In programming, Recursion is when a function calls itself, using its outputs as inputs to yield powerful results. It's definition contains a reference to itself.

Not only can recursion be useful, but it is also an entertaining concept to think about, as it touches on infinity and forces the observer to shift their way of thinking. Because of this, recursion is a long-running joke in computer science. Recursive names are one example: GNU stands for "GNU's Not Unix, and PHP stands for "PHP: Hypertext Preprocesser." If you Google recursion, Google asks, "Did you mean: recursion?"

Naming a font is hard, because the name must not clash with any existing fonts. If two fonts have matching names, they won't work well on the same computer – and there are tens of thousands of existing font families. What's more, as simple as a font name is, it has an outsized impact on the marketing success of a very complex project. Beyond all that, a font name should ideally present enough of the unique characters in the font it names to give viewers a sense of what the font is.

And so, early on in the process of this type project, I was faced with a common dilemma: what should I call it? All the good names seemed to be taken by existing fonts or by popular software projects. But then, I started wondering whether there might be some kind of recursive name I could come up with. "If only there were some sort of recursive name for this sort-of-cursive font for code," I thought.

"Oh hey, wait a minute…."

Recursive was a perfect name already. Yes, it was available. Yes, it used key letters from a monospace font, with the wide `r` and `i`. And yes, it hinted at a font that was for computers, but based on human writing. But most importantly, it said something meaningful about the project itself: Recursive was used to build itself.

That may sound strange at first, until you understand the process of type design. When a type designer works on a new design, they must "proof" it constantly – usually by printing out strings of text and marking aspects that need further work. For my project, I needed traditional proofing, but I also needed to understand how it worked (and how it needed improvement) in code editors and on the web.

Something that many people don't realize about modern-day type design is that it's just as much a development process as a design process (especially in new areas such as variable fonts). Complex font projects need Python scripts to facilitate design production, font building, and visual testing. Like any other form of code, this scripting is primarily done with monospace fonts in code editors.

Therefore, in order for me to understand how my font felt to use in actual practice, one of the best things I could do was to code with it. So, the output became the visual tool of input – recursion, in a sense particular to type design.