Variables

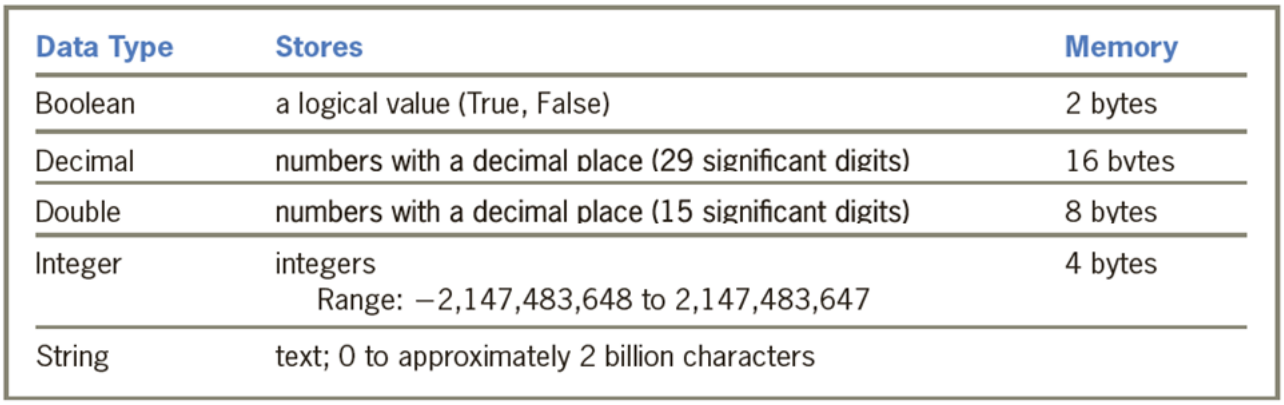
A variable is a computer memory location where a programmer can temporarily store an item of data while an application is running. The memory location is called a variable because its contents can change (vary) during run time. Examples of data stored in variables include all of the user items that will be included in a calculation, as well as the result of any calculation made by the application. Storing the data in a variable allows the programmer to control the preciseness of the data, and save the data for later use within the application's code. It also makes your code run more efficiently because the computer can process data stored in a variable much faster than it can process data stored in the property of a control.

You will use the Dim statement to declare a variable within the procedure that needs it. The statement assigns a name, a data type, and an initial value to the variable .

For example, the ***Dim M As Integer*** statement, declares a variable whose name is M, data type is Integer, and initial value is the number 0. When the computer processes the Dim statement, it sets aside a section in its main memory and attaches the name M to it. An instruction within the procedure can then use the name to access the memory location, perhaps to store a different value in it, use its current value in a calculation, or simply display its value. Before viewing more examples of Dim statements, you must learn how to select appropriate data types and names for your variables.

Selecting an Appropriate Data Type

A variable’s type indicates the type of data (e.g. numeric or textual) the variable will store. It also determines the variable’s size, which is the amount of memory it consumes. Visual Basic provides many different data types, but Figure 1-9 lists only the ones which we will used in this semester.



Data Type	Stores	Memory
Boolean	a logical value (True, False)	2 bytes
Decimal	numbers with a decimal place (29 significant digits)	16 bytes
Double	numbers with a decimal place (15 significant digits)	8 bytes
Integer	integers Range: −2,147,483,648 to 2,147,483,647	4 bytes
String	text; 0 to approximately 2 billion characters	

Figure 1-9 Visual Basic data types

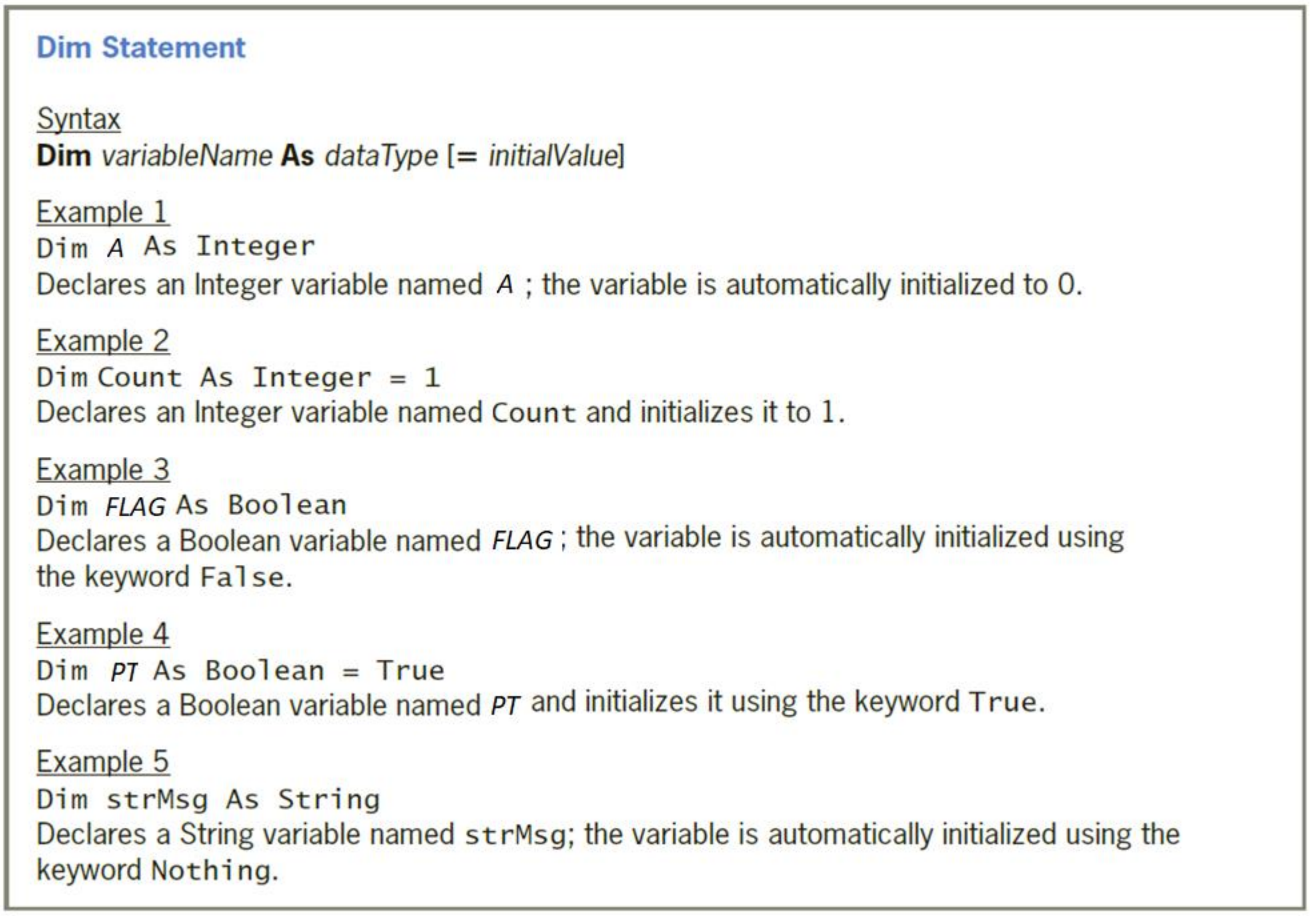
Variables assigned the Integer data type can store integers, which are positive or negative numbers that do not have any decimal places. If you need to store a number containing a decimal place, you would use either the Decimal or Double data type. The two data types differ in the range of numbers each can store and the amount of memory each needs to store the numbers. Also listed in Figure 1-9 are the String and Boolean data types. The String data type can store from zero to approximately 2 billion characters. The Boolean data type stores Boolean (or logical) values: either True or False.

Naming rules (these are required by Visual Basic)

1. Each variable declared in a procedure must have a unique name.
2. Each name must begin with a letter and contain only letters, numbers, and the underscore character.
3. The recommended maximum number of characters to use for a name is 32.
4. The name cannot be a reserved word (keyword), such as Sub or Double.

Examples of Variable Declaration Statements

Once you have chosen its data type and name, you can declare the variable in code. Figure 1-10 shows the Dim statement's syntax and includes several examples of using the statement.



Dim Statement

Syntax
Dim *variableName* **As** *dataType* [= *initialValue*]

Example 1
```
Dim A As Integer
```
Declares an Integer variable named *A* ; the variable is automatically initialized to 0.

Example 2
```
Dim Count As Integer = 1
```
Declares an Integer variable named `Count` and initializes it to `1`.

Example 3
```
Dim FLAG As Boolean
```
Declares a Boolean variable named *FLAG* ; the variable is automatically initialized using the keyword `False`.

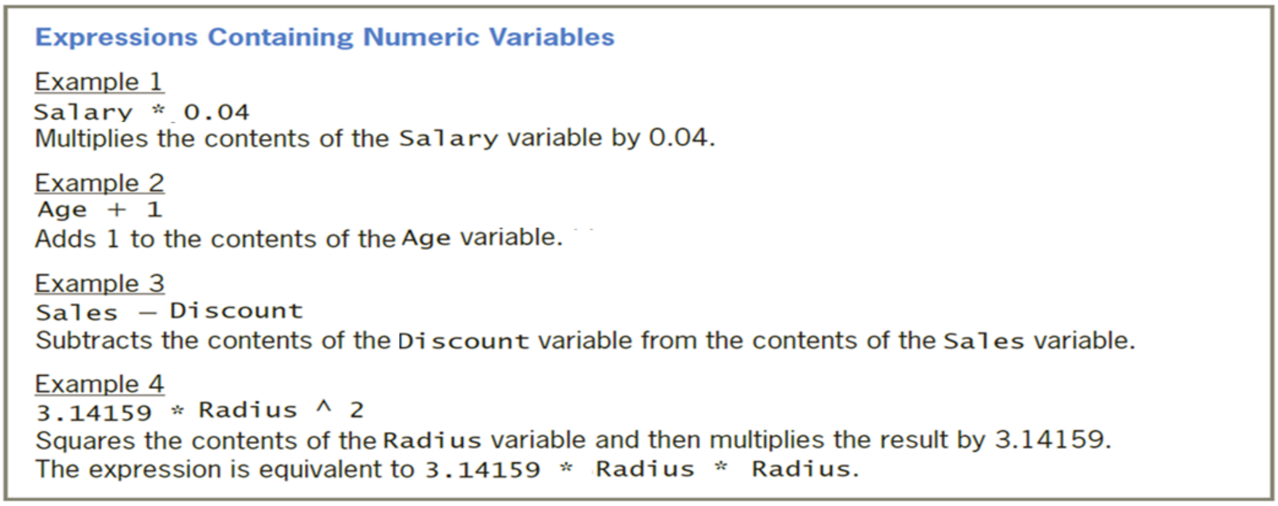
Example 4
```
Dim PT As Boolean = True
```
Declares a Boolean variable named *PT* and initializes it using the keyword `True`.

Example 5
```
Dim strMsg As String
```
Declares a String variable named `strMsg`; the variable is automatically initialized using the keyword `Nothing`.

Figure 1-10 Syntax and examples of the Dim statement

The square brackets in the syntax indicate that the "5 initialValue" part of the Dim statement is optional. If it is omitted, which most times it is, the computer stores a default value in the variable. The default value depends on the variable's data type. A variable declared using one of the numeric data types (Integer, Decimal, and Double) is automatically initialized to the number 0. The computer automatically initializes a Boolean variable using the keyword False. String variables are automatically initialized Nothing, which means they contain no data at all.

Now, in connection with the previous lecture, Arithmetic Expressions can also contain numeric variables, as shown in the examples in Figure 1-11. The computer uses the value stored inside the variable when evaluating the expression. The expression in Example 4 could be used to calculate the area of Circle.



Expressions Containing Numeric Variables

Example 1
```
Salary * 0.04
```
Multiplies the contents of the `Salary` variable by 0.04.

Example 2
```
Age + 1
```
Adds 1 to the contents of the `Age` variable.

Example 3
```
Sales – Discount
```
Subtracts the contents of the `Discount` variable from the contents of the `Sales` variable.

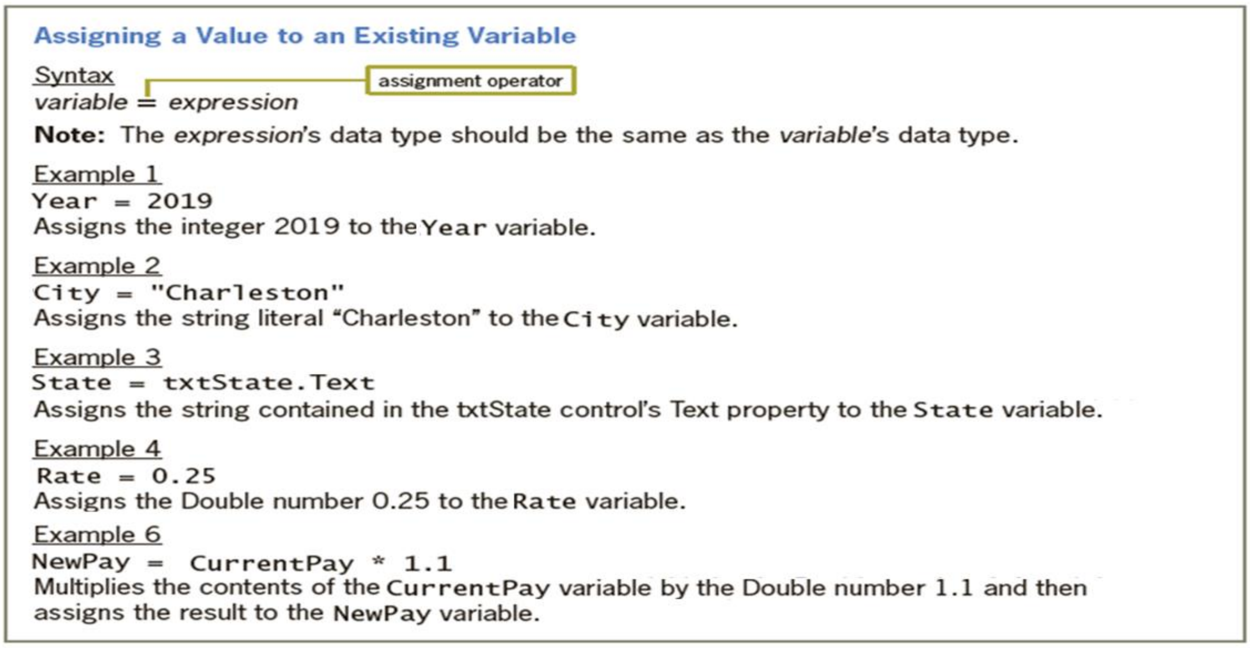
Example 4
```
3.14159 * Radius ^ 2
```
Squares the contents of the `Radius` variable and then multiplies the result by 3.14159.
The expression is equivalent to `3.14159 * Radius * Radius`.

Figure 1-11 Examples of arithmetic expressions containing numeric variables

Assigning a Value to an Existing Variable

In the previous lectures you learned how to use an assignment statement to assign a value to a control's property while an application is running. An assignment statement is also used to assign a value to a variable during run time; the syntax for doing this is shown in Figure 1-12. In the syntax, expression can contain items such as numbers, string literals, object properties

, variables, keywords, and arithmetic operators. When the computer processes an assignment statement, it evaluates the expression first and then stores the result in the variable. Keep in mind that a variable can store only one value at any one time. When you assign another value to a variable, the new value replaces the existing one.



Assigning a Value to an Existing Variable

Syntax
variable = *expression*

assignment operator

Note: The *expression*'s data type should be the same as the *variable*'s data type.

Example 1
```
Year = 2019
```
Assigns the integer 2019 to the `Year` variable.

Example 2
```
City = "Charleston"
```
Assigns the string literal "Charleston" to the `City` variable.

Example 3
```
State = txtState.Text
```
Assigns the string contained in the txtState control's Text property to the `State` variable.

Example 4
```
Rate = 0.25
```
Assigns the Double number 0.25 to the `Rate` variable.

Example 6
```
NewPay = CurrentPay * 1.1
```
Multiplies the contents of the `CurrentPay` variable by the Double number 1.1 and then assigns the result to the `NewPay` variable.

Figure 1-12 Syntax and examples of assigning a value to a variable

As noted in the figure, the expression’s data type should be the same as the variable’s data type. The statement in Example 1 assigns an integer to an Integer variable, and the statement in Example 2 assigns a string literal to a String variable. Notice that string literals are enclosed in quotation marks, but numbers and variable names are not. The quotation marks differentiate a string literal from both a number and a variable name. In other words, “2019” is a string literal, but 2019 is a number. Similarly, “Charleston” is a string literal, but Charleston (without the quotation marks) would be interpreted by the computer as the name of a variable. When the computer processes a statement that assigns a string literal to a String variable, it assigns only the characters that appear between the quotation marks; it does not assign the quotation marks themselves. The statement in Example 3 assigns the string contained in the txtState control’s Text property to a String variable. Example 4’s statement assigns the Double number 0.25 to a Double variable named Rate. In Visual Basic, a number that has a decimal place is automatically treated as a Double number. The NewPay = CurrentPay * 1.1 statement in Example 6 multiplies the contents of the CurrentPay variable by the Double number 1.1 and then assigns the result to the NewPay variable. Notice that the calculation appearing on the right side of the assignment operator is performed first, and then the result is assigned to the variable whose name appears on the left side of the operator.