John D. Blake

12 Jume 2024

IT FDN 110 A Sp 24: Foundations Of Programming: Python

Assignment08

https://github.com/jdblake2357/IT-FDN-110-Work

What I Learned in Python 101
or
"Programming and Me"

My computer programming story begins in the Sears Roebuck that anchored the west end of Northridge mall. Sears is now a shadow of its former self, and in a fitting allegory for the Rust Belt, the mall was bought out by some Chinese investors and soon abandoned. Criminals have since stripped it of its copper, and trees have taken root in the cracked asphalt of the fenced-off parking lot. It now sits, left for dead on the northern fringes of Milwaukee. But back in the late '70s things were different. The vibe was suburban aspiration and optimism, and the soundtrack was Super Tramp, Barry Manilow and Anne Murray.

Enter twelve year-old me, freshly trained up by my Hebrew School buddy, Ricky, on how to upend this peaceful world and spread chaos. You see a new curiosity was on display in the home electronics department. A home computer. And my buddy had shown me how to fill the screen of this magical device with hilarious and profane messages. Good times … good times.

Fast forward a decade or so. Past the Apple II, and past that MAGNIFICENT Commodore 64 which helped me maintain my sanity while on Fort Bragg between overseas deployments. It was the early '90s. I was back in Milwaukee. And I was heading into my second year of engineering school, doubling up over the summer term with Fortran and Modern Algebra. Fortran was pretty straightforward, but I sensed that Modern Algebra was going to be trouble. This was my first upper-division math elective, and when I opened the book ("Abstract Algebra with Applications"), there were no numbers. Only letters, squiggles and strange runes. Bottom line: Fortran was an easy A, and I struggled for a (gift of a) D- in Algebra. Thus the gauntlet was thrown, and I did the only sensible thing for someone trying to cram four years of engineering school into three years of GI-Bill funding. I double majored in applied math. I would go on to embrace "division of labor," immersing myself in the arcane math and physics that underpin the Thermal Science branch of Mechanical Engineering, while letting programmers write the code that would build tools to make me more productive. Besides, this was back when everyone knew that CompSci was a dead end. It was the lowest paid of all the engineering fields. Even lower than Civil![1]

Fast forward another three-plus decades, and I'm at the back-end of what's been a truly amazing career in Aerospace. Over the decades I've watched hardware speeds increase by orders of magnitude, while the interfaces have become friendlier and post-processing more elaborate. And all the while I've become increasingly vocal and crotchety, banging my spoon on my highchair, while prattling on about the dangers of confusing beauty and truth, and how garbage is still garbage no matter how amazing the post-processing. But at the end of the day I realize that for all of my cool experience, I can understandably be seen as an outdated guy who does math problems at lunch and is moored to a Fortran 77 understanding of how this all works. The world moved on. I hadn't.

So after talking through my situation with some very young, very sharp and very patient data analytics wizards who were walking me through some SQL queries they had written for me, I enrolled in this course.

———————————I'll post the following in the chat———————————

What I've taken away from this course:

1) Programming is (still) fun! I love puzzles, and getting stuck in code has been like getting stuck in a crossword. Hmm … huh … umm … Oh! Ha!
2) Programming tools have gotten friendlier. Starting out in the Idle shell was like going back to when I left off in the early 90's, only without the monochrome (green or amber) terminal in that nasty, nasty computer lab. But the PyCharm environment was like having a helping, friendly assistant looking over my shoulder and making suggestions on how we might just get this thing to work.
3) In the mid 90's, "Object Oriented" was all the rage. Thirty years on, I get what all the fuss was about. Neat. Portable. Efficient. Pretty cool stuff. I love the modularity and structure. And one of the most valuable things in the engineering world is to have validated tools. Once you have something that works reliably, it is key to reuse it to the fullest extent practical.

[1] Funny, but it turns out that the current Microsoft CEO (Satya Nadella) was walking out of the UW-Milwaukee Engineering department with a fresh MS in CompSci, just about the time I was walking in.

4) I really liked how the the product and project planning evolved over the course. And the discussion on requirements and architecture really pulled it together for me. A career in engineering has taught me that these are “Step Zero” on the path to a successful project. So it isn’t surprising that the software development process incorporates these concepts.
5) As for my future programming plans? I took this course on the advice of some very young and smart data analytics wizards on my team after telling them that my most-recent language was Fortran 77. So becoming a great developer is probably NOT in my future. But I’m really keen to revisit some of the numerical routines that I’ve cobbled together over the years for some esoteric engineering applications. They were super useful for very unique problems, and it would be great if they can live on after I march off to the glue factory.

Oh, and there’s one more thing: appreciation. The unbelievable IT infrastructure that envelopes us these days seems so normal that it is easy to forget the “before” times. Does anyone remember backpacking around Europe with guidebooks and calling around to hotels at train stations when you rolled in to town? Nothing was for sure. It’s why we all hauled around those stupid foam pads. Now, any of a dozen apps can manage your trip. Paper maps? LOL!

It’s easy to let it all wash over you, but at the end of the day, someone made that. So, this course has also been a reminder to appreciate.

Here’s an analogy:

Step 1 - Read something by Camus. Easy … simple plot … short sentences.

Step 2 - Learn about writing, and try to write something.

Step 3 - Now, READ something by Camus.