Swift Data Types

In Swift, data types are essential for defining the nature of data and ensuring type safety. Swift provides a range of built-in data types, which can be categorized into several groups: basic data types, collection types, and optional types, among others.

Basic Data Types

Int: Represents integer values.

let age: Int = 25

——

Float: Represents 32-bit floating-point numbers.

let floatNumber: Float = 3.14

——

Double: Represents 64-bit floating-point numbers. It is preferred over **Float** for greater precision.

let pi: Double = 3.14159265358979

——

Bool: Represents boolean values, which can be either **true** or **false**.

let isSwiftAwesome: Bool = true

——-

String: Represents a sequence of characters.

let greeting: String = "Hello, Swift!"

——-

Character: Represents a single character.

let letter: Character = "A"

———————

Swift Properties

In Swift, properties are values associated with a class, struct, or enum. They can be variables or constants and can store data or compute values dynamically. There are several types of properties in Swift, including stored properties, computed properties

Stored Properties: These are properties that store a value as part of an instance of a class or struct. They can be either variables (with the **var** keyword) or constants (with the **let** keyword).

Example:

```
struct Person {
    var name: String      // Stored property
    let birthYear: Int    // Stored property
}
```

Computed Properties: These properties do not store a value directly. Instead, they provide a getter (and optionally a setter) to compute a value when accessed.

Example:

```
struct Rectangle {
    var width: Double
    var height: Double

    var area: Double {    // Computed property
        return width * height
    }
}
```

Difference Between var and let

var: This keyword declares a variable property, meaning its value can be changed after it is initially set.

```
var age = 25
age = 26    // This is allowed
```

let: This keyword declares a constant property, meaning its value cannot be changed once it is set.

```
let birthYear = 1990
birthYear = 1991    // This will cause an error
```