I've long been interested in stock trading and financial analysis, so I enjoy reading articles from time to time on techniques for analyzing/predicting stock market fluctuations. One idea that has been floating around for a while is the application of Fast Fourier Transforms (FFTs) to stock trend analysis. The idea is that stocks prices have time-dependent patterns, cycling up and down depending on days of the week, seasons, years, etc. Theoretically, an FFT could be used to extract those intrinsic business cycles and decompose them into a format more digestible to humans. In the last few weeks we have studied the cuRAND and cuFFT libraries, so this seemed like a perfect opportunity to practice for my future day-trading career by generating a practice set of stock prices using cuRAND and analyzing them using cuFFT. To start I generated a set of 25 random stocks, with 365 prices apiece representing a year's worth of daily prices changes. I've included an example below I generated and plotted with Python.
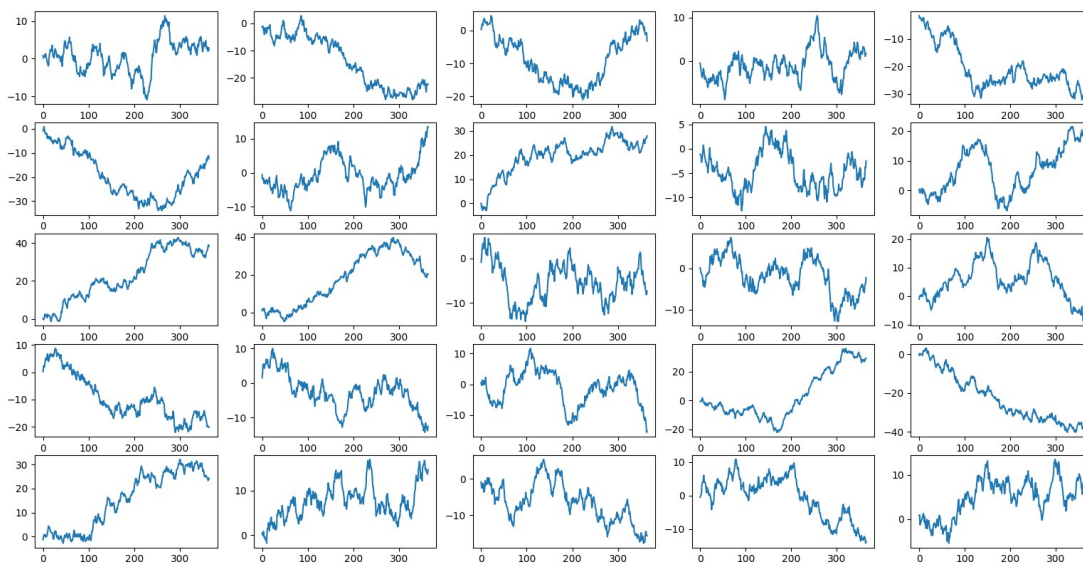

Figure: 25 Example Stock Prices Generated in Python

Afterwards, I simply ran the FFT and printed the values. The assignment seemed pretty simple and straight-forward, with no major issues on my end. After my first pass I realized I had set the blockSize to 365, so went back and brought the block size back down to 256, and implemented a stride to make up for the delta. Later I also realized that I had forgotten to normalize the FFT output, so I divided each result by nDays before printing.

Appendix: Python Code

```python
1   import matplotlib.pyplot as plt
2   from numpy.random import normal
3   from itertools import accumulate
4
5   fig, axs = plt.subplots(5,5)
6   axs = axs.flatten()
7
8   for i in range(25):
9       rand_nums = [min(max(normal(),-5),5) for _ in range(365)]
10      rand_sequence = list(accumulate(rand_nums))
11      axs[i].plot(rand_sequence)
12  plt.show()
```

Appendix: Sample Output