

 SHA-256 in Action – Cryptographic Hashing

**Objective/Aim:**  
  
 To use an online SHA-256 hash generator to practically understand its core cryptographic properties. The goal is to observe how the algorithm transforms any input into a unique, fixed-length output and to witness the avalanche effect, where a tiny change in the input drastically alters the resulting hash.

**Apparatus/Software Used:**

* Laptop/PC
* PowerPoint/Word for documentation
* The online SHA-256 generator tool: <https://emn178.github.io/online-tools/sha256.html>

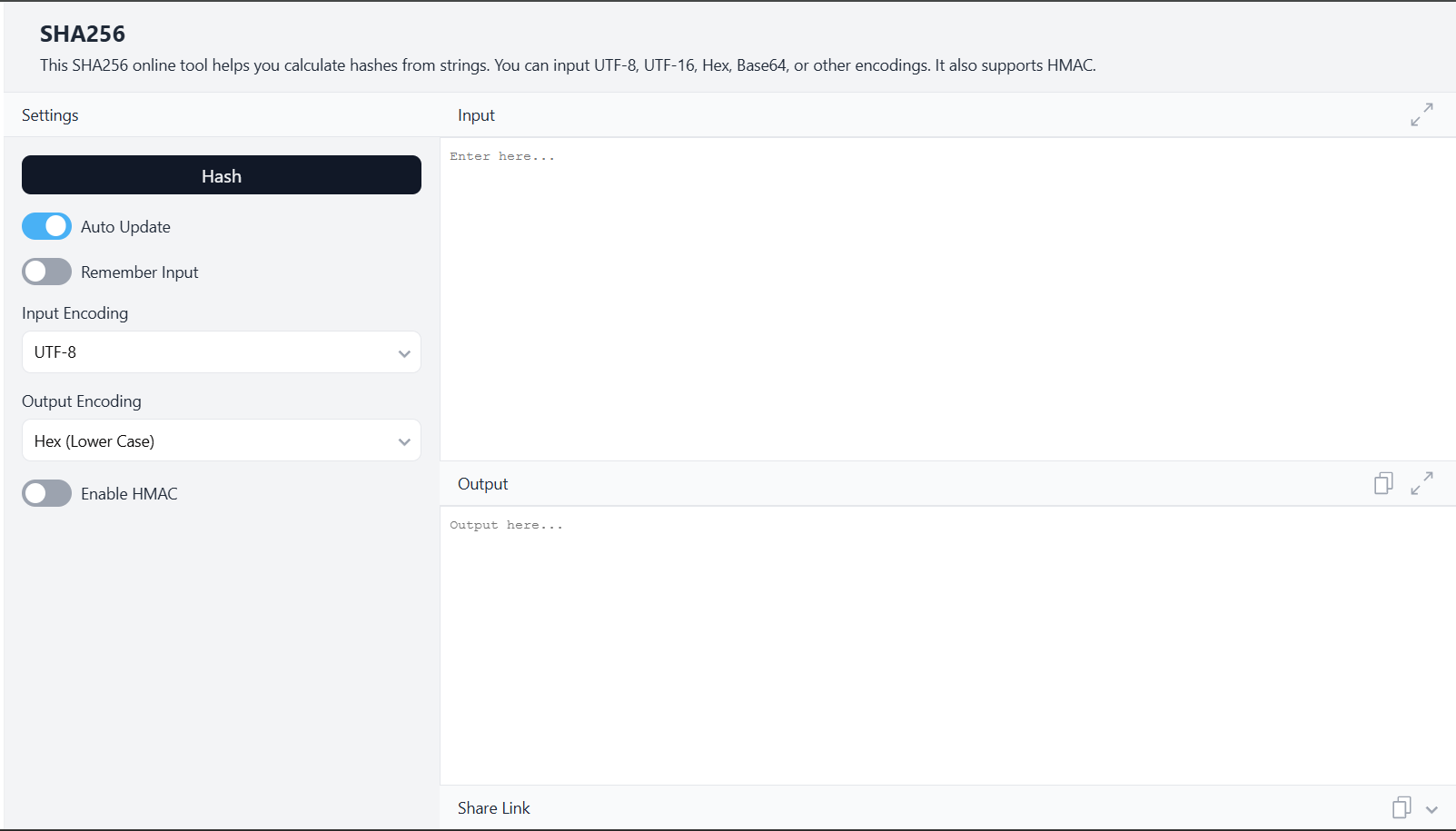
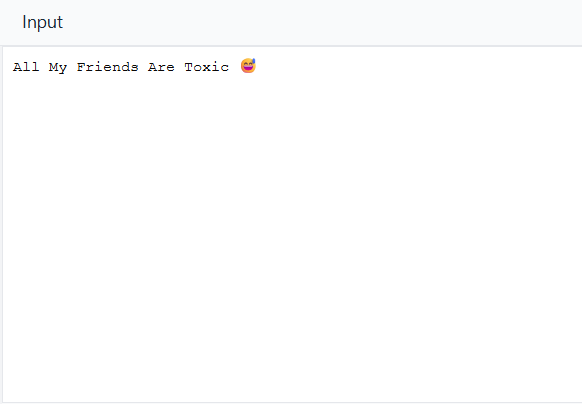
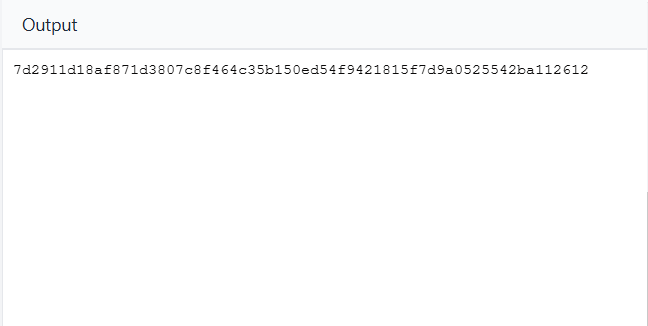
**Theory/Concept:**

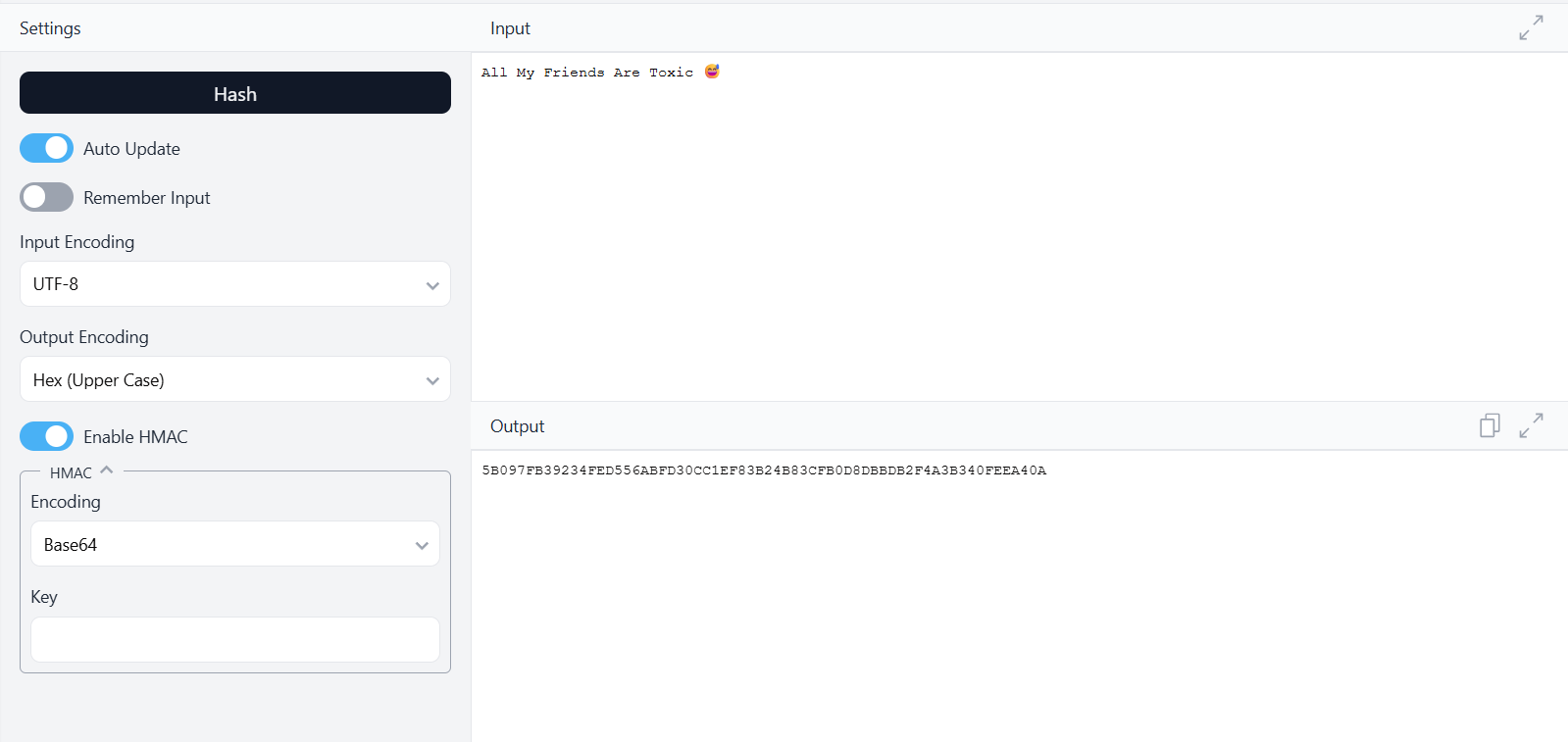
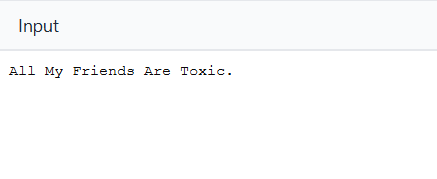
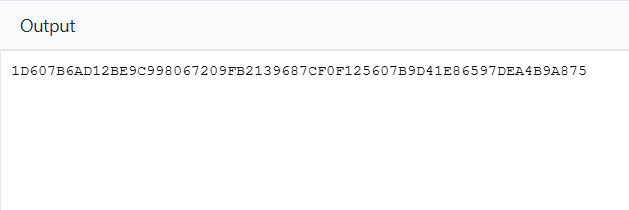
SHA-256 is a cryptographic hash function that acts like a digital fingerprint generator. It takes any input—text, a file, anything—and produces a unique, 256-bit (64-character) string. This process is deterministic (the same input always yields the same output) and one-way, making it practically impossible to reverse-engineer the original input from the hash. Its most crucial feature is the "avalanche effect": even the smallest change to the input creates a completely different hash, which is fundamental for ensuring data integrity.



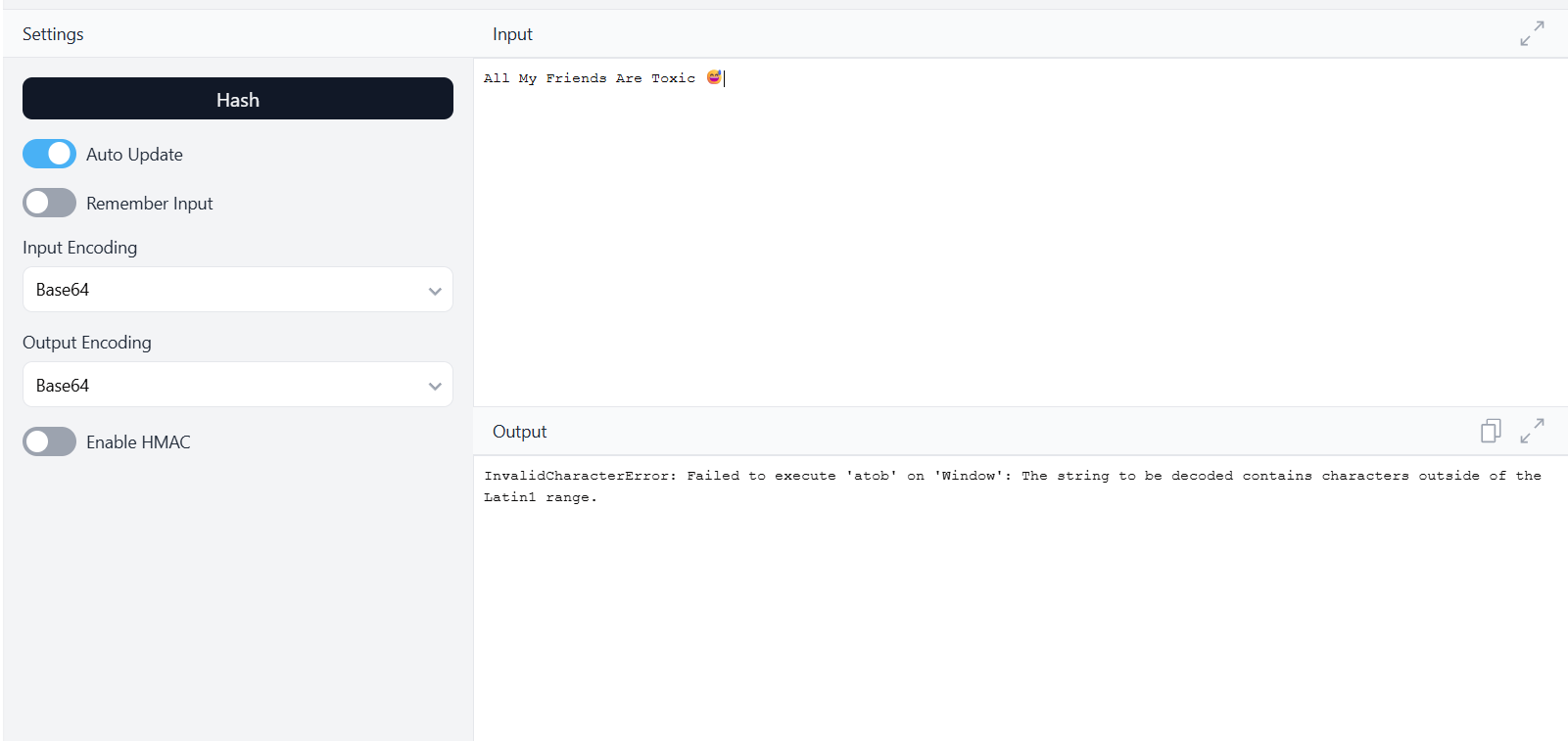
**Procedure:**

Navigate to the online SHA-256 tool, which provides a simple interface for hashing strings with various settings.



Enter the text "All My Friends Are Toxic" into the input field to generate the initial SHA-256 hash.

Add a single period to the end of the input text and observe that the output hash changes completely, demonstrating the avalanche effect.

 Attempt to use Base64 as the *input* encoding for text containing an emoji, which results in a character range error, showing encoding limitations.

Enable the HMAC option with a Base64 key to generate a keyed hash, a method used to verify both data integrity and authenticity.

**Observation Table:**

It was observed that the SHA-256 hash function consistently produces a 64-character hexadecimal string, regardless of the input's size. The output is deterministic, as identical inputs always resulted in the same hash. Most notably, altering even a single character in the input caused a complete and unpredictable change in the output hash, clearly demonstrating the avalanche effect.