A short set of anecdotes, apropos of nothing.

When I was younger, I really liked programming! I loved the sense of accomplishment, I loved the problem solving, I loved sharing what I made with the people around me to both amuse and assist.

One particularly wise adult (somewhere around 1996) took me aside and said, “You know, you’re lucky you *enjoy* programming, because you won’t be able to make a living on it in the future. Doing it for love over money is a good idea.”

“Coding is over, with Object Oriented programming one person who is *much smarter than any of us could hope to be* will develop the library just *once* and we will all use it going forward, forever. Once a problem is solved it never needs solving again.

“In 5 years there’s going to be a library of objects, like books on a bookshelf, and every software problem will be solved by business people just snapping the object libraries they need together like LEGOs. They won’t need you at all.”

I thought about this advice, and how Software Engineering would be ending by the time I entered school. I realized I had not even thought about my education yet. I was in middle school. Programming was not it, though, I knew that.

I’m here nearly 30 years later and software continues to pay my bills, despite everything. Open source exists, there are libraries I can use to piece things together to solve all the time. New problem sets not covered by the garden path come up all the time. Clicking the LEGOs together continues to be a hard task. Every time we fix it at one level of abstraction we operate one level higher and the world keeps turning.

Whenever I’m threatened with a good time and someone proclaims “this is it for you” all that happens is my job becomes more annoying. Haven’t gotten the sweet release of extinction quite yet.

Around 1993 or so was the advent of the “Multimedia Age.” Multimedia was the buzzword. Software has to be *multimedia ready*. Education had to teach children to be ready for *the multimedia age*. If your tool, however inappropriate as it was, did not have multimedia features, you were going to be left behind. You *needed* a video guide. You *needed* to be on CD-ROM. This is just the new normal.

“Multimedia” just means “sound and video.” We had a high concept term for a very direct, low concept concept.

And the multimedia boom fizzled out. It became boring. Nobody is impressed by a video on a website and nobody thinks less of a website that doesn’t use sound and video if it’s not appropriate. You pop a tag in your HTML and your job is done. The amazing thing became mundane. The dream of “multimedia” became commonplace and everyone just accepted it as normal. I’m not aware of any industries that collapsed dramatically due to multimedia. Nobody really reskilled. Video editing is still a pretty rare thing to find, and we don’t commonly have sound engineers working on the audio UX of software products.

In 2000 a coworker took me aside and showed me his brand-new copy of IntelliJ IDE. “It’s over for us,” he said, “this thing makes it so programmers aren’t strictly necessary, like one person can operate this tool and they can lay the rest of us off.”

I was pretty awestruck, he got some amazing autocomplete right in the IDE. Without having to have a separate JavaDocs window open to the side, and without having to manually open the page for the class he needed documentation on, it just was there inline. It gave him feedback before the compile cycle on a bunch of issues that you normally don’t see until build. That was a nice bit of preventative work and seemed to have the potential to keep a developer in flow longer.

And then he showed me the killer feature “that’s going to get us all out of a job:” the refactoring tools.

He then proceeded to show me the tools, easily moving around code to new files, renaming classes across the codebase, all kinds of manual things that would have taken a person a few days to do on their own. It was magical.

After some thought I said, “that’s amazing, but does it write new logic too or does it just move code around?”

He didn’t seem fazed by that, and doubled down on the insistence that these powerful tools were our doom. I made a distinction between “useful” code and “filler” code, but apparently what is valued is not the quality and nature of the code but its volume and presence. This tool definitely gave both volume and presence to the tiny human-written nuggets within.

At my first job in High School I was working in an office in a suburban office park with programmers from many different local agencies. One guy I chatted up was a contractor: these people were highly regarded, somewhat feared specialists. The guy in question was working on a multi-year migration of some county health computer system from MUMPS to a more modern relational system. He showed me the main family of problems he was solving to show off how smart he was for solving them; they were largely rote problems of migrating table schemas and records in a pretty uniform way. But there were a lot of them, and he was working hard to meet his deadline!

I thought about it, and seeking his approval and validation, set out to help him. To show what I could do. I wrote a Python script that could solve the 85% case (it was mostly string manipulation) and even put a little TkInter dialog around it so he could select the files he wanted to migrate visually. It ran great, but he looked a little afraid when I demonstrated it to him:

“You didn’t show this to anyone else, did you?”

“Nope.”

“Oh thank God.”

I take it he used my tool because he had a lot more free time to goof off for the remaining six months of his contract. I don’t think he told anyone else what he had either, but I’m guessing that he had a lot more MUMPS migration contracts lined up when he could finish them in a matter of days.

At the same job, I was paid to maintain a series of government agency web sites. One of my main tasks was to keep a list of mental health providers up-to-date on an HTML page and upload it to the server.

This process was pretty mechanical: take Excel sheet from inbox, open in Excel, copy Excel table to HTML table.

Within a month I had a fully automated workflow:

- I used Windows Automation to watch my Outlook inbox
- When an email came in from the person who sent me the Excels it would download it
- Open the Excel file in excel using Windows Automation
- Export it to CSV from Excel (the automation did this, I simply watched a ghost remote control an Excel window that opened and closed itself)
- Run a Python script that would inject that CSV data as an HTML table into the file
- Run another Python script that would connect to the FTP server and upload the file. It would randomly pause and issue typos so it looked like the FTP session was being operated by a human at a keyboard so nobody thought anything on my plot.

I lived in fear of being found out, and told no one that the thing I was getting paid to do was no longer being done by me.

About 9 months later the department in question hired a full-time web developer for $45k/yr to bring their website in-house. I was costing them about $25/hr, probably skating under $2000/yr for my outsourced services. This was clearly not about money.

And what I feared did not happen. When I no longer had that work to sustain me my managers just put me on something else.

There’s always more work.

In my last years of undergraduate education and my first couple of years out of college I worked on projects that did some sort of Natural Language Processing tasks. For these we required training data, and the more the better.

On that, though, we had responsibilities. We had to make sure the data we had also came with some sort of license or implicit permission. You didn't just steal a pile of PDFs or scoop up a person's web site and put it in your training set. There were ethical constrains, and legal consequences. You acted above-board when training your AI models.

There were times we'd train models on Wikipedia dumps. They were always comparatively *amazing* results when we trained on good, large data like that. Cogent. Interesting. Even a simple Markov chain on Wikipedia looked smart.

When we wrote web crawlers, we wrote them to respect . We kept them on local domains. The field of the crawlers included our email address, and if an angry webmaster didn't like the way we were crawling them we'd fix it. Getting crawled aggressively at once taxed servers and spammed logs so we'd space it out to hours or days. If their was missing or malformed and they still didn't want us there, we'd block the site from crawling.

We made sure we had explicit permission to collect data for our training corpora.

The dot com boom was a crazy time. The internet has just become mainstream and there was a new gold rush. Money was there just for the taking, so many VC funded business plans were just "*traditional business X, but on the internet!*" and the money *flowed*. How it flowed.

Most of these companies, however, didn't really have a solid business model other than buying some servers and a domain name and "we'll put this thing on the internet."

Out of this crash came green shoots: Web 2.0, which used the web natively, organically, gave a good web-native experience. Eventually the dream of the internet, the promise of the hype, was made manifest after a lot of people learned a lot of really unnecessary, really painful lessons. They spent less and put their things on the internet because they made sense on the internet of the present, not because the internet was the next big thing.

The dream of the widespread, ubiquitous internet came true, and there were very few fatalities. Some businesses died, but it was more glacial than volcanic in time scale. When ubiquitous online services became commonplace it just felt mundane. It didn't feel forced. It was the opposite of the dot com boom just five years later: *the internet is here and we're here to build a solid business within it* in contrast with *we should put this solid business on the internet somehow, because it's coming*.

This is indeed a set of passive-aggressive jabs on the continuing assault on our senses by the LLM hype lobby.