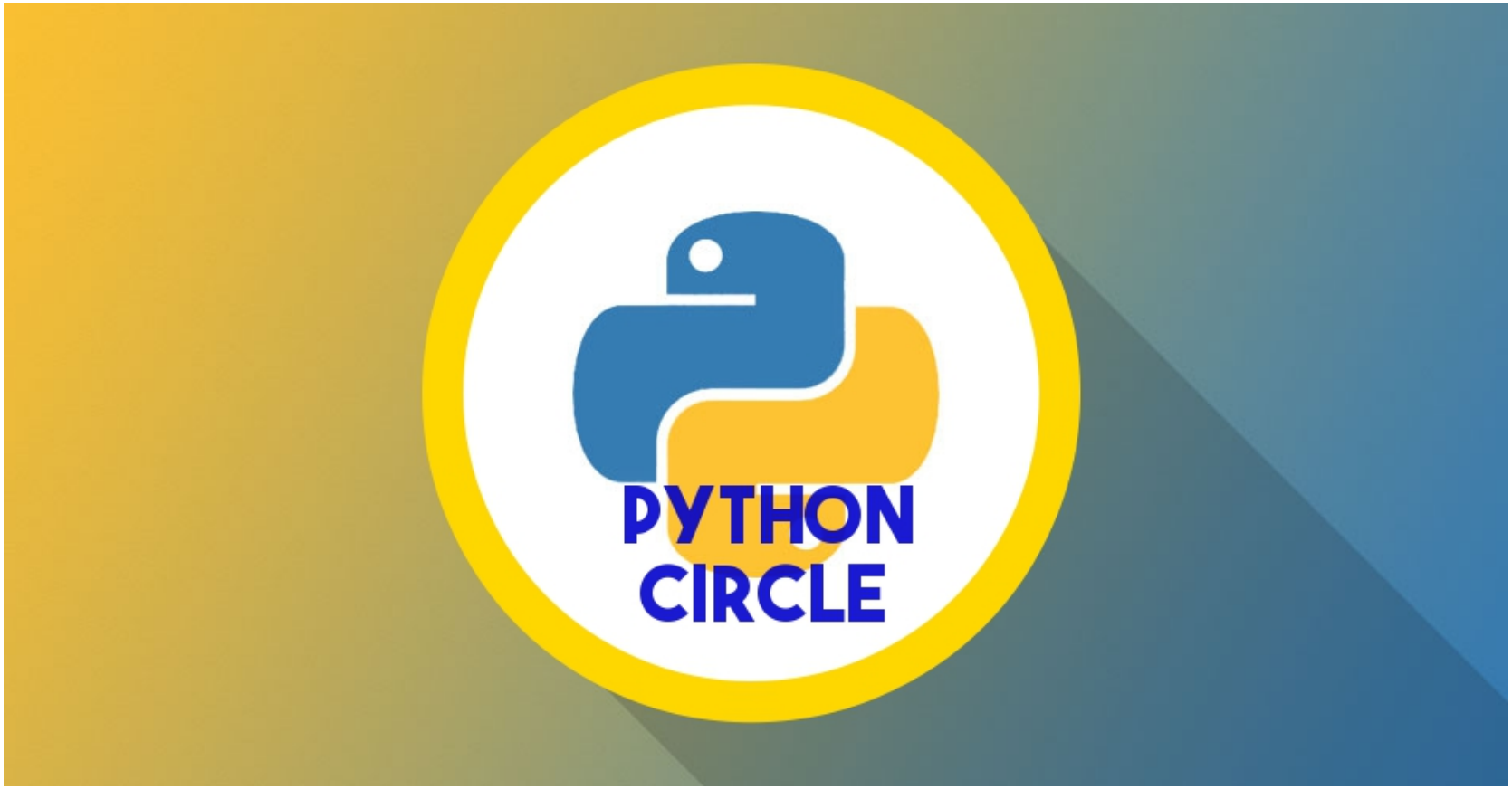
beautifulsoup (/tag/beautifulsoup/) scraping (/tag/scraping/) 0 6663

Related Articles:

Python Script 14: Scraping news headlines using python beautifulsoup (/post/678/python-script-14-scraping-news-headlines-using-python-beautifulsoup/)

Scraping news headlines using python beautifulsoup, web scraping using python, python script to scrape news, web scraping using beautifulsoup, news headlines scraping using python, python programm to get news headlines from web...

Read Full Article (/post/678/python-script-14-scraping-news-headlines-using-python-beautifulsoup/)



Python Script 2 : Crawling all emails from a website (/post/217/python-script-2-crawling-all-emails-from-a-website/)

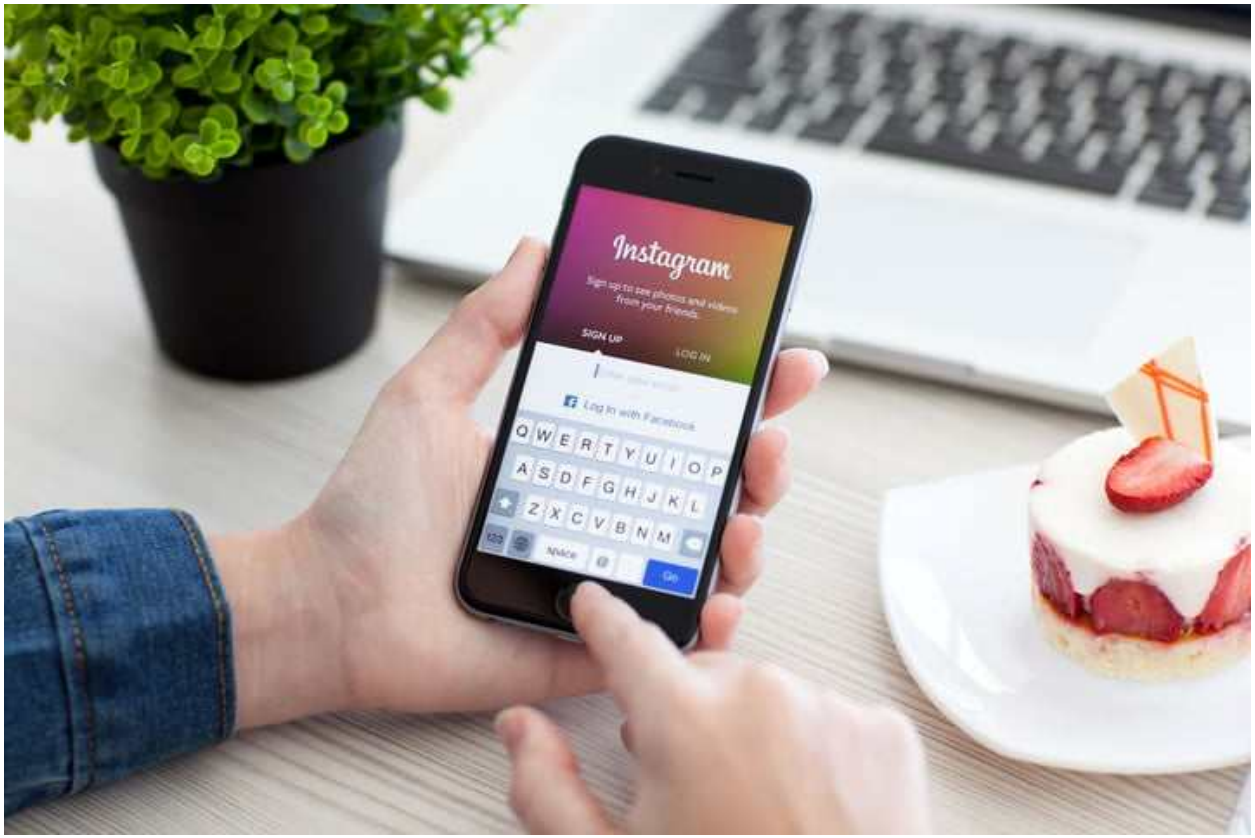
Website crawling for email address, web scraping for emails, data scraping and fetching email adress, python code to scrape all emails froma websites, automating the email id scraping using python script, collect emails using python script...

Read Full Article (/post/217/python-script-2-crawling-all-emails-from-a-website/)

How to create completely automated telegram channel with python (/post/265/how-to-create-completely-automated-telegram-channel-with-python/)

Creating a completely automated telegram channel to generate and post content using python code on regular basis. Automating the Telegram channel using python script...

Read Full Article (/post/265/how-to-create-completely-automated-telegram-channel-with-python/)



py_instagram_dl - The Python Package to Download All pictures of an Instagram User (/post/447/py_instagram_dl-the-python-package-to-download-all-pictures-of-an-instagram-user/)

Download all instagram images for any user using this python package....

Read Full Article (/post/447/py_instagram_dl-the-python-package-to-download-all-pictures-of-an-instagram-user/)

---

0 thoughts on 'Python Script 7: Scraping Tweets Using Beautifulsoup'

Leave a comment:

Type your comment here. For code use pastebin link. Limit 500 chars.

Your email please.

Your Name

Submit

*All Fields are mandatory. **Email Id will not be published publicly.

SUBSCRIBE

Please subscribe to get the latest articles in your mailbox.

Your Email ID Please

Subscribe

Recent Posts:

- 5 lesser used Django template tags (/post/694/5-lesser-used-django-template-tags/)

- Solving Python Error- KeyError: 'key_name' (/post/693/solving-python-error-keyerror-key_name/)
- Intellectual property Law and Coding (/post/692/intellectual-property-law-and-coding/)
- Python Script 17: Setting bing image of the day as desktop wallpaper (/post/691/python-script-17-setting-bing-image-of-the-day-as-desktop-wallpaper/)
- Django Template Fiddle Launched !!!! (/post/690/django-template-fiddle-launched/)
- Python Script 16: Generating word cloud image of a text using python (/post/689/python-script-16-generating-word-cloud-image-of-a-text-using-python/)
- Preventing cross-site scripting attack on your Django website (/post/688/preventing-cross-site-scripting-attack-on-your-django-website/)
- How to generate ATOM/RSS feed for Django website (/post/687/how-to-generate-atomrss-feed-for-django-website/)
- How to create sitemap of Django website (/post/686/how-to-create-sitemap-of-django-website/)
- For loop in Django template (/post/685/for-loop-in-django-template/)
- How to add Favicon to Django websites (/post/684/how-to-add-favicon-to-django-websites/)
- Scraping data of 2019 Indian General Election using Python Request and BeautifulSoup and analyzing it (/post/683/scraping-data-of-2019-indian-general-election-using-python-request-and-beautifulsoup-and-analyzing-it/)
- How to display PDF in browser in Django instead of downloading it (/post/682/how-to-display-pdf-in-browser-in-django-instead-of-downloading-it/)
- Solving Python Error - UnboundLocalError: local variable 'x' referenced before assignment (/post/680/solving-python-error-unboundlocalerror-local-variable-x-referenced-before-assignment/)
- Python Script 15: Creating a port scanner in 8 lines of python (/post/679/python-script-15-creating-a-port-scanner-in-8-lines-of-python/)
- Python Script 14: Scraping news headlines using python beautifulsoup (/post/678/python-script-14-scraping-news-headlines-using-python-beautifulsoup/)
- How to download large csv files in Django (/post/677/how-to-download-large-csv-files-in-django/)
- Text based snake and ladder game in python (/post/676/text-based-snake-and-ladder-game-in-python/)
- Logging databases changes in Django Application (/post/675/logging-databases-changes-in-django-application/)
- Python Script 13: Generating ascii code from Image (/post/674/python-script-13-generating-ascii-code-from-image/)
- Python Frequently Asked Question 1: What can I do in Python? (/post/672/python-frequently-asked-question-1-what-can-i-do-in-python/)
- How to start with Python Programming - A beginner's guide (/post/671/how-to-start-with-python-programming-a-beginners-guide/)
- Python program to find whether a given year is leap year or not (/post/670/python-program-to-find-whether-a-given-year-is-leap-year-or-not/)
- Python Django Project Idea for beginners (/post/669/python-django-project-idea-for-beginners/)
- Uploading a file to FTP server using Python (/post/668/uploading-a-file-to-ftp-server-using-python/)
- Print statement in Python vs other programming languages (/post/667/print-statement-in-python-vs-other-programming-languages/)
- Automating Facebook page posts using python script (/post/666/automating-facebook-page-posts-using-python-script/)
- try .. except .. else .. in python with example (/post/664/try-except-else-in-python-with-example/)

- Python Script 12: Drawing Indian National Flag Tricolor using Python Turtle (/post/662/python-script-12-drawing-indian-national-flag-tricolor-using-python-turtle/)
- Python Script 11: Drawing Flag of United States of America using Python Turtle (/post/661/python-script-11-drawing-flag-of-united-states-of-america-using-python-turtle/)
- Solving Django Error: TemplateDoesNotExist at /app_name/ (/post/660/solving-django-error-templatedoesnotexist-at-app_name/)
- Adding Email Subscription Feature in Django Application (/post/657/adding-email-subscription-feature-in-django-application/)
- Using PostgreSQL Database with Python (/post/656/using-postgresql-database-with-python/)
- Programming On Raspberry Pi With Python: Activate LED and Buzzer on Motion Detection (/post/654/programming-on-raspberry-pi-with-python-activate-led-and-buzzer-on-motion-detection/)
- Programming on Raspberry Pi with Python: Controlling LED (/post/652/programming-on-raspberry-pi-with-python-controlling-led/)
- Programming on Raspberry Pi with Python: Sending IP address on Telegram channel on Raspberry Pi reboot (/post/651/programming-on-raspberry-pi-with-python-sending-ip-address-on-telegram-channel-on-raspberry-pi-reboot/)
- Programming on Raspberry Pi with Python: WIFI and SSH configuration (/post/650/programming-on-raspberry-pi-with-python-wifi-and-ssh-configuration/)
- Programming on Raspberry Pi with Python: Raspberry Pi Setup (/post/649/programming-on-raspberry-pi-with-python-raspberry-pi-setup/)
- Iterator and Generators in Python: Explained with example (/post/648/iterator-and-generators-in-python-explained-with-example/)
- Top 5 Python Books (/post/646/top-5-python-books/)
- Encryption-Decryption in Python Django (/post/641/encryption-decryption-in-python-django/)
- Sending Emails Using Python and Gmail (/post/628/sending-emails-using-python-and-gmail/)
- How to Track Email Opens Sent From Django App (/post/626/how-to-track-email-opens-sent-from-django-app/)
- Python Tip 1: Accessing localhost Django webserver over the Internet (/post/618/python-tip-1-accessing-localhost-django-webserver-over-the-internet/)
- 5 common mistakes made by beginner python programmers (/post/602/5-common-mistakes-made-by-beginner-python-programmers/)
- How to upload and process the Excel file in Django (/post/591/how-to-upload-and-process-the-excel-file-in-django/)
- Creating sitemap of Dynamic URLs in your Django Application (/post/584/creating-sitemap-of-dynamic-urls-in-your-django-application/)
- Adding Robots.txt file to Django Application (/post/578/adding-robotstxt-file-to-django-application/)
- Python Script 3: Validate, format and Beautify JSON string Using Python (/post/576/python-script-3-validate-format-and-beautify-json-string-using-python/)
- Displaying custom 404 error (page not found) page in Django 2.0 (/post/564/displaying-custom-404-error-page-not-found-page-in-django-20/)

Tags:

404 (/tag/404/) 500 (/tag/500/) AJAX (/tag/ajax/) API (/tag/api/) AUTHENTICATION (/tag/authentication/) AUTOMATION (/tag/automation/)
AWS (/tag/aws/) BACKUP (/tag/backup/) BEAUTIFULSOUP (/tag/beautifulsoup/) BOOK (/tag/book/) BREADBOARD (/tag/breadboard/)
CAPTCHA (/tag/captcha/) CELERY (/tag/celery/) COMMANDS (/tag/commands/) CSV (/tag/csv/) DATA (/tag/data/) DATABASE (/tag/database/)
DECRYPTION (/tag/decryption/) DJANGO (/tag/django/) DJANGO APP (/tag/django%20app/) DJANGO ERROR (/tag/django%20error/)
DOCKER (/tag/docker/) DOWNLOAD (/tag/download/) EC2 (/tag/ec2/) ELASTIC SEARCH (/tag/elastic%20search/) EMAIL (/tag/email/)
ENCRYPTION (/tag/encryption/) ERROR (/tag/error/) ERROR PAGE (/tag/error%20page/) EXCEL (/tag/excel/) EXCEPTION (/tag/exception/)
FACEBOOK (/tag/facebook/) FAQ (/tag/faq/) FEEDS (/tag/feeds/) FOR LOOP (/tag/for%20loop/) FORM (/tag/form/) FREE EMAILS (/tag/free%20emails/)
FTP (/tag/ftp/) GAME (/tag/game/) GENERATOR (/tag/generator/) GMAIL (/tag/gmail/) GPIO (/tag/gpio/) GRAPH API (/tag/graph%20api/)
HARDWARE (/tag/hardware/) HOSTING (/tag/hosting/) IMAGE (/tag/image/) INSTAGRAM (/tag/instagram/) IP ADDRESS (/tag/ip%20address/)
IP LAW (/tag/ip%20law/) ITERATOR (/tag/iterator/) JSON (/tag/json/) KIBANA (/tag/kibana/) LED (/tag/led/) LOGGING (/tag/logging/)
MISTAKE (/tag/mistake/) MONGODB (/tag/mongodb/) PACKAGE (/tag/package/) PAYMENT GATWAY (/tag/payment%20gatway/) PDF (/tag/pdf/)
PITFALLS (/tag/pitfalls/) PROGRAM (/tag/program/) PROJECT (/tag/project/) PROXY (/tag/proxy/) PYTHON BITES (/tag/python%20bites/)
PYTHON TIPS (/tag/python%20tips/) PYTHONANYWHERE (/tag/pythonanywhere/) RABBITMQ (/tag/rabbitmq/) RASPBERRYPI (/tag/raspberrypi/)
REQUESTS (/tag/requests/) RESPONSE (/tag/response/) SCRAPING (/tag/scraping/) SCRAPY (/tag/scrapy/) SCRIPT (/tag/script/)
SECURITY (/tag/security/) SENSOR (/tag/sensor/) SEO (/tag/seo/) SERVER (/tag/server/) SETUP (/tag/setup/) SIGNALS (/tag/signals/)
SITEMAP (/tag/sitemap/) SOCKET (/tag/socket/) SSH (/tag/ssh/) TAG (/tag/tag/) TELEGRAM (/tag/telegram/) TEMPLATES (/tag/templates/)
TIPS (/tag/tips/) TKINTER (/tag/tkinter/) TOR (/tag/tor/) TRACKING (/tag/tracking/) TRY CATCH (/tag/try%20catch/) TURTLE (/tag/turtle/)
UPLOAD (/tag/upload/) VIRTUAL ENV (/tag/virtual%20env/) WIFI (/tag/wifi/) WORD CLOUD (/tag/word%20cloud/)