

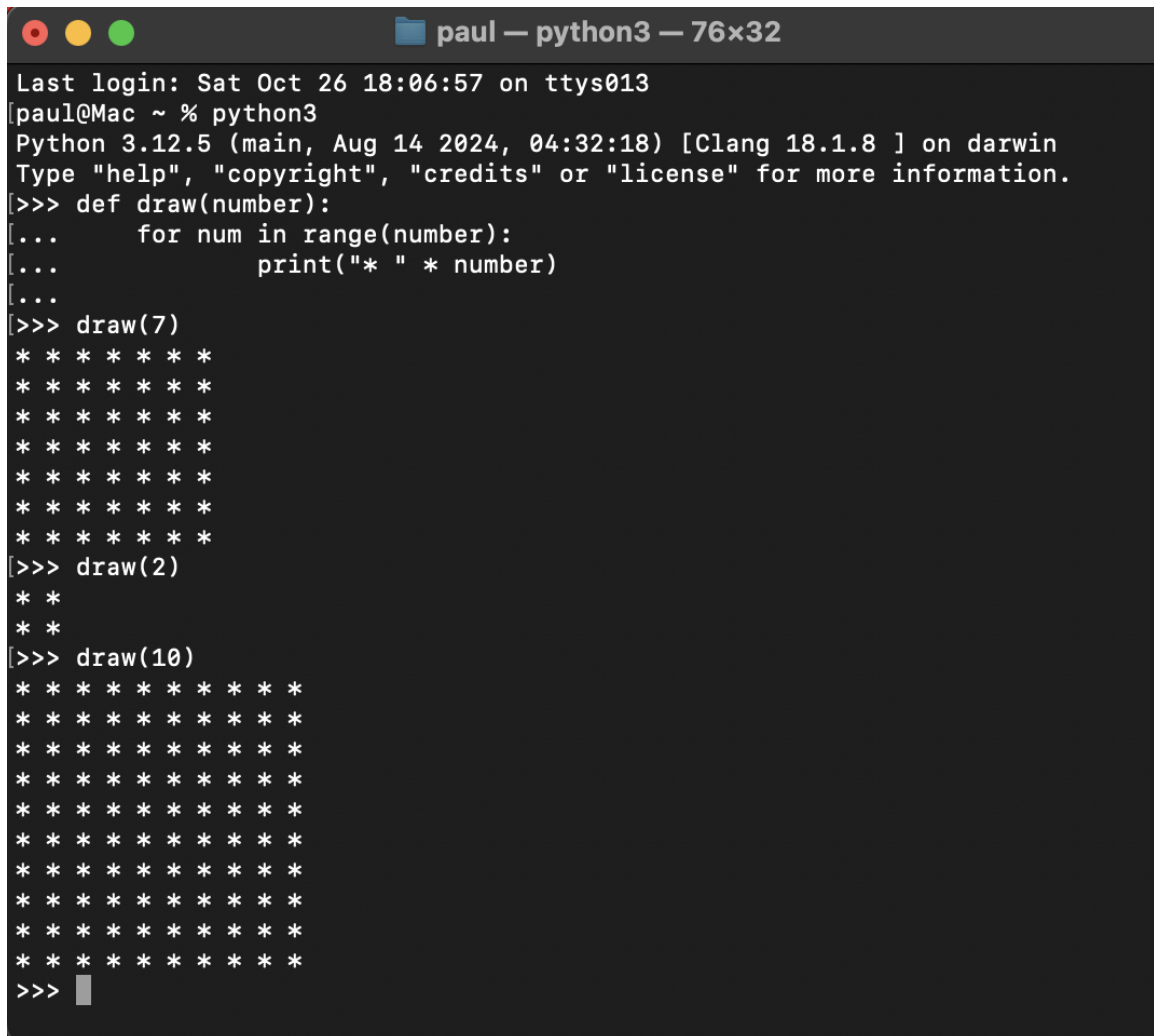
The function in the corresponding python file “def drawSquare” would not be able to draw other shapes, since it is only taking in a single argument which is being used for both the width and length. It takes in this argument and uses it inside a range() function to tell a for loop when to stop iterating, and for each iteration it will be able to print to the terminal a row of the square. Once the loop resolves, the result will be a square with the exact dimensions specified by the argument.

If other arguments were to be passed in then the function would be able to make different shapes. For example, a function could take in a “width” and a “height” argument which could draw a rectangle or even a triangle instead of a square. But as it stands, the function “def drawSquare” only takes in one argument, so its expected output would only be to draw a square.

It's an interesting conversation to have if the requirements were asking to draw different shapes depending on the amount of arguments passed. It would be more complex, not only because it would be dealing with more arguments when it came time to run the program, but also because there would need to be consideration if all of this could be done in a single function, or if the program would require different functions instead. In other words, would it be more correct to have a single “def draw” function and include all logic in that to draw whatever shape was necessary depending on the argument(s)? Or would it be better for example to separate logic for “def draw_square”, “def draw_triangle”, and “def draw_rectangle” as separate functions with each containing its own logic depending on arguments?

It seems like the latter is the more appropriate answer for Python programming. The idea of having one function being able to reproduce so many different answers is not always ideal – in fact, this seems to go against the functional programming paradigm (Thulie, n.d., para. 2). Additionally, being able to separate logic into different functions makes a program modular, and thus a program's code becomes “reusable so that we can call the same functions without rewriting them” (Tagliaferri, 2021 August 20, para. 13).

Finally, to answer the question: “Could you make a draw square method on the terminal scribe class itself?” I believe what this is asking is if it's possible to make a draw square method on the terminal using python3, to which the answer would be yes:



```
paul — python3 — 76x32
Last login: Sat Oct 26 18:06:57 on ttys013
paul@Mac ~ % python3
Python 3.12.5 (main, Aug 14 2024, 04:32:18) [Clang 18.1.8 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> def draw(number):
...     for num in range(number):
...         print("* " * number)
...
>>> draw(7)
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
>>> draw(2)
* *
* *
>>> draw(10)
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
>>>
```

Sublime HQ Pty Ltd. (n.d.). *Sublime Text*. <https://www.sublimetext.com>

Tagliaferri, L. (2021, August 20). *How to Define Functions in Python 3*. DigitalOcean. <https://www.digitalocean.com/community/tutorials/how-to-define-functions-in-python-3>

Thulie. (n.d.). *Introduction to Functional Programming*. Turing. <https://www.turing.com/kb/introduction-to-functional-programming>