Schedule the python program that computes the prefix average to run at 2 different times  
using cron.

Here is the prefix average python code:

import time

def prefix\_average(S):  
    n = len(S)  
    A = [0] \* n  
    for j in range (n):  
        total =0  
        for i in range(j+1):  
            total += S[i]  
        A[j] = total / (j+1)  
    return A  
start\_time = time.time()  
prefix\_average(range(20000))  
print("--- %s seconds ---" % (time.time() - start\_time))

1. First run the program every minute for 5 minutes. Record how long it takes the program   
to run each time.  The first recorded time we will call T1.  
Remove it from cron after it runs 5 times.

**HINT:** You can use the symbol ">>" in the crontab to append results to a file.

2. Next, generate CPU load on you system to see what happens to the execution time.  CPU load creates processes that compete for the CPU.  If you dont have alot of competing processes lowering the process priority will have little impact on the execution time.

To generate CPU load you can create a bash script called cpu\_load.sh and enter the following command:

cat /dev/urandom | gzip -9 | gzip -d | gzip -9 | gzip -d > /dev/null

Put this command in a bash script.  Make sure to put !#/bin/bash at the top of the script and then make it executable.  You can run multiple instances if you need to create more CPU load.

Now, run your cron job again and record the time for 5 minutes.  You should have 5 different timestamps, one for each minute.

We call this run sample T2  
How long did it take to run T2 with the CPU load script compared to T1?

3. Same as problem 2, but now run the program using cron but lower the process priority to +19.  Change the process priority when the process starts.

We will call this execution time T3.  How does the execution time of T3 compare to T2?

4. Now increase the process priority to -19.  You do not need to run this through cron.  You can run the prefix\_avg script directly from the command line just one time.  We will call this execution time T4.

How long did it take for T4 to complete compared to T1, T2, and T3?

5. Create a signal handler that generates an alarm after T1. For example if the execution time for T1 is 10 seconds, generate an alarm at 10 seconds.  Here are the steps:

1. In one window generate CPU load and start the python script with a low priority of +19.  This will cause the script to run longer than 10 seconds.

2. In another window run your signal handler.

You can also run the prefix\_avg script from the command line along with the CPU load script by pressing Ctrl-Z to run in the background.  This way you wont need to open multiple windows.  Just make sure you use kill the processes when you're done.