The Basics of Python Multithreading and Queues

JANUARY 23, 2014 | TROY FAWKES | 20 COMMENTS | DEVELOPMENT

I'm getting really annoyed at reading through programmer speak. Hey guys, this stuff really isn't that hard. You just suck at explaining things.

Multithreading in Python, for example. Or how to use Queues.

So here's something for myself next time I need a refresher. It's the bare-bones concepts of Queuing and Threading in Python.

Let's start with Queuing in Python.

Before you do anything else, import Queue.

```
Python

1
2 from Queue import Queue
3
```

A queue is kind of like a list:

```
python

1
2 my_list = []
3 my_list.append(1)
4 my_list.append(2)
5 my_list.append(3)
6 print my_list.pop(0)
7 # Outputs: 1
```

The above code creates a list, assigns it three values, then removes the first value in so the list now has only 2 values (which are 2 and 3).

```
python

1
2 my_queue = Queue(maxsize=0)
3 my_queue.put(1)
4 my_queue.put(2)
5 my_queue.put(3)
6 print my_queue.get()
7 my_queue.task_done()
8 * Outputs: 1
```

There are only a couple differences in how queues work visually. First we set a maximum size to the queue, where 0 means infinite. It's pretty dumb but I'm sure it's useful somehow.

Hey! We just launched The Conversation Handbook. Research carried out by the Carnegie Institute of Technology shows that 85 percent of your financial success is due to skills in "human engineering," your personality and ability to communicate, negotiate, and lead. If you're looking for something more than technical knowledge, take a look at the book!



Top Links

Top Links

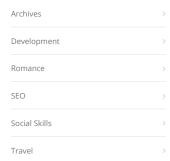
Get The Conversation Handbook

How To Improve Conversation Skills

How to Speak Clearly & Confidently

Communication Skills Training

Categories



The se**cond visite** difference is the **task_done()** bit at the end. That tells the queue that not only have I retrieved

threading. So let's just say in Queues, you have to call this.

The big important point about Queues is that they work really well with threading. In fact, you just can't use lists the way you can use queues in threading. That's why I'm even bothering to bring them up here.

Here's an example of a simple program that uses Queues:

```
python

from Queue import Queue

def do_stuff(q):
    while not q.empty():
    print q.get()
    q.task_done()

q = Queue(maxsize=0)

for x in range(20):
    q.put(x)

do_stuff(q)
```

It outputs 0-19. In like the most complicated way possible to output 0-19.

Notice how do_stuff() is just whipping through the whole queue. That's nice. But what if it was trying to do a big task, or a task that required a lot of waiting (like pulling data from APIs)? Assume for example that do_stuff() takes 30 second to run each time and it's just waiting on stupid APIs to return something. The function would take 30 seconds every time it ran, and it would run 20 times so it would take 10 minutes to get through just 20 items. That's really shitty.

Enter Python Threading.

Start with importing the right stuff:

Top Links

Threads are probably really complex. Or so I'm lead to believe. All you need to know for now, though, is that they use a worker function to get stuff done, they run at the same time, and you can pull them all together when they're done. So first you need to set up a worker function:

```
Python

2 def do_stuff(q):
3 while True:
4 print q.get()
5 q.task_done()
6
```

We're more or less just stealing the function from the last bit except we're setting it up for an infinite loop (while True). It just means that I want my threads always ready to accept new tasks.

Now I want to create the actual threads and set them running. **Before I do that, though**, I need to give them a Queue to work with. The Queue doesn't have to have anything on it, it just needs to be defined so that my threads know what they'll be working on. Here's how I set my (10) threads running:

So you see the Queue set up (as "q"), then I define a loop to run the thread creation bits 10 times. The first line in the loop sets up a thread and points it first at the do_stuff function, and then passes it "q" which is the Queue we just defined. Then something about a daemon, and we start the bugger. That's 10 threads running (remember the infinite loop in do_stuff()?) and waiting for me to put something in the Queue.

The rest of the code is the same as the Queue example so I'm just going to put it all together and let you figure it out:

```
python

from Queue import Queue
from threading import Thread

def do_stuff(q):
    while True:
    print q.get()
    q.task_done()

q = Queue(maxsize=0)
    num_threads = 10

for i in range(num_threads):
    worker = Thread(target=do_stuff, args=(q,))
    worker.setDaemon(True)
    worker.start()

for x in range(100):
    q.put(x)

q.join()
```

The only bit that should be new is the **q.join()** bit right at the end. This basically just waits until the queue is empty and all of the threads are done working (which it knows because **task_done()** will have been called on every element of the queue). If you were running a program in batches, you might use q.join() to wait for the batch to finish and then write the results to a file, and then just throw more tasks into the queue.

Consider revising the last 3 lines into a loop:

Fawkes Online Marketing.

Call Me At 647-575-9889

e Top Links

About

Blog

print "Batch " + str(y) + " Done'

It's cool that Queues can get added to willy nilly and these Threads will just pick them up, and whenever I want to I can stop and join all of them together for a second so I can check in, maybe write to a file or database or just let the user know that I'm still working away.

Remember the example I gave before about each run of do_stuff() taking 30 seconds? And since I had to run it 20 times it'd take 10 minutes? Now I can just run 20 different threads and the whole program will be done in about 30 seconds rather than 10 minutes. Obviously your results may vary, but it's definitely faster.

In any case, hope this helped. If you want the nitty gritty details, go read the documentation. This should get you started though.

f Facebook

▼ Twitter t Tumblr

② Pinterest

§+ Google+ in LinkedIn

E-Mail



ABOUT THE AUTHOR

My name is Troy Boileau but I go by Troy Fawkes. I'm a digital marketer working in Toronto. On top of that, my passions include social skills, networking and dating; subjects that I've been semi-professionally involved with for almost five years.

About Hello, I'm Troy. I'm a digital marketer specialized in Search Engine Optimization (SEO) and Web Channel Management. I can be your one stop shop for managing your website & digital marketing campaigns, or I can work with you on building and implementing an SEO strategy that will drive huge traffic and revenue to your website. © Copyright 2016 Troy Fawkes Contact Info Resources The Conversation Handbook Terms & Conditions Terms & Seconditions Terms & Conditions Terms & Conditions