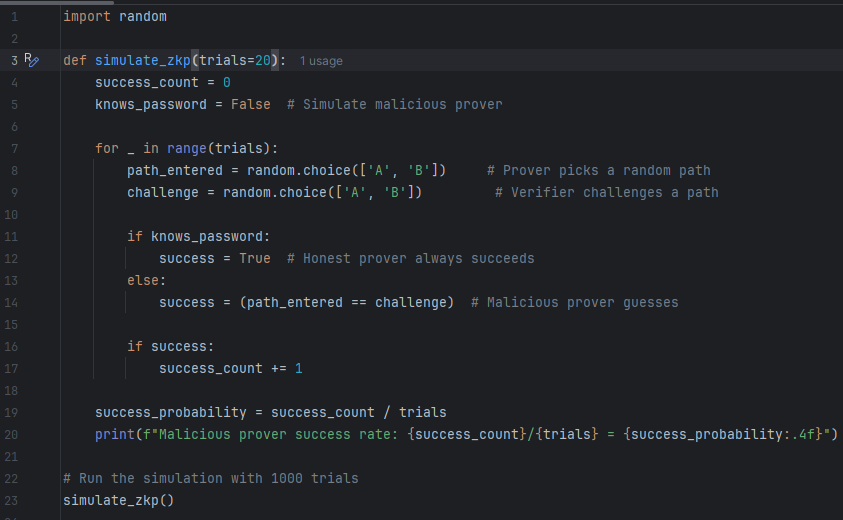
**Week 7 Lab**

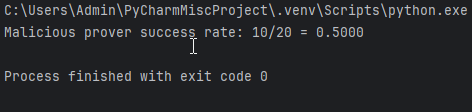
**Part 1: Zero-Knowledge Proof (ZKP) Simulation**

**Task: Modify the simulation to calculate success probability for a malicious prover.**

This Python script simulates a malicious prover attempting to cheat in a Zero-Knowledge Proof (ZKP) scenario, specifically modeled after the Ali Baba Cave protocol. It runs the experiment over 20 trials and calculates the probability of success when the prover is guessing randomly, without knowing the secret.

****

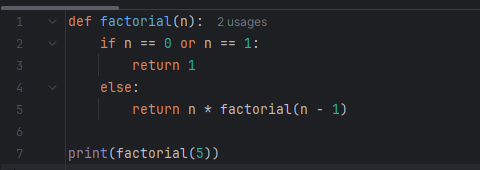
As expected, the malicious prover succeeds in approximately 50% of the trials by guessing. This highlights that while guessing might work in a single round, repeating the challenge many times makes it statistically improbable for an attacker to cheat successfully - a core strength of Zero-Knowledge Proof protocols.

****

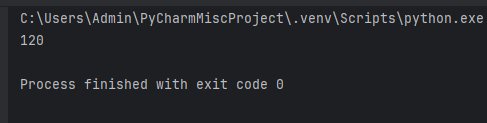
**Part 2: Code Obfuscation Challenge**

**1. Create a simple function (e.g., factorial or Fibonacci).**

The following Python function calculates the factorial of a number using recursion.



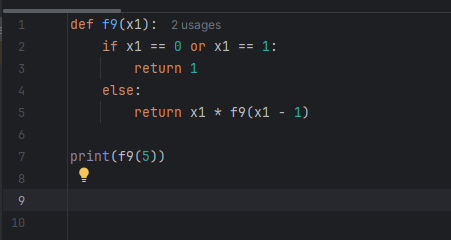
The output confirms that the function correctly calculates the factorial of 5



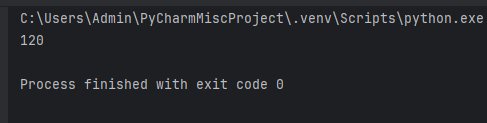
**2. Apply manual obfuscation:**

**○ Rename variables to meaningless values.**

This is lexical (name-based) obfuscation, a manual method where meaningful identifiers like factorial or n are replaced with arbitrary names like f9 or x1. While the functionality remains the same, readability and maintainability are significantly reduced, which helps deter reverse engineering.

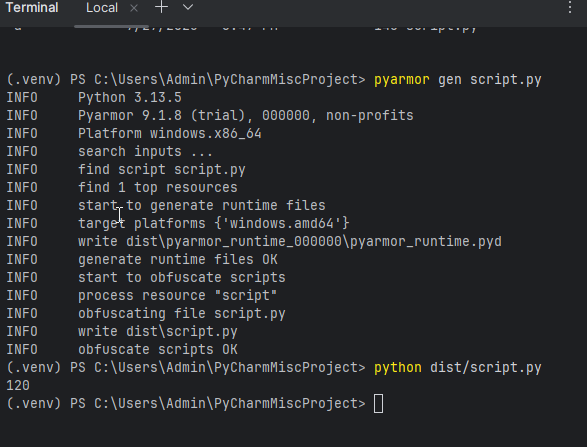


Output confirms that even though you manually renamed the function and variable to meaningless names (like f9 and x1), the underlying logic of the program still works as intended.



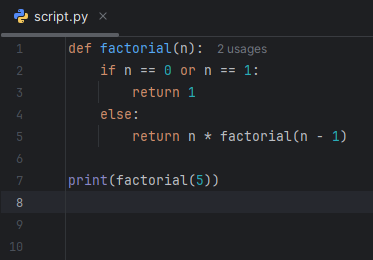
**3. Apply automatic obfuscation use any tool**

I used PyArmor for obfuscation and this output proves the obfuscated script retains the original functionality.

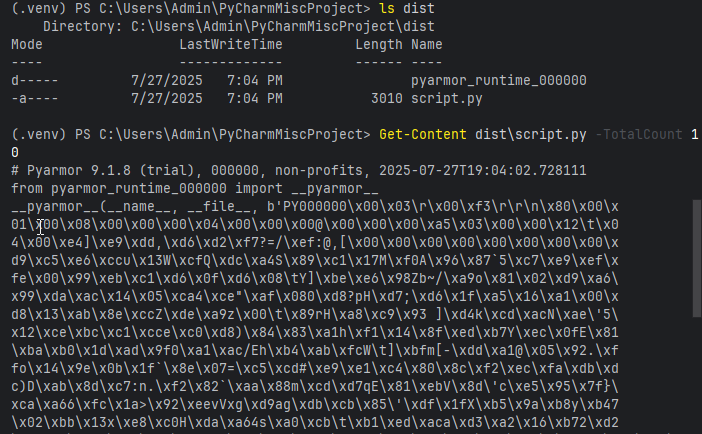


**4. Share both original and obfuscated code.**

**This is an original code:**

****

**This is an obfuscated code:**



**Type of Obfuscation Used and Why**

I used automatic code obfuscation with the tool PyArmor. This type of obfuscation transforms the original Python source code into an encrypted and encoded form that is difficult to read or reverse engineer, while preserving the program’s functionality.