Page No. TARANG
Date : / /

Variables

Variables are Containers for storing data values.

To Create a variable, use var, val and assign a value to it with the equal sign (=)

Syntax-

```
var variablename = value
val variable_name = value
```

Example:-

```
var name = "krishna";
val marks = 615

println(name)
println(marks)
```

↳ print the value of name

↳ print the value of marks.

Difference Between val and var,

val and var is that

variable declared with the var keyword can be changed/ modified. while val variable cannot.

Example:

using var

```
fun main()
{
    var name = "krishna";
    println(name);
    name = "Radha";
    println( name);
}
```

using val

```
fun main()
{
    val name = "krishna"
    println(name)
    name = "Radha";
    println( name)
}
```

can not modified name, it declare with val

Rules to define variable name

Names can contain letters, digits, underscore and dollar signs.

Names should be start with letter

Names can also belong $ and (_)

Names are case sensitive

It should start with a lowercase letter and it cannot contain whitespace.

Cannot used Reserved words.

Data Types

Kotlin is statically typed programming language, which means that variable types are known at Compile time. Kotlin provides a rich set of data types to work with.

Common data Types in Kotlin:

1. Numbers:

- Int: Represents 32-bit signed Integers
- Long: Represent 64 bit Signed Integers
- Short: Represent 16 bit signed Integers.
- Byte: Represent 8 bit signed Integers.
- Double: Represents 64 bit double precision floating point numbers.
- Float: Represents 32 bit single precision floating-point numbers.

2. Characters:

'Char': Represents a single 16-bit Unicode character.

3. Booleans:

Boolean: Represents as true or false value.

4. Textual Data

String: Represents a sequence of characters.

5. Arrays:

Arrays in Kotlin are represented using the 'Array' class, which can hold Elements of a specific type.

6. Collections:

Kotlin provides various collection types like 'List', 'Set' and 'Map' for working group of data.

7. Ranges:

'CloseRanges' and 'HalfRanges' are used to represent ranges of value.

8. Nullable Types

In Kotlin, you can make any type nullable by adding '?' to the type declaration. For Example Int? represents an Int that can be 'null'.

9. User-defined data Types

You can defined your own data Type using classes and data classes. These allow you to encapsulate data and behaviours into custom types.

10. Enums

Enums are used to represent a set of constant values. They are defined using the 'enum' keyword.

11. Types Aliases:

You can create type aliases using the 'typealias' keyword to given a more descriptive name to an

existing type.

Example:-

```
val age: Int = 30
val name: String = "Kashyap"
val Do
val weight: Double = 175.5
val isStudent: Boolean = true
val grades: List<Int> = listOf(85, 90, 100)
val temp: Float? = null
```