2.0 INTRODUCTION

VisualBasic is a programming language and any application that is developed using Visual Basic can be made very powerful by using code. Coding a program is simple enough with an understanding of certain fundamental concepts of programming like the role of variables, use of programming constructs for decisions and looping. This unit addresses all these concepts.

2.1 OBJECTIVES

At the end of this unit you will be able to,

- Understand the meaning of variables
- Declare and use variables
- Understand and use various programming constructs like decision-making and looping
- Understand the use of modules
- Understand the use of procedures
- Understand the use of functions

2.2 VARIABLES

We often need to store values temporarily when performing calculations with Visual Basic. For example, we might want to calculate several values, compare them and perform different operations on them, depending on the result of the comparison. We need to retain the values if we want to compare them but not store them in a property.

VB, like most programming languages uses variables for storing values. Variables have a name and data type.

2.2.1 Declaring Variables

We do not have to explicitly declare or create variables in VB. If we use a name in our code that is not the name of any existing variable, VB creates a new variable with that name. However, it is good programming practice to explicitly declare our variables because it might save debugging time later. We can specify that variable declaration as required by adding an Option Explicit statement to the declaration section of each form, standard or class module. This will force us to declare the variables.

We declare a variable using Dim statement: Dim variable [As type]

Variable names follow the same rule as everything else in VB. For example, we create variable called "Result" with following code -

Dim Result

A variable declared with the Dim statement within a procedure exists only as long as the procedure is executing. When the procedure finishes, the value of the variable disappears. In addition, the value of the variable in the procedure is local to the procedure. i.e. we cannot access a variable of one procedure from another procedure. These characteristics allow us to use the same variable names in different procedures without worrying about conflicts or accidental changes.

There are other ways to declare variables:

- Declaring variable in the declaration section of a form, standard or class module, rather than within the procedure makes the variable available to all the procedures in the module.
- Declaring a variable using the Public keyword makes it available throughout our application.
- Declaring the local variable using the Static keyword preserves its value even when a procedure ends.

2.2.2 Storing and Retrieving Data in Variables

We use assignment statements to perform calculations and assign the result to a variable.

```
x = 10          ' The value 10 is passed to the variable x.
x = x+1         ' The variable is incremented.
```

To display the result,

```
Text1.text=x
```

2.2.3 Variable Data Types

By default VB variables are of the Variant data type. The variant data type can store arrays and objects, numeric, date/time or string data. We need not convert between these types of data when assigning them to a variant variable; VB automatically performs the necessary conversion. If we know that a variable will always store data of a particular type, however, VB can handle that data more efficiently, if we declare a variable of that type.

E.g.
1. Dim x as Integer
2. Public s as String
3. Static flag as Boolean

2.2.4 Scope and Life Time of Variables

When we declare a variable within a procedure, only code within that procedure can access or change the value of that variable. It has a scope that is local to that procedure. Sometimes, however we need to use a variable with a broader scope, such as one whose value is available to all the procedures, within the same module, or even to all procedures in our entire application, VB allows us to specify the scope of a variable when we declare it.

Scoping Variables

Depending on how it is declared, a variable is scoped as either a procedure level (local) or module level variable.

Scope	Private	Public
Procedure level	All identifiers are private to the procedure in which they appear	Not Applicable. We cannot declare public variables within a procedure.
Module level	Identifiers are private to the module in which they appear	Identifiers are available to all modules.

Variables Used Within a Procedure

Variables used within a procedure are known as local variables. We declare them with keyword, Dim or keyword Static. Local variables declared with static, exist the entire time your application is running. Local variables declared with Dim, exist only as long as the procedure is executing. Local variables are declared for any type of temporary calculation.

Variables Used Within a Module

By default, a module level variable is available to all the procedures in that module, but not to code in other modules. We create module level variables by declaring them with Public or Private in the declaration section at the top of the module.

At the module level there is no difference between Private and Dim. But, Private is preferred because, it readily contrasts with Public and makes our code easier to read.

Variables Used by all Modules

To make a module level variable available to other modules, we use the Public keyword while declaring. The values in public variables are available to all procedures in our application. Like all module level variables, Public variables are declared in the Declaration section at the top of the module.

2.3 DATA TYPES

All variables have a data type that determines what kind of data they can store. By default, if we do not supply a data type, the variable is given the variant data type.

Fundamental VB Data Types

Type Name	Storage size	Range
Integer	2 Bytes	-32768 to 32767
Long	4 Bytes	-2147483648 to 2147483647
Single	4 Bytes	-3.02823E38 to -1.401298E-45 (Negative-values) 1.401298E-45 to 3.402823E38 (Positive values)
Double	8 Bytes	-1.79769313486231E308 to -4.94065645841247E-324 (Negative-values) 4.94065645841247E-324 to 1.79769313486231E308 (Positive Values)
Currency	8 Bytes	-922337203685477.5808 to

		922337203685477.5807
String	1 Byte per character	0 to approximately 65500 characters 0 to 2E32 on 32 bit systems
Byte	1 Byte	0 to 255
Boolean	2 Bytes	True or False
Date	8 Bytes	January 1, 100 to December 31, 9999
Object	4 Bytes	Any object reference
Variant	16 Bytes + 1 Byte for each character	NULL, ERROR, Any numeric value upto the range of double or any character text, object or array

2.4 MODULES

Code in VB is stored in modules. There are 3 kinds of modules, Class modules, Form modules (Form module is a type of Class module) and Standard modules.

Simple applications can consist of just a single Form and all of the code in the application resides in that Form module. As our applications get larger and more sophisticated, we add additional Forms. Eventually, we might find that there are common codes we want to execute in several forms. We do not want to duplicate the code in all Forms, we create a separate module containing the procedure that implements the common code. This separate module should be a Standard module.

Each Standard, Class and Form module can contain:

- Declarations: We can place constant, type, variable and DLL procedure declarations at the module level of Form, Class or Standard modules.