1. Numeric Data Types

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and more. Examples of numeric data types are examination marks, height, weight, the number of students in a class, share values, price of goods, monthly bills, fees and others. In Visual Basic, numeric data are divided into 7 types, depending on the range of values they can store.

Numeric Data Types

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).

2. Non-numeric Data Types

Nonnumeric data types are data that cannot be manipulated mathematically using standard arithmetic operators. The non-numeric data comprises text or string data types, the Date data types, the Boolean data types that store only two values (true or false), Object data type and Variant data type .They are summarized in Table 5.2

Nonnumeric Data Types

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant(numeric)	16 bytes	Any value as large as Double
Variant(text)	Length+22 bytes	Same as variable-length string

Variables

In term of VB, variables are areas allocated by the computer memory to hold data. To name a variable in Visual Basic, you have to follow a set of rules.

5.2.1 Variable Names

The following are the rules when naming the variables in Visual Basic

- It must be less than 255 characters
- No spacing is allowed
- It must not begin with a number
- Period is not permitted

Examples of valid and invalid variable names are displayed in Table 5.4

Table 5.4

Valid Name	Invalid Name
My_Car	My.Car
ThisYear	1NewBoy
Long_Name_Can_beUSE	He&HisFather *& is not acceptable

5.2.2 Declaring Variables

In Visual Basic, to declare the variables before using them by assigning names and data types. They are normally declared in the general section of the codes' windows using the **Dim** statement.
The format is as follows:

Dim *Variable Name* As *Data Type*

Example 5.1

```
Dim password As String
Dim yourName As String
Dim firstnum As Integer
Dim secondnum As Integer
Dim total As Integer
Dim doDate As Date
```

You may also combine them in one line , separating each variable with a comma, as follows:

Dim password As String, yourName As String, firstnum As Integer,.....

If data type is not specified, VB will automatically declare the variable as a Variant.
For string declaration, there are two possible formats, one for the variable-length string and another for the fixed-length string. For the variable-length string, just use the same format as example 5.1 above. However, for the fixed-length string, you have to use the format as shown below:

*Dim VariableName as String * n*, where n defines the number of characters the string can hold.

Example 5.2:

Dim yourName as String * 10

yourName can holds no more than 10 Characters.

5.3 Constants

Constants are different from variables in the sense that their values do not change during the running of the program.

5.3.1 Declaring a Constant

The format to declare a constant is

Const *Constant Name* As *Data Type = Value*

Example 5.3

Const Pi As Single=3.142

Const Temp As Single=37

Const Score As Single=100

6.1 Assigning Values to Variables

After declaring various variables using the Dim statements, we can assign values to those variables. The general format of an assignment is

Variable=Expression

The variable can be a declared variable or a control property value. The expression could be a mathematical expression, a number, a string, a Boolean value (true or false) and more. The following are some examples:

```
firstNumber=100
secondNumber=firstNumber-99
userName="John Lyan"
userpass.Text = password
Label1.Visible = True
Command1.Visible = false
Label4.Caption = textbox1.Text
ThirdNumber = Val(usernum1.Text)
total = firstNumber + secondNumber+ThirdNumber
```

6.2 Operators in Visual Basic

To compute inputs from users and to generate results, we need to use various mathematical operators. In Visual Basic, except for + and -, the symbols for the operators are different from normal mathematical operators, as shown in Table 6.1.

Table 6.1: Arithmetic Operators

Operator	Mathematical function	Example
^	Exponential	2^4=16
*	Multiplication	4*3=12, (5*6))2=60
/	Division	12/4=3
Mod	Modulus(return the remainder from an integer division)	15 Mod 4=3 255 mod 10=5
\	Integer Division(discards the decimal places)	19\4=4
+ or &	String concatenation	"Visual"&"Basic"="Visual Basic"

Example

```
Dim firstName As String
Dim secondName As String
Dim yourName As String

 Private Sub Command1_Click()
        firstName = Text1.Text
        secondName = Text2.Text
        yourName = secondName
        + "  " + firstName
        Label1.Caption =
yourName
End Sub
```

In this example, three variables are declared as string. For variables firstName and secondName will receive their data from the user's input into textbox1 and textbox2, and the variable yourName will be assigned the data by combining the first two variables. Finally, yourName is displayed on Label1.

Example

```
Dim number1, number2, number3 as Integer
Dim total, average as variant
Private sub Form_Click
        number1=val(Text1.Text)
        number2=val(Text2.Text)
        number3= val(Text3.Text)
        Total=number1+number2+number3
        Average=Total/5
        Label1.Caption=Total
        Label2.Caption=Average
End Sub
```

In the example above, three variables are declared as integer and two variables are declared as variant. Variant means the variable can hold any data type. The program computes the total and average of the three numbers that are entered into three text boxes.

7.1 Conditional Operators

To control the VB program flow, we can use various conditional operators. Basically, they resemble mathematical operators. Conditional operators are very powerful tools, they let the VB program compare data values and then decide what action to take, whether to execute a program or terminate the program and more. These operators are shown in Table 7.1.

7.2 Logical Operators

In addition to conditional operators, there are a few logical operators which offer added power to the VB programs. There are shown in Table 7.2.

Table 7.1: Conditional Operators

Operator	Meaning
=	Equal to
>	More than
<	Less Than
>=	More than and equal
<=	Less than and equal
<>	Not Equal to

Table 7.2:Logical Operators

Operator	Meaning
And	Both sides must be true
or	One side or other must be true
Xor	One side or other must be true but not both
Not	Negates truth

* You can also compare strings with the above operators. However, there are certain rules to follows: Upper case letters are less than lowercase letters, "A"<"B"<"C"<"D".....<"Z" and number are less than letters.

7.3 Using If.....Then.....Else Statements with Operators

To effectively control the VB program flow, we shall use **If...Then...Else** statement together with the conditional operators and logical operators.
The general format for the **if...then...else** statement is

If conditions **Then**

VB expressions

Else

VB expressions

End If

* any If..Then..Else statement must end with End If. Sometime it is not necessary to use Else.

Example:

```
Private Sub OK_Click()
  firstnum=Val(usernum1.Text)
  secondnum=Val(usernum2.Text)
  If total=firstnum+secondnum And Val(sum.Text)<>0 Then

  correct.Visible = True
 wrong.Visible = False
Else
  correct.Visible = False
  wrong.Visible = True
End If

End Sub
```

Lesson 8 : Select Case....End select Control Structure

Select Case is preferred when there exist many different conditions because using If...Then..ElseIf statements might become too messy.
The format of the Select Case control structure is show below:

```
Select Case expression

   Case value1
        Block of one or more VB statements
   Case value2
        Block of one or more VB Statements
   Case value3
            .
        .
   Case Else
        Block of one or more VB Statements

End Select
```

Example 8.1

```
Dim grade As String

Private Sub Compute_Click( )

grade=txtgrade.Text

Select Case grade

  Case  "A"
       result.Caption="High  Distinction"

  Case "A-"
      result.Caption="Distinction"

  Case "B"
        result.Caption="Credit"

  Case "C"
        result.Caption="Pass"

  Case Else
        result.Caption="Fail"

  End Select

End Sub
```

Example 8.2

```
Dim mark As Single

Private Sub Compute_Click()
'Examination Marks
```

```
 mark = mrk.Text

Select Case mark
 Case Is >= 85

     comment.Caption = "Excellence"
Case Is >= 70

    comment.Caption = "Good"

Case Is >= 60
   comment.Caption = "Above Average"

Case Is >= 50
comment.Caption = "Average"

Case Else
comment.Caption = "Need to work harder"


End Select

End Sub
```

Example 8.3

Example 8.2 could be rewritten as follows:

```
Dim mark As Single

Private Sub Compute_Click()

'Examination Marks
 mark = mrk.Text

Select Case mark
 Case 0 to 49

     comment.Caption = "Need to work harder"

Case 50 to 59

    comment.Caption = "Average"

Case 60 to 69
   comment.Caption = "Above Average"

Case 70 to 84
comment.Caption = "Good"

Case Else
comment.Caption = "Excellence"

End Select


End Sub
```

Lesson 9: Looping

Visual Basic allows a procedure to be repeated many times as long as the processor until a condition or a set of conditions is fulfilled. This is generally called looping . Looping is a very useful feature of Visual Basic because it makes repetitive works easier. There are two kinds of loops in Visual Basic, the Do...Loop and the For.....Next loop

9.1 Do Loop

The formats are

a) Do While condition
    Block of one or more VB statements
  Loop

b) Do
    Block of one or more VB statements
  Loop While condition

c) Do Until condition
    Block of one or more VB statements
  Loop

d) Do
    Block of one or more VB statements
  Loop Until condition

9.2 Exiting the Loop

Sometime we need exit to exit a loop prematurely because of a certain condition is fulfilled. The syntax to use is known as Exit Do. You can examine Example 9.2 for its usage.

9.3 For....Next Loop

The format is:

```
For counter=startNumber to endNumber (Step increment)
   One or more VB statements
Next
```

Please refer to example 9.3a,9.3b and 9.3 c for its usage.

Sometimes the user might want to get out from the loop before the whole repetitive process is executed, the command to use is **Exit For**. To exit a For….Next Loop, you can place the **Exit For** statement within the loop; and it is normally used together with the If…..Then… statement. Let's examine example 9.3 d.

Example 9.1

```
Do while counter <=1000
     num.Text=counter
     counter =counter+1
Loop
```

* The above example will keep on adding until counter >1000.

The above example can be rewritten as

```
Do
     num.Text=counter
     counter=counter+1
Loop until counter>1000
```

Example 9.2

```
Dim sum, n As Integer
 Private Sub Form_Activate()
List1.AddItem "n" & vbTab & "sum"
Do
  n = n + 1
  Sum = Sum + n
              List1.AddItem n & vbTab & Sum
              If n = 100 Then
              Exit Do
              End If
               Loop
       End Sub
```

Explanation

In the above example, we compute the summation of 1+2+3+4+……+100. In the design stage, you need to insert a ListBox into the form for displaying the output, named List1. The program uses the AddItem method to populate the ListBox. The statement List1.AddItem "n" & vbTab & "sum" will display the headings in the ListBox, where it uses the vbTab function to create a space between the headings n and sum.

Example 9.3 a	**Example 9.3 b**
For counter=1 to 10 display.Text=counter Next	For counter=1 to 1000 step 10 counter=counter+1 Next

Example 9.3 c	**Example 9.3 d**

For counter=1000 to 5 step -5
counter=counter-10
Next
*Notice that increment can be negative
	Private Sub Form_Activate( )
	For n=1 to 10
	If n>6 then
	Exit For
	End If
	
	Else
	Print n
	End If
	End Sub