

Forum:[Hack Forums](#)

Thread: [Fastest way to generate random data? \(Faster than while looping\)](#)

Smirking_Emoji - December 3, 2019, 2:49 pm

Hello everyone, first post and I'm just diving in. I'm a noob, but an eager to learn noob.

I'm trying to figure out how I can run through .txt files of lines and grabbing a random line to generate random data.

I have this paste here, it works for my experimentation using a while loop though it's rather slow. To generate 1000 lines takes like 10 seconds. I'm banging my head looking at asynchronous and generator methods but I can't figure it out. I haven't found any asyncio examples where I can tell the loop how many lines I want etc. I was thinking using the asyncio method would go fast if it were using more coroutines etc.

Anyway, this is the code. IOT's simple, it works .. it's just slow.

Thanks for any help in advance.

Seems I can't post pastebin links yet ...

PHP Code

```
<?php
import asyncio
import time
import random

kwTXT1 = 'global'
kwTXT1 = "C:\\Users\\Stryker\\Desktop\\py\\100_questions.txt"
kwTXT2 = 'global'
kwTXT2 = "C:\\Users\\Stryker\\Desktop\\py\\91K_nouns.txt"
kwTXT3 = 'global'
kwTXT3 = "C:\\Users\\Stryker\\Desktop\\py\\common_conjugated_verbs.txt"
kwTXT4 = 'global'
kwTXT4 = "C:\\Users\\Stryker\\Desktop\\py\\31K_verbs.txt"
kwTXT5 = 'global'
kwTXT5 = "C:\\Users\\Stryker\\Desktop\\py\\6K_adverbs.txt"
```

```
def fileA(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
def fileB(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
def fileC(fname):
    lines = open(fname).read().splitlines()
```

```

    return random.choice(lines)
def fileD(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
def fileE(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
def fileF(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)

oneK = 1
while oneK < 300:
    print((fileA(kwTXT1)) + ' ' + (fileB(kwTXT2)) + ' ' + (fileC(kwTXT3)) + ' ' + (fileD(kwTXT4)) + ' ' + (
fileE(kwTXT5)))

    oneK += 1

```

Lindros - December 4, 2019, 12:12 am

Instead of reading/iterating through the text files numerous times you should just read them in once to a data structure like a list or a tuple. That way you don't have to read through and split again and again. If you aren't going to be modifying the data then I would recommend a tuple because it is a bit faster. It might be possible that you'll have too much data and in that case it won't work because you'll need to use a database.

You can have separate functions for each thing nouns, verbs, adverbs, etc. The function or functions would return the lists and you won't have to continuously read each file again and again.

Hopefully you don't have to and if your program works currently with those file sizes it should work even better if you implement my suggestions.

Let me know :)

Also you should declare a main function and use this if statement at the bottom for best practices, it is easier to follow.

EDITED: Sorry left out the string on main.

```

Code if __name__ == "__main__":
    main()

```

Also use docstrings in your functions, an easy way to start getting into the habit is like this with triple quotes.

Code def function():

```

"""This is the one thing the function will be doing/basic description"""

```

```

    code block

```

```

    return something

```

Here is some python documentation on docstrings. Your function should typically be doing one thing only input -> output. This will help you not get confused looking at your code later.

<https://www.python.org/dev/peps/pep-0257/>

realsunjester - December 4, 2019, 12:36 am

get max number of lines in the file the select a random line from 1 to max using linecache

```
Code linecache.getline('myfilename', rand_num)
```

Kallu609 - December 4, 2019, 2:06 am

(Dec 4, 2019 01:12 AM)ZietzSeV Wrote: [\[CITE\]](#) Also you should declare a main function and use this if statement at the bottom for best practices, it is easier to follow.

```
Code if __name__ == __main__:
    main()
```

That wouldn't work, the "__main__" needs to be a string and it's not even needed for this case, see: <https://stackoverflow.com/a/419185/11679602>

Main function is good idea because of Python's scope.

(Dec 4, 2019 01:12 AM)ZietzSeV Wrote: [\[CITE\]](#) Also use docstrings in your functions, an easy way to start getting into the habit is like this with triple quotes.

Talk about docstrings doesn't solve his issue lol.

The user above me posted the best way to get line from file. You probably first want to open the file and retrieve the line count and then use "random" module to generate random integer.

Lindros - December 4, 2019, 10:44 am

(Dec 4, 2019 03:06 AM)Kallu609 Wrote: [\[CITE\]](#) That wouldn't work, the "__main__" needs to be a string and it's not even needed for this case, see: <https://stackoverflow.com/a/419185/11679602>

Main function is good idea because of Python's scope.

Talk about docstrings doesn't solve his issue lol.

The user above me posted the best way to get line from file. You probably first want to open the file and retrieve the line count and then use "random" module to generate random integer.

You're 100% right about the string main it was super late, I'll change that.

I already explained to him why it was unnecessarily slow. Teaching someone best practices is definitely important. I'm trying to improve his code generally so it is actually easier to work with.

Why would you want to read the file in every iteration of the loop? He needs to store the information it will make it a lot more efficient.

Kallu609 - December 4, 2019, 10:53 am

(Dec 4, 2019 11:44 AM)ZietzSeV Wrote: [\[CITE\]](#) Why would you want to read the file in every iteration of the loop? He needs to store the information it will make it a lot more efficient.

You only read the line count once outside the loop and use the count in the loop where linecache is used.

This is probably the best way to get line count from file:

<https://stackoverflow.com/a/19001475/11679602>

Lindros - December 4, 2019, 10:59 am

(Dec 4, 2019 11:53 AM)Kallu609 Wrote: [\[CITE\]](#) You only read the line count once outside the loop and use the count in the loop where linecache is used.

This is probably the best way to get line count from file:

<https://stackoverflow.com/a/19001475/11679602>

Oh I was talking about the original code. He is calling his functions every iteration and it reads and splits each iteration of the while loop to return his random line choices.

I'm assuming your method works fine, I just wasn't familiar with it.

Smirking_Emoji - December 4, 2019, 9:13 pm

Thanks guys for the answers, there's no 'like' or 'thanks' buttons on this forum? If so please point them out.

This is good stuff, it gets what I want super fast now, though I am having problem with printing out my results in one concatenated line. It's always coming out as a list, no matter if I use "end=' '" or not. I am inserting this into a textbox in a Tkinter widget and it's doing the same. ie:

Jolie
sin
prodigiously
When
Jolie
didn't
sin
prodigiously

When
Jolie
didn't
sin
prodigiously

It's not doing it like example below:

When Jolie didn't sin prodigiously

And oddly, "end=' '" does work for me if the string isn't a string variable ie,

PHP Code

```
<?php
print("real string", end=' ')
print("real fo real", end=' ')
```

This takes away the list output, but not when it's a sting variable.

PHP Code

```
<?php
import random
import linecache
```

```
questions = sum(1 for line in open('100_questions.txt'))
nouns = sum(1 for line in open('91K_nouns.txt'))
converbs = sum(1 for line in open('common_conjugated_verbs.txt'))
verbs = sum(1 for line in open('31K_verbs.txt'))
adverbs = sum(1 for line in open('6K_adverbs.txt'))
```

```

oneK = 1
while oneK < 1000:

    rQ = random.randint(1,questions + 1)
    rN = random.randint(1,nouns + 1)
    rC = random.randint(1,converbs + 1)
    rV = random.randint(1,verbs + 1)
    rA = random.randint(1,adverbs + 1)

    lineQ = linecache.getline('100_questions.txt', rQ)
    lineN = linecache.getline('91K_nouns.txt', rN)
    lineC = linecache.getline('common_conjugated_verbs.txt', rC)
    lineV = linecache.getline('31K_verbs.txt', rV)
    lineA = linecache.getline('6K_adverbs.txt', rA)

# ----- these all do the same wrong thing

    print(lineQ, end=' ')
    print(lineN, end=' ')
    print(lineC, end=' ')
    print(lineV, end=' ')
    print(lineA, end=' ')

# ----- these all do the same wrong thing

    print(lineQ, end=' '),
    print(lineN, end=' '),
    print(lineC, end=' '),
    print(lineV, end=' '),
    print(lineA, end=' '),

# ----- these all do the same wrong thing

    print((lineQ),(lineN),(lineC),(lineV),(lineA))

# ----- these all do the same wrong thing

    print((lineQ) + (lineN) + (lineC) + (lineV) + (lineA))

# ----- these all do the same wrong thing

    list = [lineQ, lineN, lineC, lineV, lineA]
    for c in list: print (c, end=' ', flush=True)

    oneK += 1

```

realsunjester - December 4, 2019, 9:35 pm

(Dec 4, 2019 10:13 PM)Smirking_Emoji Wrote: [\[CITE\]](#) Thanks guys for the answers, there's no 'like' or 'thanks' buttons on this forum? If so please point them out.

This is good stuff, it gets what I want super fast now, though I am having problem with printing out my results in one concatenated line. It's always coming out as a list, no matter if I use "end=' '" or not. I am inserting this into a textbox in a Tkinter widget and it's doing the same. ie:

```
Jolie
sin
prodigiously
When
Jolie
didn't
sin
prodigiously
```

```
When
Jolie
didn't
sin
prodigiously
```

It's not doing it like example below:

When Jolie didn't sin prodigiously

And oddly, "end=' '" does work for me if the string isn't a string variable ie,

PHP Code

```
<?php
print("real string", end=' ')
print("real fo real", end=' ')
```

This takes away the list output, but not when it's a sting variable.

PHP Code

```
<?php
import random
import linecache
```

```
questions = sum(1 for line in open('100_questions.txt'))
nouns = sum(1 for line in open('91K_nouns.txt'))
converbs = sum(1 for line in open('common_conjugated_verbs.txt'))
verbs = sum(1 for line in open('31K_verbs.txt'))
adverbs = sum(1 for line in open('6K_adverbs.txt'))
```

```
oneK = 1
while oneK < 1000:
```

```
    rQ = random.randint(1,questions + 1)
    rN = random.randint(1,nouns + 1)
    rC = random.randint(1,converbs + 1)
```

```
rV = random.randint(1,verbs + 1)
rA = random.randint(1,adverbs + 1)
```

```
lineQ = linecache.getline('100_questions.txt', rQ)
lineN = linecache.getline('91K_nouns.txt', rN)
lineC = linecache.getline('common_conjugated_verbs.txt', rC)
lineV = linecache.getline('31K_verbs.txt', rV)
lineA = linecache.getline('6K_adverbs.txt', rA)
```

----- these all do the same wrong thing

```
print(lineQ, end=' ')
print(lineN, end=' ')
print(lineC, end=' ')
print(lineV, end=' ')
print(lineA, end=' ')
```

----- these all do the same wrong thing

```
print(lineQ, end=' '),
print(lineN, end=' '),
print(lineC, end=' '),
print(lineV, end=' '),
print(lineA, end=' '),
```

----- these all do the same wrong thing

```
print((lineQ),(lineN),(lineC),(lineV),(lineA))
```

----- these all do the same wrong thing

```
print((lineQ) + (lineN) + (lineC) + (lineV) + (lineA))
```

----- these all do the same wrong thing

```
list = [lineQ, lineN, lineC, lineV, lineA]
for c in list: print (c, end=' ', flush=True)
```

```
oneK += 1
```

instead of printing the lines right away to the console/stdout you could concatenate it into string then print the final string.

Smirking_Emoji - December 4, 2019, 9:57 pm

(Dec 4, 2019 10:35 PM)realsunjester Wrote: [\[CITE\]](#) instead of printing the lines right away to the console/stdout you could concatenate it into string then print the final string.

Yeah I tried that and the same thing:

PHP Code

```
<?php
online = lineQ + lineN + lineC + lineV + lineA
```

```
print(online)
```

Output:

```
Why
corner
will think
pinning
physiologically
```

This:

PHP Code

```
<?php
online = lineQ, lineN, lineC, lineV, lineA
print(online)
```

Does this:

```
('How\n', 'chalkboard\n', "didn't go\n", 'pollards\n', 'heaps\n')
```

realsunjester - December 4, 2019, 10:08 pm

(Dec 4, 2019 10:57 PM)Smirking_Emoji Wrote: [\[CITE\]](#) Yeah I tried that and the same thing:

PHP Code

```
<?php
online = lineQ + lineN + lineC + lineV + lineA
print(online)
```

Output:

```
Why
corner
will think
pinning
physiologically
```

This:

PHP Code

```
<?php
online = lineQ, lineN, lineC, lineV, lineA
print(online)
```

Does this:

```
('How\n', 'chalkboard\n', "didn't go\n", 'pollards\n', 'heaps\n')
try using rtrim
```

```
Code linecache.getline('100_questions.txt', rQ).rstrip('\n')
```

Smirking_Emoji - December 4, 2019, 10:11 pm

(Dec 4, 2019 11:08 PM)realsunjester Wrote: [\[CITE\]](#) try using rtrim

```
Code linecache.getline('100_questions.txt', rQ).rstrip('\n')
```

My man, 100% respect. It's working flawlessly now!

There's no 'thank you' buttons on this forum?

Thanks man.

realsunjester - December 4, 2019, 10:13 pm

(Dec 4, 2019 11:11 PM)Smirking_Emoji Wrote: [\[CITE\]](#) My man, 100% respect. It's working flawlessly now!

There's no 'thank you' buttons on this forum?

Thanks man.

no worries

magicparty - December 5, 2019, 8:26 pm

Little snippet on Synchronous and Asynchronous:

Programming in a synchronous fashion is a plausible design for early computing devices (I'm talking 90s and 80s). When you make a call to perform I/O in this case writing to a file the application will literally not respond until it completes; hence when windows says "Not Responding". To combat this you should use the asynchronous technique allowing the program to perform other tasks while it is waiting for an I/O response.

-- I would also like to add that Python is an excellent language for processing large scale data but if you want to boost performance you could look into making direct System Calls to the Kernel using C/CPP. If you went with this route a small application would only be needed and could be executed via. bash from your Python program or script.