

Forum:[Hack Forums](#)

Thread: [Counting](#)

systx - April 15, 2017, 7:46 pm

I want to create a program that will count to a super high number

I have tried the following...

PHP Code

```
<?php
import threading
x = 0
def count():
    x += 1
def threads():
    for x in range(800):
        t = threading.Thread(target = x)
        t.start()
```

Obviously this fucks up badly so I tried using Queue, but each thread acted as it's own thread not sharing variables :(

I have also tried...

PHP Code

```
<?php
import threading
x = 0
lock = threading.Lock()
def count():
    lock.acquire()
    x += 1
    lock.release()
def threads():
    t = threading.Thread(target = x)
    t.start()
```

It's like only have one threading running though

So any tips on how to do this would be AWSOME!!! ?

Woes - April 15, 2017, 8:07 pm

(Apr 15, 2017 07:46 PM)systx Wrote: [\[CITE\]](#) I want to create a program that will count to a super high #

Wait what?! Lmao.

Code x=0

```
while True:
    x=x+1
    print(x)
```

Ri?dle - April 15, 2017, 9:19 pm

(Apr 15, 2017 07:46 PM)systx Wrote: [\[CITE\]](#) I want to create a program that will count to a super high #

I have tried the following...

PHP Code

```
<?php
import threading
x = 0
def count():
    x += 1
def threads():
    t = threading.Thread(target = x)
    t.start()
```

Obviously this fucks up badly so I tried using Queue, but each thread acted as it's own thread not sharing variables :(

I have also tried...

PHP Code

```
<?php
import threading
x = 0
lock = threading.Lock()
def count():
    lock.acquire()
    x += 1
    lock.release()
def threads():
    t = threading.Thread(target = x)
    t.start()
```

It's like only have one threading running though

So any tips on how to do this would be AWSOME!!! ?just a bit confused.. if you're on 2.7 and you're lagging try PHP Code

```
<?php
xrange(1, 99999)
so you're not making a list (atleast I think). python 3+ Code range(1,9999) you can also do,PHP
Code
<?php
range(9991,9999)
or Code range(1, 200, 5) # counts by 5
```

systx - April 15, 2017, 9:26 pm

(Apr 15, 2017 08:07 PM)Woes Wrote: [\[CITE\]](#) Wait what?! Lmao.

Code x=0

while True:

 x=x+1

 print(x)

Lol sorry # = Number

(Apr 15, 2017 09:19 PM)Ghosting12 Wrote: [\[CITE\]](#) just a bit confused.. if you're on 2.7 and you're lagging try PHP Code

```
<?php
xrange(1, 99999)
so you're not making a list (atleast I think). python 3+ Code range(1,9999) you can also do,PHP
Code
<?php
range(9991,9999)
or Code range(1, 200, 5) # counts by 5
Yeah I'm using py 3.x
Sadly I need to count up by 1.
It needs to count up to 2000000000000 sadly ?
It would take a couple years to do with one thread :(
```

Ri?dle - April 15, 2017, 10:43 pm

(Apr 15, 2017 09:26 PM)systx Wrote: [\[CITE\]](#) Lol sorry # = Number

Yeah I'm using py 3.x
Sadly I need to count up by 1.
It needs to count up to 2000000000000 sadly ?
It would take a couple years to do with one thread :(
may I ask why you need to count that?

```
try this
PHP Code
<?php
import threading
from itertools import count
```

```
counter = count(1)
```

```
def counting():
    while True:
        next(counter)
        print(counter)
```

```
for i in range(10):
    t = threading.Thread(target=counting).start()
```

obby - April 16, 2017, 5:26 am

He is creating a list. (to person who said xrange(int) or range(int) wasnt a list)

Secondly if you want to count to a higher humber just do xrange(9999999999999) or something?
Why all of that nonsense?

I can't imagine threading, async or multi processing making this faster...

But...

Code from threading import Thread

```
theInt = 0
```

```
def count():  
    global theInt  
    while True:  
        theInt += 1
```

```
for i in xrange(4):  
    Thread(target=count).start()
```

IWannaLearn144 - April 16, 2017, 6:02 am

Using threading to make the program count faster is not very useful at all. Threading, in my opinion, is only really useful if you are doing a lot of file i/o and networking tasks like port scanning. Threading use to be more useful before python 3.2 because there were a few efficiency issues. There was a guy that done a whole talk on this but i cant remember his name.

systx - April 16, 2017, 10:01 am

(Apr 16, 2017 05:26 AM)obby Wrote: [\[CITE\]](#) He is creating a list. (to person who said xrange(int) or range(int) wasnt a list)

Secondly if you want to count to a higher humber just do xrange(999999999999) or something?
Why all of that nonsense?

I can't imagine threading, async or multi processing making this faster...

But...

Code from threading import Thread

```
theInt = 0
```

```
def count():  
    global theInt  
    while True:  
        theInt += 1
```

```
for i in xrange(4):  
    Thread(target=count).start()
```

All the nonsense is so I can't count to a high number very quickly. For x in range(99999) is extremely slow as it only uses one thread. Also you don't need theInt globalized
Thanks for the input ?

(Apr 15, 2017 10:43 PM)Ghosting12 Wrote: [\[CITE\]](#) may I ask why you need to count that?

try this
PHP Code
<?php
import threading
from itertools import count

```
counter = count(1)
```

```
def counting():  
    while True:  
        next(counter)  
        print(counter)
```

```
for i in range(10):  
    t = threading.Thread(target=counting).start()
```

Sadly your program doesn't share variables correctly :(
I need it for a project of mine.

Ri?dle - April 16, 2017, 11:19 am

(Apr 16, 2017 10:01 AM)systx Wrote: [\[CITE\]](#) All the nonsense is so I can't count to a high number very quickly. For x in range(99999) is extremely slow as it only uses one thread. Also you don't need theInt globalized
Thanks for the input ?

Sadly your program doesn't share variables correctly :(
I need it for a project of mine.wym it doesnt share variables correctly? it works fine for me. maybe I don't understand what you mean, could you explain it

TheWarOwl - April 16, 2017, 1:02 pm

Do you mean something like this?

PHP Code

```
<?php  
from multiprocessing import Process, RawValue, Lock  
import time
```

```
class Counter(object):  
    def __init__(self, value=0):  
        # RawValue because we don't need it to create a Lock:  
        self.val = RawValue('i', value)  
        self.lock = Lock()  
  
    def increment(self):  
        with self.lock:  
            self.val.value += 1  
  
    def value(self):  
        with self.lock:  
            return self.val.value  
  
def inc(counter):  
    for i in range(1000):  
        counter.increment()
```

```
if __name__ == '__main__':  
    thread_safe_counter = Counter(0)
```

```
procs = [Process(target=inc, args=(thread_safe_counter,)) for i in range(100)]

for p in procs: p.start()
for p in procs: p.join()

print (thread_safe_counter.value())
```

systx - April 16, 2017, 2:39 pm

(Apr 16, 2017 01:02 PM)Piero-Bird Wrote: [\[CITE\]](#) Do you mean something like this?

PHP Code

```
<?php
from multiprocessing import Process, RawValue, Lock
import time

class Counter(object):
    def __init__(self, value=0):
        # RawValue because we don't need it to create a Lock:
        self.val = RawValue('i', value)
        self.lock = Lock()

    def increment(self):
        with self.lock:
            self.val.value += 1

    def value(self):
        with self.lock:
            return self.val.value

def inc(counter):
    for i in range(1000):
        counter.increment()

if __name__ == '__main__':
    thread_safe_counter = Counter(0)
    procs = [Process(target=inc, args=(thread_safe_counter,)) for i in range(100)]

    for p in procs: p.start()
    for p in procs: p.join()

    print (thread_safe_counter.value())
```

I'm not sure. I'm a little confused as how it works.

SIM-1866423 - April 17, 2017, 5:03 am

Idk whatever you are trying to do but im pretty sure there has some better ways to do. Currently just seems like you are trying to do some useless loop, why?

systx - April 17, 2017, 8:27 am

(Apr 17, 2017 05:03 AM)Mnemosyne Wrote: [\[CITE\]](#) Idk whatever you are trying to do but im pretty sure there has some better ways to do. Currently just seems like you are trying to do some useless loop, why?

Yeah I think your right, I decided to go in a different direction that works just fine, anyways thanks guys. ?

obby - April 18, 2017, 11:41 pm

(Apr 16, 2017 10:01 AM)systx Wrote: [CITE](#) All the nonsense is so I can't count to a high number very quickly. For x in range(99999) is extremely slow as it only uses one thread. Also you don't need theInt globalized

Thanks for the input ?

just because you said im wrong i decided to actually run the code :)

runs fine:

```
Code from threading import Thread
```

```
theInt = 0
```

```
def count():
```

```
    global theInt
```

```
    while True:
```

```
        theInt += 1
```

```
for i in range(4):
```

```
    Thread(target=count).start()
```

runs with exception: UnboundLocalError: local variable 'theInt' referenced before assignment

```
Code from threading import Thread
```

```
theInt = 0
```

```
def count():
```

```
    while True:
```

```
        theInt += 1
```

```
for i in range(4):
```

```
    Thread(target=count).start()
```

Thank you I'll take your apology in full.

Also expanding on the topic:

Counting to 4 billion!

```
Code root@DN16:~# cat omfg.py
```

```
from threading import Thread
```

```
theInt = 0
```

```
def count():
```

```
    global theInt
```

```
    for i in range(1000000000):
```

```
        theInt += 1
```

```
for i in range(4):
```

```
Thread(target=count).start()
```

```
root@DN16:~# time python3 omfg.py
```

```
real 0m26.972s
user 0m27.096s
sys 0m0.080s
root@DN16:~#
```

Fastest

```
Code root@DN16:~# cat omfg2.py
for i in range(400000000):
    continue
```

```
root@DN16:~# time python3 omfg2.py
```

```
real 0m10.600s
user 0m10.588s
sys 0m0.000s
root@DN16:~#
```

This ran for 3+ minutes I didnt bother letting it finish.

This code came from:

<http://eli.thegreenplace.net/2012/01/04/shared-counter-with-pythons-multiprocessing>

```
Code root@DN16:~# cat omfg3.py
import time
from multiprocessing import Process, Value, Lock
```

```
def func(val, lock):
    for i in range(100000000):
        with lock:
            val.value += 1

if __name__ == '__main__':
    v = Value('i', 0)
    lock = Lock()
    procs = [Process(target=func, args=(v, lock)) for i in range(4)]

    for p in procs: p.start()
    for p in procs: p.join()

    print(v.value)
root@DN16:~#
```

Multiprocessing requires locking/releasing due to needing to be concurrent. However 4 processes counting to 1 million without concurrency was faster but defeats the purpose.

In conclusion in my experience no parallelism will count faster than just single process/single thread mainly due to overhead (i think)

systx - April 19, 2017, 9:36 pm

(Apr 18, 2017 11:41 PM)obby Wrote: [\[CITE\]](#) Multiprocessing requires locking/releasing due to needing to be concurrent. However 4 processes counting to 1 million without concurrency was faster but defeats the purpose.

In conclusion in my experience no parallelism will count faster than just single process/single thread mainly due to overhead (i think)

Your last paragraph is exactly what I was thinking.

I decided to make a program that counts from a range of different numbers. For example, one thread would count from 1-100 while another one at the same time ran from 100-200 and so on. Hope this helps someone else with this problemo!

Elite Hacker 69 - April 19, 2017, 9:40 pm

I don't know much about threading but isn't that for parallel computing? Why would you want multiple threads for something sequential?

obby - April 19, 2017, 9:42 pm

(Apr 19, 2017 09:36 PM)systx Wrote: [\[CITE\]](#) Your last paragraph is exactly what I was thinking.

I decided to make a program that counts from a range of different numbers. For example, one thread would count from 1-100 while another one at the same time ran from 100-200 and so on. Hope this helps someone else with this problemo!

This is very simple.. My last example can be simply modded using map iirc to do this.

systx - April 20, 2017, 5:48 pm

(Apr 19, 2017 09:40 PM)Drink Tea Wrote: [\[CITE\]](#) I don't know much about threading but isn't that for parallel computing? Why would you want multiple threads for something sequential?

Yes, it is for parallel computing. If I counted to 200000000 sequentially than it would take a life time. By using threads I can count all of them fast and easily.

(Apr 19, 2017 09:42 PM)obby Wrote: [\[CITE\]](#) This is very simple.. My last example can be simply modded using map iirc to do this.

Thanks for the info!

Goodies - April 20, 2017, 11:20 pm

ALWAYS JOIN YOUR THREADS

Unless you daemonize them (which you didn't).