

Forum:[Hack Forums](#)

Thread: [Python Twitter Username Checker](#)

05317832892 - February 18, 2017, 12:24 pm

I think it's pathetic that people sell this for \$10 in the marketplace, it's literally just a few lines of code, anyways, I made my own. It works on the simple basis of if the username is available, visiting their profile will return a "HTTP 404 Not Found".

Spoiler ([Click to View](#))

Features:

- Optional Save to file
- Fairly quick
- Proxy-less

Ideas for the future:

- Multithread
- Maybe more sites supported

Open to suggestions and feedback! ???

phunted - February 18, 2017, 12:26 pm

I appreciate the fact you don't need to sell everything you can think of and want to give something back here.

Good job mate.

05317832892 - February 18, 2017, 12:30 pm

(Feb 18, 2017 01:26 PM)Jumps Wrote: [\[CITE\]](#) I appreciate the fact you don't need to sell everything you can think of and want to give something back here.

Good job mate.

Thanks, I like sharing my code so people can learn. Can't be dropping links or names but some idiot started selling a chrome password stealer \$5/month. It's literally just a stealer, it emails the pass back to you, not even FUD. Almost identical to my one just a different language, but \$5/month to milk some money out of little kids on HF.

Ri?dle - February 18, 2017, 12:41 pm

(Feb 18, 2017 01:30 PM)?Cam? Wrote: [\[CITE\]](#) Thanks, I like sharing my code so people can learn. Can't be dropping links or names but some idiot started selling a chrome password stealer \$5/month. It's literally just a stealer, it emails the pass back to you, not even FUD. Almost identical to my one just a different language, but \$5/month to milk some money out of little kids on HF.

i've done something like making post request on instagram too check if its available or not, and it filtered out all the bannned/invalid usernames. Anyway you can do that?

edit: just tried got limited

Ensky - February 18, 2017, 1:43 pm

Looks great.

I've made a few adjustments to the code in terms of presentability (mainly).

<https://ghostbin.com/paste/t9d9u>

Also, in your if statements you don't need to type

```
Code if save == True:
```

```
    store(username)
```

That's the equivalent of saying

```
Code if True == True
```

```
    # code
```

I'm not expert by any means, just throwing in some constructive criticism, loving the work you're outputting!

Suggestion: Make something similar to this, but it scans: Kik, Instagram Snapchat(If possible) and Twitter. It'd be like a "set" finder or something :)

05317832892 - February 18, 2017, 2:27 pm

(Feb 18, 2017 02:43 PM)aviv. Wrote: [\[CITE\]](#) Looks great.

I've made a few adjustments to the code in terms of presentability (mainly).

<https://ghostbin.com/paste/t9d9u>

Also, in your if statements you don't need to type

```
Code if save == True:
```

```
    store(username)
```

That's the equivalent of saying

```
Code if True == True
```

```
    # code
```

I'm not expert by any means, just throwing in some constructive criticism, loving the work you're outputting!

Suggestion: Make something similar to this, but it scans: Kik, Instagram Snapchat(If possible) and Twitter. It'd be like a "set" finder or something :)

Love the idea and the edits.

Can only edit the paste bin on my PC so il update it tomorrow. Gonna look into kik, snapchat and Facebook.

Ensky - February 18, 2017, 2:30 pm

(Feb 18, 2017 03:27 PM)?Cam? Wrote: [\[CITE\]](#) Love the idea and the edits.

Can only edit the paste bin on my PC so il update it tomorrow. Gonna look into kik, snapchat and Facebook.

I'm not sure if it works or not as I don't use URLLib. I'll make my own in requests and post it here :)

Here's my code:

PHP Code

```
<?php
import requests

class User:

    def __init__(self, username):
        self.name = username.lower()

    def is_available(self):
        r = requests.get('https://twitter.com/' + self.name)
        return True if('<title>Twitter / ?</title>' in r.text) else False

    def write(self):
        with open('available.txt', 'a') as f:
            f.write(self.name)
            f.close()

for username in open('usernames.txt', 'r'):
    user = username.rstrip('\n')
    user = User(username)

    if(user.is_available()):
        print('[+] ', user.name)
        user.write()
    else:
        print('[-] ', user.name)
```

obby - February 18, 2017, 2:52 pm

A few problems with this, checking peoples twitter.com/%s you will get a 404 if they're suspended, switch over to the username_available endpoint on the twitter signup page, your code is really messy this could be done in ~20 lines

(i broke the url because i cant post urls)

PHP Code

```
<?php
from __future__ import print_function

import requests
import sys

for user in open(sys.argv[1]):
```

```

user = user.strip().rstrip()
res = requests.get(
    'http?s?://twitter.com/users/username_available',
    params={
        'context': 'signup',
        'custom': 'true',
        'suggest': '1',
        'suggest_on_username': 'true',
        'username': user,
    }
)
if res.json()['reason'] == 'taken':
    print('{0} taken'.format(user))
else:
    print('{0} available'.format(user))
    print(user, file=open(sys.argv[2], 'a'))

```

PHP Code

```

<?php
C:\Users\lobby\Documents>checker.py omfg.txt okay.txt
cocks taken
fewfiewfjwee available
omfgfijewfew available
teste2oeeee available

```

```

C:\Users\lobby\Documents>cat okay.txt
fewfiewfjwee
omfgfijewfew
teste2oeeee

```

```

C:\Users\lobby\Documents>

```

youw0t - February 18, 2017, 2:57 pm

(Feb 18, 2017 03:30 PM)aviv. Wrote: [\[CITE\]](#) PHP Code

```

<?php
def write(self):
    with open('available.txt', 'a') as f:
        f.write(self.name)
        f.close()

```

No need to f.close() here, as it closes the file for you if you are using 'with'.

Ensky - February 18, 2017, 4:19 pm

(Feb 18, 2017 03:57 PM)youw0t Wrote: [\[CITE\]](#) No need to f.close() here, as it closes the file for you if you are using 'with'.

Oh cool, I never knew that. Thanks!

Goodies - February 18, 2017, 8:41 pm

You should really get in the habit of writing asynchronous code when it comes to anything web related.

I took 2500 random words from some online dictionary and checked them with this script in just under a minute. Also, this script is very easily portable for pretty much any other site provided it 404's when a username is not valid.

```
Code [aarcher@Arch:Python/UnameChecker]$ head check.txt
```

```
lacuna
pandemonium
incantational
lymphadenosis
folletti
internodian
evil
```

```
prov
rainwash
copiability
```

```
[aarcher@Arch:Python/UnameChecker]$ cat check.txt | wc -l
2500
```

```
[aarcher@Arch:Python/UnameChecker]$ time ./twitter.py -f check.txt -o available.txt -q
./twitter.py -f check.txt -o available.txt -q 59.55s user 3.46s system 69% cpu 1:31.32 total
```

```
[aarcher@Arch:Python/UnameChecker]$
```

Source:

PHP Code

```
<?php
```

```
#!/usr/bin/env python3
```

```
from argparse import ArgumentParser
from functools import partial
from os.path import isfile
from string import ascii_letters, digits
import aiohttp
import asyncio
import re
import sys
```

```
USERNAME_URL = "https://twitter.com/{"
```

```
USERNAME_REGEX = r"^[a-zA-Z0-9_]{1,15}$"
```

```
async def check_username(username:str, output:str, quiet:bool, timeout:float, sem:asyncio.
```

```
BoundedSemaphore, loop:asyncio.AbstractEventLoop=None):
```

```
    loop = loop or asyncio.get_event_loop()
```

```
    async with sem:
```

```
        try:
```

```
            async with aiohttp.ClientSession(loop=loop) as client:
```

```
                async with client.get(USERNAME_URL.format(username), timeout=timeout) as response:
```

```
                    taken = response.status != 404
```

```
                    char, msg = {False: ('+', "Available"), True: ('-', "Taken")}.get(taken)
```

```
                    if not quiet:
```

```
                        print("[{}] {:<16s}: {}".format(char, username, msg))
```

```
                    if output and not taken:
```

```
                        with open(output, "a") as f:
```

```
                            print(username, file=f)
```

```

        return username, taken
    except Exception as e:
        print("[!] Error with username '{} ({}' ".format(username, e), file=sys.stderr)
        return username, None

def valid_username(username):
    return re.match(USERNAME_REGEX, username) is not None

def main(filename:str, output:str, quiet:bool, connections:int, timeout:float):
    try:
        with open(filename) as fin:
            usernames = list(filter(valid_username, map(str.strip, fin)))
        if output:
            with open(output, "w") as fout:
                pass # delete the contents of the file
    except IOError:
        print("{}' could not be read.".format(filename))
        return
    loop = asyncio.get_event_loop()
    sem = asyncio.BoundedSemaphore(connections)
    if not quiet:
        print("Loaded {} usernames...\n".format(len(usernames)))
    tasks = [check_username(username, output, quiet, timeout, sem, loop) for
username in usernames]
    loop.run_until_complete(asyncio.gather(*tasks))

if __name__ == "__main__":
    ap = ArgumentParser()
    ap.add_argument("-f", "--file", required=True, type=str, help="File of usernames to check.")
    ap.add_argument("-n", "--num-connections", type=int, default=20, help=
"Number of simultaneous asynchronous connections.")
    ap.add_argument("-t", "--timeout", type=float, default=5.0, help="Timeout in seconds.")
    ap.add_argument("-o", "--output", type=str, default=None, help="Output valid usernames.")
    ap.add_argument("-q", "--quiet", action="store_true", help="Do not output to stdout.")
    args = ap.parse_args()
    main(args.file, args.output, args.quiet, args.num_connections, args.timeout)

```

You can increase the concurrent connections if you have a better internet connection. Otherwise, some of the N connections may time out.

aihttp actually "borrowed" a lot of its API design from requests. There is also another module called grequests which makes requests asynchronous by adding a gevent wrapper. That module would be fine for 2.7, although asyncio is standard in Python 3 (part of standard library as of 3.4, but compatible with 3.3+).

On one of my low end servers with a good internet connection, using 250 concurrent connections, I was able to reliably check 5,000 usernames in 50 seconds. Unfortunately the host provider doesn't like when you generate THAT much traffic and shut my server off, but that's why it's a low end box that I basically only use for stunnel and testing scripts like this lol.

05317832892 - February 19, 2017, 2:28 am

(Feb 18, 2017 09:41 PM)Goodies Wrote: [\[CITE\]](#) You should really get in the habit of writing asynchronous code when it comes to anything web related.

I took 2500 random words from some online dictionary and checked them with this script in just

under a minute. Also, this script is very easily portable for pretty much any other site provided it 404's when a username is not valid.

```
Code [aarcher@Arch:Python/UnameChecker]$ head check.txt
```

```
lacuna
pandemonium
incantational
lymphadenosis
```

```
folletti
```

```
internodian
```

```
evil
```

```
prov
```

```
rainwash
```

```
copiability
```

```
[aarcher@Arch:Python/UnameChecker]$ cat check.txt | wc -l
```

```
2500
```

```
[aarcher@Arch:Python/UnameChecker]$ time ./twitter.py -f check.txt -o available.txt -q
```

```
./twitter.py -f check.txt -o available.txt -q 59.55s user 3.46s system 69% cpu 1:31.32 total
```

```
[aarcher@Arch:Python/UnameChecker]$
```

Source:

PHP Code

```
<?php
```

```
#!/usr/bin/env python3
```

```
from argparse import ArgumentParser
```

```
from functools import partial
```

```
from os.path import isfile
```

```
from string import ascii_letters, digits
```

```
import aiohttp
```

```
import asyncio
```

```
import re
```

```
import sys
```

```
USERNAME_URL = "https://twitter.com/{"
```

```
USERNAME_REGEX = r"^[a-zA-Z0-9_]{1,15}$"
```

```
async def check_username(username:str, output:str, quiet:bool, timeout:float, sem:asyncio.
```

```
BoundedSemaphore, loop:asyncio.AbstractEventLoop=None):
```

```
    loop = loop or asyncio.get_event_loop()
```

```
    async with sem:
```

```
        try:
```

```
            async with aiohttp.ClientSession(loop=loop) as client:
```

```
                async with client.get(USERNAME_URL.format(username), timeout=timeout) as response:
```

```
                    taken = response.status != 404
```

```
                    char, msg = {False: ('+', "Available"), True: ('-', "Taken")}.get(taken)
```

```
                    if not quiet:
```

```
                        print("{} {} {:<16s}: {}".format(char, username, msg))
```

```
                    if output and not taken:
```

```
                        with open(output, "a") as f:
```

```
                            print(username, file=f)
```

```
                    return username, taken
```

```
        except Exception as e:
```

```
            print("[!] Error with username '{}' ({}).format(username, e), file=sys.stderr)
```

```
        return username, None
```

```

def valid_username(username):
    return re.match(USERNAME_REGEX, username) is not None

def main(filename:str, output:str, quiet:bool, connections:int, timeout:float):
    try:
        with open(filename) as fin:
            usernames = list(filter(valid_username, map(str.strip, fin)))
        if output:
            with open(output, "w") as fout:
                pass # delete the contents of the file
    except IOError:
        print("{} could not be read.".format(filename))
        return
    loop = asyncio.get_event_loop()
    sem = asyncio.BoundedSemaphore(connections)
    if not quiet:
        print("Loaded {} usernames...\n".format(len(usernames)))
    tasks = [check_username(username, output, quiet, timeout, sem, loop) for
username in usernames]
    loop.run_until_complete(asyncio.gather(*tasks))

if __name__ == "__main__":
    ap = ArgumentParser()
    ap.add_argument("-f", "--file", required=True, type=str, help="File of usernames to check.")
    ap.add_argument("-n", "--num-connections", type=int, default=20, help=
"Number of simultaneous asynchronous connections.")
    ap.add_argument("-t", "--timeout", type=float, default=5.0, help="Timeout in seconds.")
    ap.add_argument("-o", "--output", type=str, default=None, help="Output valid usernames.")
    ap.add_argument("-q", "--quiet", action="store_true", help="Do not output to stdout.")
    args = ap.parse_args()
    main(args.file, args.output, args.quiet, args.num_connections, args.timeout)

```

You can increase the concurrent connections if you have a better internet connection. Otherwise, some of the N connections may time out.

aihttp actually "borrowed" a lot of its API design from requests. There is also another module called grequests which makes requests asynchronous by adding a gevent wrapper. That module would be fine for 2.7, although asyncio is standard in Python 3 (part of standard library as of 3.4, but compatible with 3.3+).

On one of my low end servers with a good internet connection, using 250 concurrent connections, I was able to reliably check 5,000 usernames in 50 seconds. Unfortunately the host provider doesn't like when you generate THAT much traffic and shut my server off, but that's why it's a low end box that I basically only use for stunnel and testing scripts like this lol.

I understand where your coming from, asyncio gives me terrible headaches though.

Still very nice what you've done. I'm thinking about better ways to do this, looking for 404 doesn't seem that reliable.

Wawawest - February 19, 2017, 3:40 pm

cool thread learnd alot thank you guys

Goodies - February 19, 2017, 6:52 pm

(Feb 19, 2017 03:28 AM)?Cam? Wrote: [CITE](#) I understand where your coming from, asyncio gives me terrible headaches though.

Still very nice what you've done. I'm thinking about better ways to do this, looking for 404 doesn't seem that reliable.

It was the hardest part imho about switching to Python 3. I didn't want to always rely on Twisted for async operations. Asyncio was very hard to learn, but I can't recommend it enough. Pop a few Excedrin like they're tic tacs and dive into the docs. It's hella worth it in the end.

05317832892 - February 19, 2017, 11:40 pm

(Feb 19, 2017 07:52 PM)Goodies Wrote: [\[CITE\]](#) It was the hardest part imho about switching to Python 3. I didn't want to always rely on Twisted for async operations. Asyncio was very hard to learn, but I can't recommend it enough. Pop a few Excedrin like they're tic tacs and dive into the docs. It's hella worth it in the end.

Ok fine. Gonna sit down and learn this, busy with school so it will probably be next weekend.

Goodies - February 20, 2017, 12:28 am

(Feb 20, 2017 12:40 AM)?Cam? Wrote: [\[CITE\]](#) Ok fine. Gonna sit down and learn this, busy with school so it will probably be next weekend.

If I can be fucked, I'll be releasing a few videos on it. I've been wanting to for a while actually. I even wrote up a script and a powerpoint for the first couple of videos.

05317832892 - February 20, 2017, 10:22 am

(Feb 20, 2017 01:28 AM)Goodies Wrote: [\[CITE\]](#) If I can be fucked, I'll be releasing a few videos on it. I've been wanting to for a while actually. I even wrote up a script and a powerpoint for the first couple of videos.

Yh that would be great.

Infamia - February 21, 2017, 7:36 am

>only a few lines of code

>shows 55 lines of code

wat

(?°?°)?(???? - February 21, 2017, 7:38 am

(Feb 21, 2017 08:36 AM)Infamia Wrote: [\[CITE\]](#) >only a few lines of code

>shows 55 lines of code

wat

The actual meat, the logic, of the code is only 10 lines.

Goodies - February 21, 2017, 9:02 am

(Feb 21, 2017 08:36 AM)Infamia Wrote: [\[CITE\]](#) >only a few lines of code

>shows 55 lines of code

wat

Is... is 55 a lot?

Retribute - February 21, 2017, 8:36 pm

(Feb 20, 2017 01:28 AM)Goodies Wrote: [\[CITE\]](#) If I can be fucked, I'll be releasing a few videos on it. I've been wanting to for a while actually. I even wrote up a script and a powerpoint for the first couple of videos. Yes. Yes please. ?

