Logging Currently Running Processes with Python

blog.pythonlibrary.org/2014/10/21/logging-currently-running-processes-with-python/

I was looking through some of my old code and noticed this old script where I was creating a log of all running processes every 5 minutes. I believe I originally wrote the code to help me diagnose rogue processes that were eating memory or pegging the CPU. I was using the psutil project to get the information I needed, so if you'd like to follow along you will need to download and install it as well.

Here's the code:

```
import os
import psutil
import time

#----------------------------------------------------------------------
def create_process_logs(log_dir):
    """
    Create a log of all the currently running processes
    """
    if not os.path.exists(log_dir):
        try:
            os.mkdir(log_dir)
        except:
            pass


    separator = "-" * 80
    col_format = "%7s %7s %12s %12s %30s"
    data_format = "%7.4f %7.2f %12s %12s %30s"
    while 1:
        procs = psutil.get_process_list()
        procs = sorted(procs, key=lambda proc: proc.name)

        log_path = os.path.join(log_dir, "procLog%i.log" % int(time.time()))
        f = open(log_path, 'w')
        f.write(separator + "\n")
        f.write(time.ctime() + "\n")
        f.write(col_format % ("%CPU", "%MEM", "VMS", "RSS", "NAME"))
        f.write("\n")

        for proc in procs:
            cpu_percent = proc.get_cpu_percent()
            mem_percent = proc.get_memory_percent()
            rss, vms = proc.get_memory_info()
            rss = str(rss)
            vms = str(vms)
            name = proc.name
            f.write(data_format % (cpu_percent, mem_percent, vms, rss, name))
            f.write("\n\n")
        f.close()
        print "Finished log update!"
        time.sleep(300)
        print "writing new log data!"

if __name__ == "__main__":
    log_dir = r"c:\users\USERNAME\documents"
    create_process_logs(log_dir)
```

Let's break this down a bit. Here we pass in a log directory, check if it exists and create it if it does not. Next we set up a few variables that contain formatting for the log file. Then we start an infinite loop that uses **psutil** to get all the currently running processes. We also sort the processes by name. Next, we open up a uniquely named log file and we write out each process'es CPU and memory usage along with it's VMS, RSS and name of the executable. Then we close the file and wait 5 minutes before doing it all over again.

In retrospect, it would probably have been better to write this information to a database like SQLite so that the data could be searched and graphed. In the meantime, hopefully you will find some useful tidbits in here that you can use for your own project.