Subject: Python Floating Point Arithmetic Issue
From: Joseph Picone <joseph.picone@gmail.com>
Date: 6/7/23, 6:34 AM
To: ECE 1111 <temple_engineering_ece1111@googlegroups.com>
CC: nedc_research <nedc_research@googlegroups.com>

We will discuss this in the third week in ECE 1111:

> "float64 831.2549999999974" --> this is from the IPython Variable
> Explorer which resulted from Timestamp subtractions accumulated over
> iterations; the value is wrong due to floating point errors.
>

> I believe I solved the problem converting floats to integers, performing add and sub and then converting back to
> floats. Floating point errors were accumulating and then
> indexing events at

I talk about this in ECE 1111 - my programming course. In DSP, we often want to do this:

```
 for (float time = 0.0; time <= end_time; time += sample_duration) {
```

Of course, if you do this, the variable "time" accumulates a lot of round off error. It turns out a common bad case was when sample_duration was 1/8000 Hz, a common frequency for telephony 😄

So it is better to do this:

```
 for (long i = 0; i <= num_samples; i++) {
   float time = i * sample_duration;
```

I used to see this in action at a VCR production facility I worked at in college as a summer job. We life-tested VCR units. We would rack up about 10 units of a model, start them at the same time, and run them for a week. After a couple of days, the time displayed on the 10 VCRs would be different - off by as much as 10 secs. This is because the internal clocks on each unit were slightly different in terms of frequency.

It was a valuable lesson to learn even before I knew a lot about programming 😄

-Joe