**RANDOM PASSWORD GENERATOR**

1. **bufio — Buffered I/O**

* **Purpose**: Helps read input efficiently.
* **Why it's used here**: You're using bufio.NewScanner(os.Stdin) to read user input from the terminal.

Think of bufio.Scanner as a smart way to read user input, line by line.

1. **What is os.Exit()?**

os.Exit() is a function from the os package that tells your program to **stop running immediately** and exit.

**🔢 What does the number inside os.Exit() mean?**

The number you pass in is called an **exit code** (or **status code**). It tells the operating system (or whoever ran the program) **why** the program ended.

**✅ Common Exit Codes**

| **Code** | **Meaning** |
| --- | --- |
| 0 | ✅ Success – program ran fine |
| 1 | ❌ Error – something went wrong |
| Other numbers | Also indicate errors or specific reasons (custom) |
| 1. **What is bufio.NewScanner(os.Stdin)?**   This gives you a **scanner object** that:   * Reads **entire lines of input** (not just single words). * Is **more flexible** and **cleaner** for line-by-line input. * Returns input as a string with scanner.Text().   **✅ Great for:**   * Reading full sentences * Taking clean user input without surprises * Input parsing where you want to control everything after reading |  |

1. **What Is rand.Seed(...)?**

In Go, the math/rand package is used to generate **random numbers**.

BUT — by default, it always gives you the **same sequence** every time you run your program. That's not truly random.

🧠 To make it unpredictable (different every time), you must **seed** it.

That’s where rand.Seed(...) comes in. It **initializes** the random number generator with a starting number (called a *seed*).

1. **passwordRune := []rune(password)**

This line converts the password string into a **slice of runes** (which are like characters in Go).

* password is a **string**.
* A **rune** is essentially a **Unicode character**. When you convert a string to a slice of runes ([]rune), you’re treating each character as a distinct item that you can manipulate.

**Why convert to rune?**

* In Go, strings are **immutable** (you can't modify them directly). To modify the individual characters, we convert the string into a **slice of runes**. A slice is a mutable sequence of elements, so we can swap characters in it.
* Strings can contain **multi-byte characters** (like emojis or non-English letters). By using rune, we handle characters properly, even if they are more than one byte.

1. **rand.Shuffle(len(passwordRune), func(i, j int) { ... })**

This is where the **shuffling** happens!

* **rand.Shuffle(n, swapFunc)** is a function that **shuffles** the elements of a slice randomly.
  + n is the length of the slice you want to shuffle.
  + swapFunc(i, j) is a function that tells Go how to swap two elements at the indices i and j.

**Key points about rand.Shuffle:**

* It **randomly swaps** pairs of elements in the slice to shuffle it.
* The **swapFunc** is called multiple times to shuffle the slice