F# - VARIABLES

A variable is a name given to a storage area that our programs can manipulate. Each variable has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

Variable Declaration in F#

The **let** keyword is used for variable declaration −

For example,

```
let x = 10
```

It declares a variable x and assigns the value 10 to it.

You can also assign an expression to a variable −

```
let x = 10
let y = 20
let z = x + y
```

The following example illustrates the concept −

Example

```
let x = 10
let y = 20
let z = x + y

printfn "x: %i" x
printfn "y: %i" y
printfn "z: %i" z
```

When you compile and execute the program, it yields the following output −

```
x: 10
y: 20
z: 30
```

Variables in F# are **immutable,** which means once a variable is bound to a value, it can't be changed. They are actually compiled as static read-only properties.

The following example demonstrates this.

Example

```
let x = 10
let y = 20
let z = x + y

printfn "x: %i" x
printfn "y: %i" y
printfn "z: %i" z

let x = 15
let y = 20
let z = x + y

printfn "x: %i" x
```

```
printfn "y: %i" y
printfn "z: %i" z
```

When you compile and execute the program, it shows the following error message −

```
Duplicate definition of value 'x'
Duplicate definition of value 'Y'
Duplicate definition of value 'Z'
```

Variable Definition With Type Declaration

A variable definition tells the compiler where and how much storage for the variable should be created. A variable definition may specify a data type and contains a list of one or more variables of that type as shown in the following example.

Example

```
let x:int32 = 10
let y:int32 = 20
let z:int32 = x + y

printfn "x: %d" x
printfn "y: %d" y
printfn "z: %d" z

let p:float = 15.99
let q:float = 20.78
let r:float = p + q

printfn "p: %g" p
printfn "q: %g" q
printfn "r: %g" r
```

When you compile and execute the program, it shows the following error message −

```
x: 10
y: 20
z: 30
p: 15.99
q: 20.78
r: 36.77
```

Mutable Variables

At times you need to change the values stored in a variable. To specify that there could be a change in the value of a declared and assigned variable, in later part of a program, F# provides the **mutable** keyword. You can declare and assign mutable variables using this keyword, whose values you will change.

The **mutable** keyword allows you to declare and assign values in a mutable variable.

You can assign some initial value to a mutable variable using the **let** keyword. However, to assign new subsequent value to it, you need to use the ← operator.

For example,

```
let mutable x = 10
x ← 15
```

The following example will clear the concept −

Example

```
let mutable x = 10
```

```
let y = 20
let mutable z = x + y

printfn "Original Values:"
printfn "x: %i" x
printfn "y: %i" y
printfn "z: %i" z

printfn "Let us change the value of x"
printfn "Value of z will change too."

x <- 15
z <- x + y

printfn "New Values:"
printfn "x: %i" x
printfn "y: %i" y
printfn "z: %i" z
```

When you compile and execute the program, it yields the following output −

```
Original Values:
x: 10
y: 20
z: 30
Let us change the value of x
Value of z will change too.
New Values:
x: 15
y: 20
z: 35
```