Pattern Drawing in Python using Functions and OOP

Project Description:

This project creates a customizable square pattern using both a standalone function and an object-oriented class structure. It includes input validation, loop-based pattern generation, and a modular design that allows easy extension to other shapes. The project demonstrates proficiency in procedural programming, OOP concepts, and clean code practices.

Introduction

In this project, I used Python to write a simple function that prints a square made of * characters, based on the size provided by the user. The function takes one argument—the size—and then draws a square with equal width and height. This Midterm Project is a great way to practice what I have learned in Modules 1 to 6. I reviewed how to define and call functions, work with loops, and even looked into some object-oriented programming ideas, although that part was optional.

Objective

The objective of this assignment was:

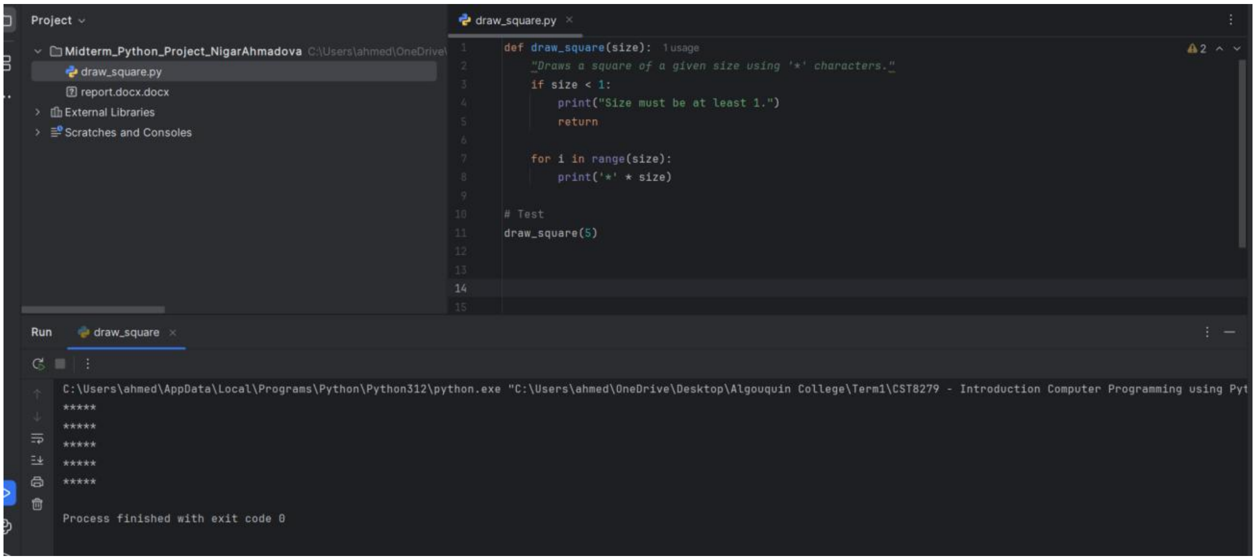
- To design a Python function that prints a square of a user-defined size.
- To demonstrate advanced understanding by creating a class that includes this functionality.
- To practice writing clear, structured Python code using best practices.

Implementation and Explanation

Function-Based Version

The first part of the solution uses a simple function draw_square(size), which uses a for loop to print a square. The * character is repeated based on the input size. If the size is invalid (less than 1), the function prints a message.

```
def draw_square(size):
    if size < 1:
        print("Size must be at least 1.")
        return
    for i in range(size):
        print('*' * size)
```



```
def draw_square(size):
    "Draws a square of a given size using '*' characters."
    if size < 1:
        print("Size must be at least 1.")
        return

    for i in range(size):
        print('*' * size)

# Test
draw_square(5)
```

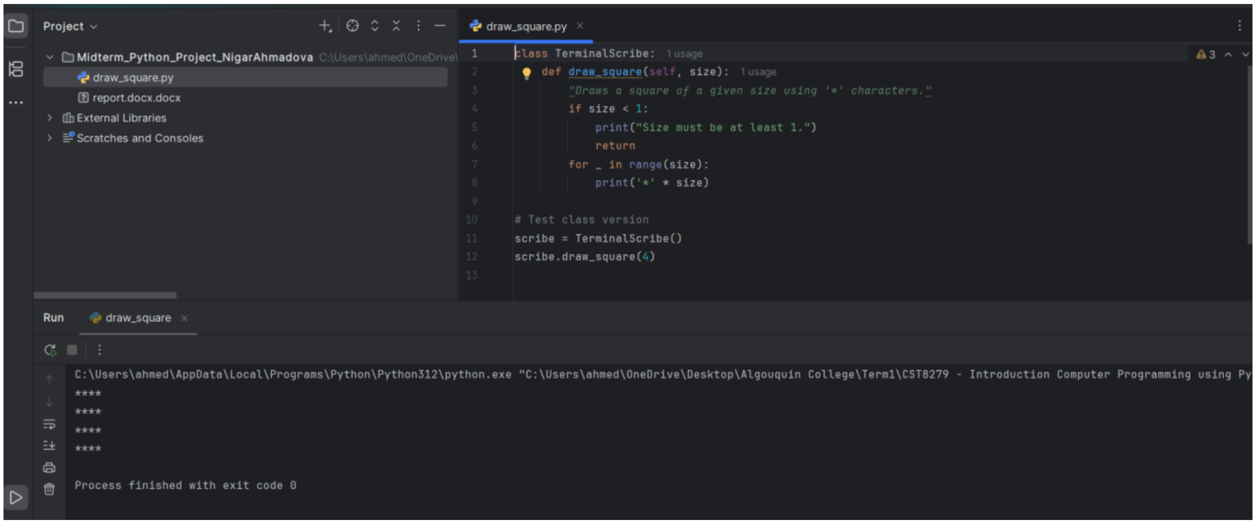
```
C:\Users\ahmed\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\ahmed\OneDrive\Desktop\Algouquin College\Term1\CST8279 - Introduction Computer Programming using Pyt
*****
*****
*****
*****
*****

Process finished with exit code 0
```

Class-Based Version

To explore object-oriented programming, I created a class called TerminalScribe with a method draw_square. The logic is similar but now wrapped in a reusable structure.

```
class TerminalScribe:
    def draw_square(self, size):
        if size < 1:
            print("Size must be at least 1.")
            return
        for i in range(size):
            print('*' * size)
```



```
class TerminalScribe:
    def draw_square(self, size):
        "Draws a square of a given size using '*' characters."
        if size < 1:
            print("Size must be at least 1.")
            return
        for _ in range(size):
            print('*' * size)

# Test class version
scribe = TerminalScribe()
scribe.draw_square(4)
```

```
C:\Users\ahmed\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\ahmed\OneDrive\Desktop\Algouquin College\Term1\CST8279 - Introduction Computer Programming using Py
****
****
****
****

Process finished with exit code 0
```

Answers to the Questions:

Could you draw other shapes?

Yes, I can draw other shapes like rectangles, triangles, or even hollow squares using nested loops and conditional statements logic. For example, drawing a hollow square requires checking if the current row is the first or last or if the column is at the edges, to determine where to place the * characters.

Could you make a draw square method on the terminal scribe class itself?

Yes, I implemented a class called TerminalScribe, which contains a method called draw_square(self, size). This method works the same way as the standalone function, but is wrapped inside a class. It is a good way to apply object-oriented programming, and I like that it helps keep things organized. I can easily add more methods later if I want to draw other shapes too.

What I Learned

- **Function definition and parameter usage:** I learned how to define reusable functions and pass arguments to control their behaviour.
- **Loop structures in Python (for):** I used *for* loops to repeat code and build the square pattern line by line.
- **Input validation:** I included checks to make sure the user input is valid before running the drawing logic.
- **The basics of class creation and method definitions:** I explored object-oriented programming by creating a class with its own drawing method.
- **Reusability and modular design principles:** I realized how organizing code into functions and classes makes it easier to expand and maintain.

References

Python Software Foundation. (2023). *The Python Tutorial*. https://docs.python.org/3/tutorial/

Real Python. (n.d.). *Defining Your Own Python Function*. https://realpython.com/defining-your-own-python-function/

Purdue OWL. (n.d.). *APA Formatting and Style Guide*. https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html