Chapter 1 An Introduction to Programming

1.1 What is Programming?

A **program** is a list of instructions that is executed by a computer to accomplish a particular task.

Creating those instructions is **programming**.

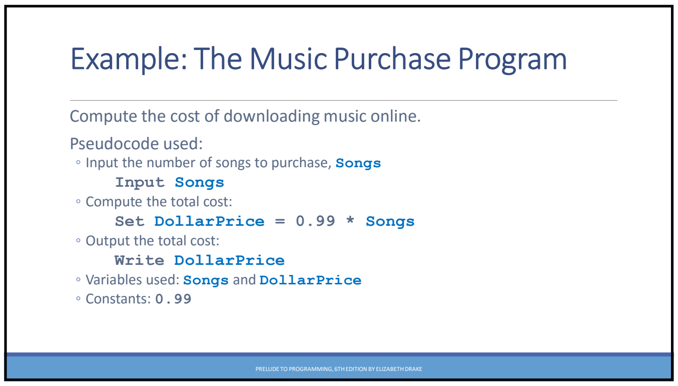
Program development cycle:

- Analyze the problem
- Design a program to solve the problem
- Code the program
- Test the program
- Revise as necessary

1.2 Basic Programming Concepts

Important concepts:

- **Pseudocode** is used to plan out the logic of a computer program, using English words and phrases.
- **Input** refers to information that is accepted into the program from the user or other sources.
- **Variables** represent values within a program.
- **Constants** are values used within a program. The value of a constant does not change (i.e., it is constant) throughout the program.



Example: The Music Purchase Program

Compute the cost of downloading music online.

Pseudocode used:

- Input the number of songs to purchase, `Songs`

```
Input Songs
```

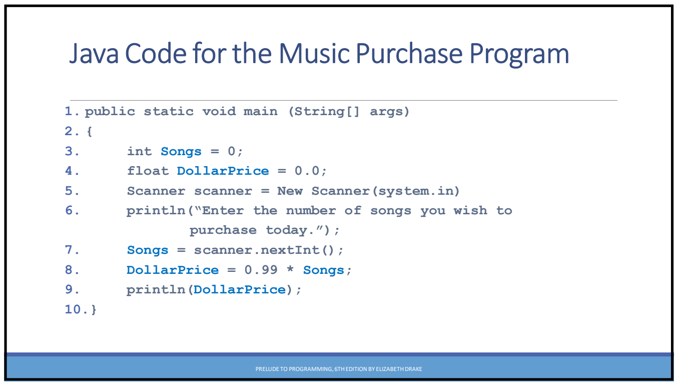
- Compute the total cost:

```
Set DollarPrice = 0.99 * Songs
```

- Output the total cost:

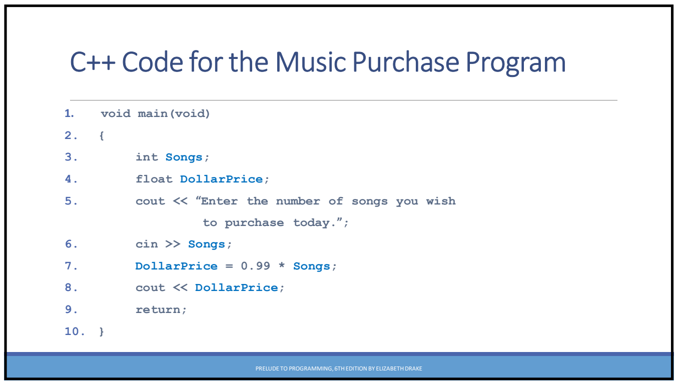
```
Write DollarPrice
```

- Variables used: `Songs` and `DollarPrice`
- Constants: `0.99`



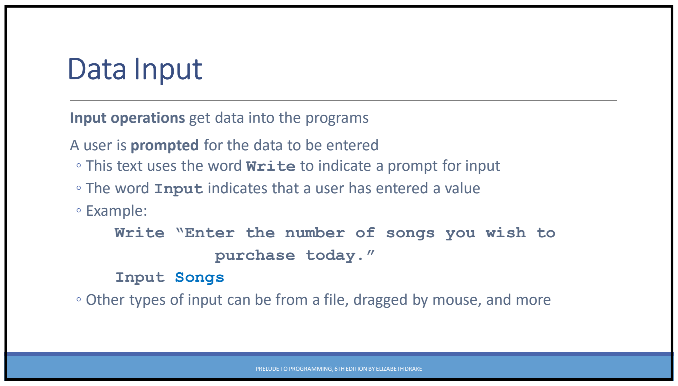
Java Code for the Music Purchase Program

```
public static void main (String[] args)
{
     int Songs = 0;
     float DollarPrice = 0.0;
     Scanner scanner = New Scanner(system.in)
     println("Enter the number of songs you wish to
             purchase today.");
     Songs = scanner.nextInt();
     DollarPrice = 0.99 * Songs;
     println(DollarPrice);
}
```



C++ Code for the Music Purchase Program

```
void main(void)
{
     int Songs;
     float DollarPrice;
     cout << "Enter the number of songs you wish
              to purchase today.";
     cin >> Songs;
     DollarPrice = 0.99 * Songs;
     cout << DollarPrice;
     return;
}
```



Data Input

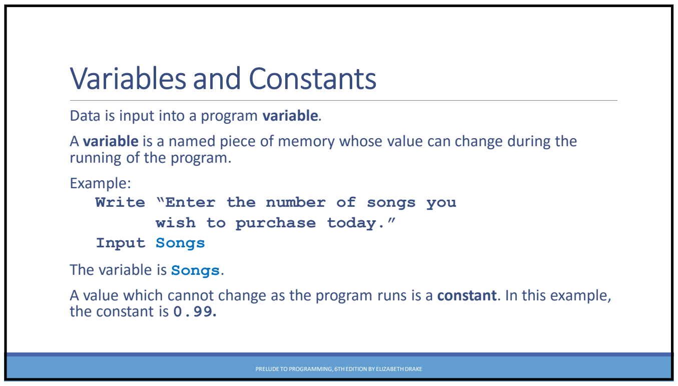
Input operations get data into the programs

A user is **prompted** for the data to be entered

- This text uses the word `Write` to indicate a prompt for input
- The word `Input` indicates that a user has entered a value
- Example:

```
    Write "Enter the number of songs you wish to
              purchase today."
    Input Songs
```

- Other types of input can be from a file, dragged by mouse, and more



Variables and Constants

Data is input into a program **variable**.

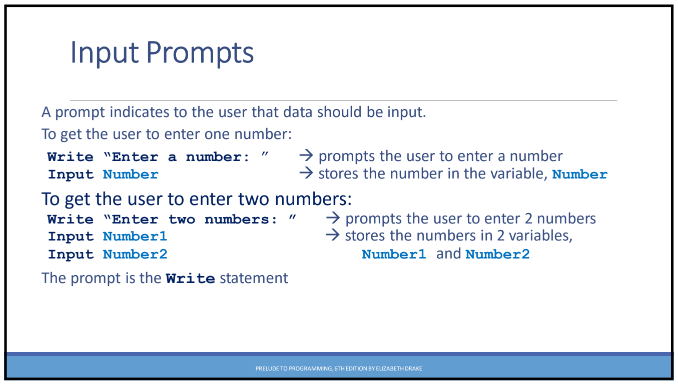
A **variable** is a named piece of memory whose value can change during the running of the program.

Example:

```
   Write "Enter the number of songs you
         wish to purchase today."
   Input Songs
```

The variable is `Songs`.

A value which cannot change as the program runs is a **constant**. In this example, the constant is `0.99`.



Input Prompts

A prompt indicates to the user that data should be input.

To get the user to enter one number:

```
Write "Enter a number: "    → prompts the user to enter a number
Input Number                → stores the number in the variable, Number
```

To get the user to enter two numbers:

```
Write "Enter two numbers: "    → prompts the user to enter 2 numbers
Input Number1                  → stores the numbers in 2 variables,
Input Number2                       Number1 and Number2
```

The prompt is the `Write` statement

Naming Variables

- All variable names must be one word
- Spaces are never allowed
- Variables cannot begin with a number
- Names should be meaningful
- Long names are allowed but names should be as short as possible, yet still be meaningful

Variable Name Examples

Some examples:
`Miles_traveled` is fine
`Miles Traveled` is not (space)
`TaxRate_1` is fine
`1_TaxRate` is not (begins with a number)
`Variable1` is fine but not meaningful
`Z` is fine but not meaningful

What's wrong with these?
```
My Number
2_4_6_8_go
CowWhoJumpedOverTheMoon
#
```

What's really happening?

A variable is the name for a **storage location** in the computer's internal memory.

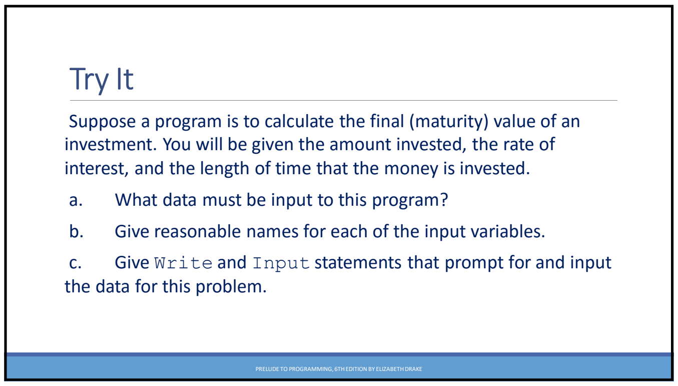
The **value of a variable is the contents of that location**. The contents of the computer's memory after the `Input` statement in the Music Purchase program is executed and the user wants to download 78 songs:

78	
`Songs`	`DollarPrice`

The `DollarPrice` mailbox is empty – it has not yet been assigned a value. At the end of the program, the contents of memory would be:

78	77.22
`Songs`	`DollarPrice`

Every time you run the program with a different number of songs, the values in the memory locations will change. The contents of the `Songs` memory "box" will be replaced by the new number of songs and, after the calculation is made, the contents of the `DollarPrice` memory location will be replaced by the new value.



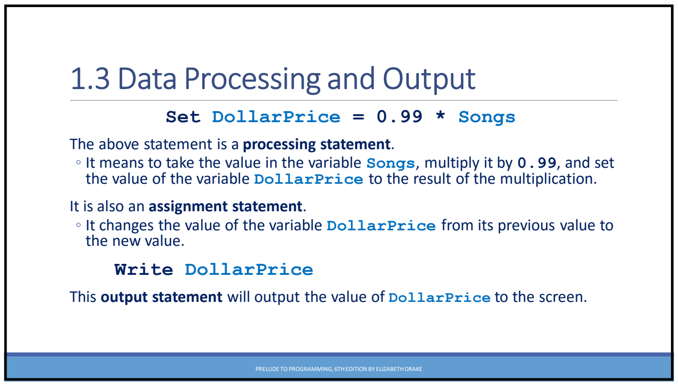
Try It

Suppose a program is to calculate the final (maturity) value of an investment. You will be given the amount invested, the rate of interest, and the length of time that the money is invested.

a. What data must be input to this program?

b. Give reasonable names for each of the input variables.

c. Give `Write` and `Input` statements that prompt for and input the data for this problem.



1.3 Data Processing and Output

```
Set DollarPrice = 0.99 * Songs
```

The above statement is a **processing statement**.

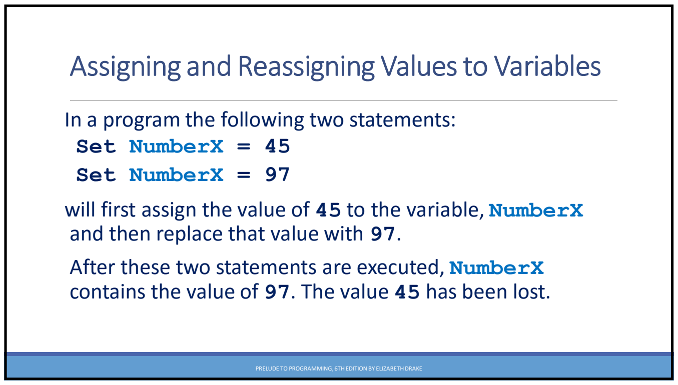
- It means to take the value in the variable `Songs`, multiply it by `0.99`, and set the value of the variable `DollarPrice` to the result of the multiplication.

It is also an **assignment statement**.

- It changes the value of the variable `DollarPrice` from its previous value to the new value.

```
Write DollarPrice
```

This **output statement** will output the value of `DollarPrice` to the screen.



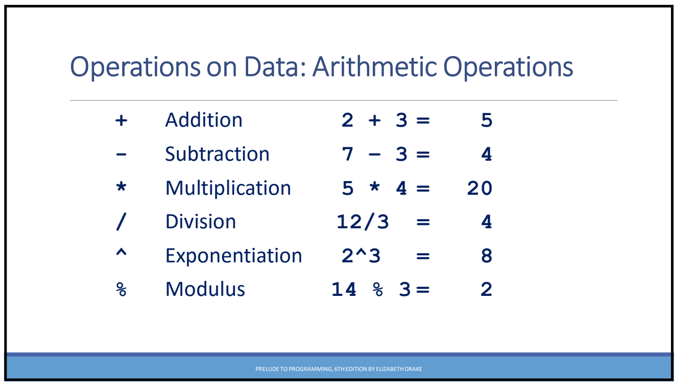
Assigning and Reassigning Values to Variables

In a program the following two statements:

```
Set NumberX = 45
Set NumberX = 97
```

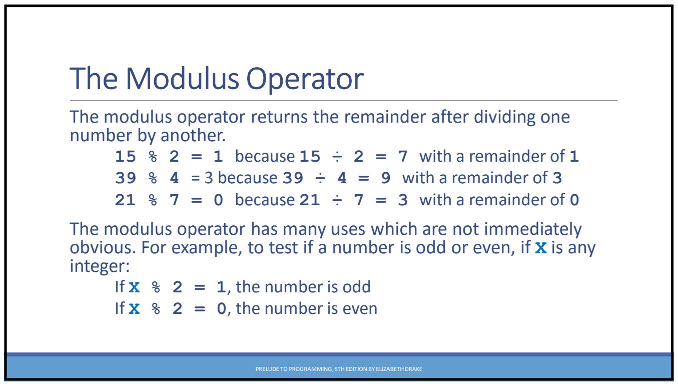
will first assign the value of **45** to the variable, `NumberX` and then replace that value with **97**.

After these two statements are executed, `NumberX` contains the value of **97**. The value **45** has been lost.



Operations on Data: Arithmetic Operations

`+`	Addition	`2 + 3 =`	`5`
`–`	Subtraction	`7 – 3 =`	`4`
`*`	Multiplication	`5 * 4 =`	`20`
`/`	Division	`12/3 =`	`4`
`^`	Exponentiation	`2^3 =`	`8`
`%`	Modulus	`14 % 3 =`	`2`



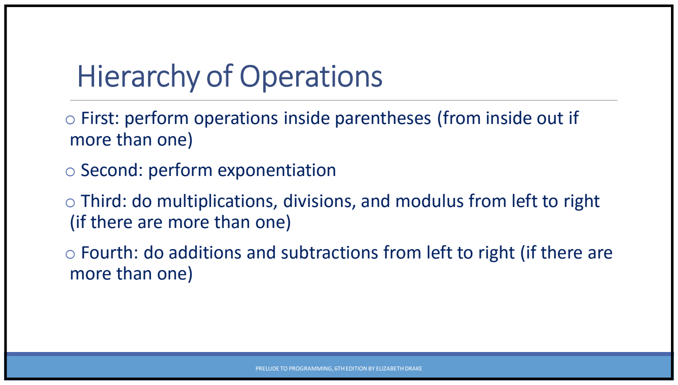
The Modulus Operator

The modulus operator returns the remainder after dividing one number by another.

`15 % 2 = 1` because `15 ÷ 2 = 7` with a remainder of `1`
`39 % 4` = 3 because `39 ÷ 4 = 9` with a remainder of `3`
`21 % 7 = 0` because `21 ÷ 7 = 3` with a remainder of `0`

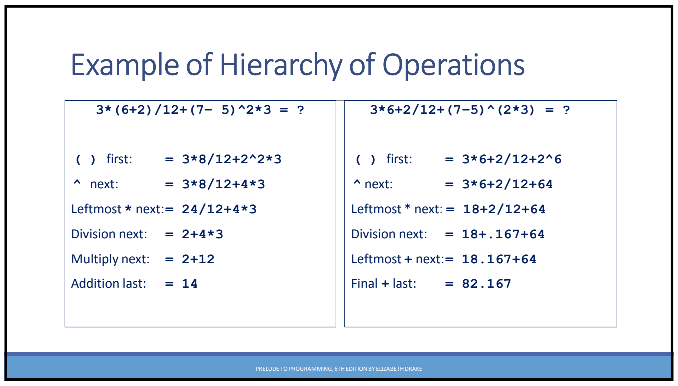
The modulus operator has many uses which are not immediately obvious. For example, to test if a number is odd or even, if `X` is any integer:

If `X % 2 = 1`, the number is odd
If `X % 2 = 0`, the number is even



Hierarchy of Operations

- First: perform operations inside parentheses (from inside out if more than one)
- Second: perform exponentiation
- Third: do multiplications, divisions, and modulus from left to right (if there are more than one)
- Fourth: do additions and subtractions from left to right (if there are more than one)



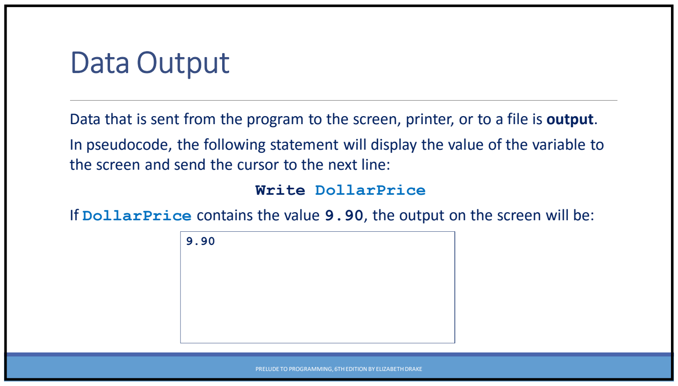
Example of Hierarchy of Operations

```
3*(6+2)/12+(7- 5)^2*3 = ?

( ) first:        = 3*8/12+2^2*3
^ next:           = 3*8/12+4*3
Leftmost * next:= 24/12+4*3
Division next:    = 2+4*3
Multiply next:    = 2+12
Addition last:    = 14
```

```
3*6+2/12+(7–5)^(2*3) = ?

( ) first:        = 3*6+2/12+2^6
^ next:           = 3*6+2/12+64
Leftmost * next:  = 18+2/12+64
Division next:    = 18+.167+64
Leftmost + next:= 18.167+64
Final + last:     = 82.167
```



Data Output

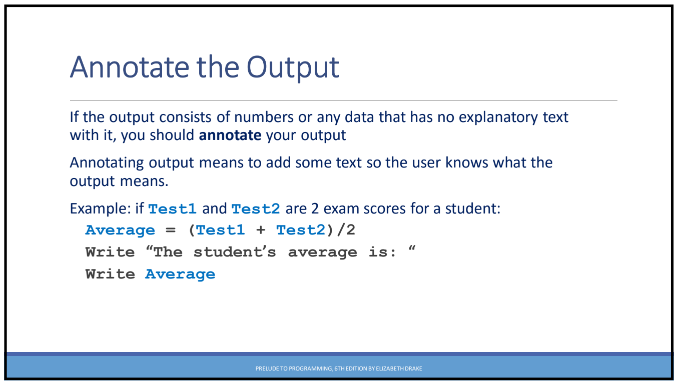
Data that is sent from the program to the screen, printer, or to a file is **output**.

In pseudocode, the following statement will display the value of the variable to the screen and send the cursor to the next line:

```
Write DollarPrice
```

If `DollarPrice` contains the value `9.90`, the output on the screen will be:

```
9.90
```



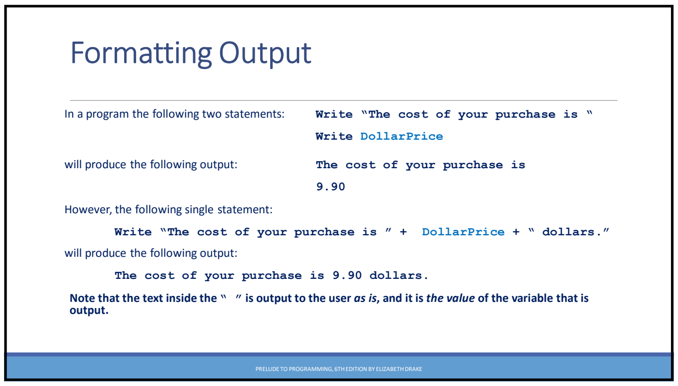
Annotate the Output

If the output consists of numbers or any data that has no explanatory text with it, you should **annotate** your output

Annotating output means to add some text so the user knows what the output means.

Example: if `Test1` and `Test2` are 2 exam scores for a student:

```
Average = (Test1 + Test2)/2
Write "The student's average is: "
Write Average
```



Formatting Output

In a program the following two statements:

```
Write "The cost of your purchase is "
Write DollarPrice
```

will produce the following output:

```
The cost of your purchase is
9.90
```

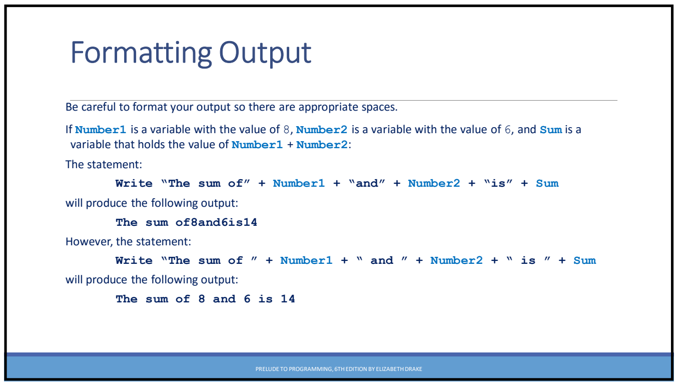
However, the following single statement:

```
        Write "The cost of your purchase is " +  DollarPrice + " dollars."
```

will produce the following output:

```
        The cost of your purchase is 9.90 dollars.
```

Note that the text inside the " " is output to the user *as is*, and it is *the value* of the variable that is output.



Formatting Output

Be careful to format your output so there are appropriate spaces.

If `Number1` is a variable with the value of `8`, `Number2` is a variable with the value of `6`, and `Sum` is a variable that holds the value of `Number1` + `Number2`:

The statement:

```
        Write "The sum of" + Number1 + "and" + Number2 + "is" + Sum
```

will produce the following output:

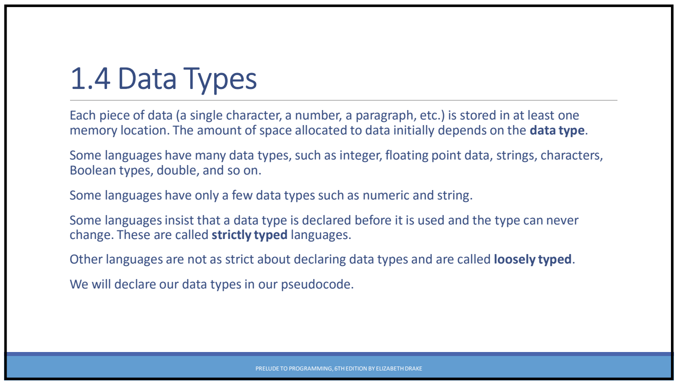
```
        The sum of8and6is14
```

However, the statement:

```
        Write "The sum of " + Number1 + " and " + Number2 + " is " + Sum
```

will produce the following output:

```
        The sum of 8 and 6 is 14
```



1.4 Data Types

Each piece of data (a single character, a number, a paragraph, etc.) is stored in at least one memory location. The amount of space allocated to data initially depends on the **data type**.

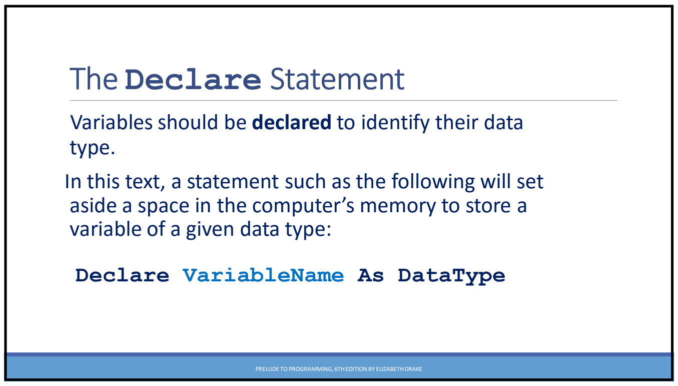
Some languages have many data types, such as integer, floating point data, strings, characters, Boolean types, double, and so on.

Some languages have only a few data types such as numeric and string.

Some languages insist that a data type is declared before it is used and the type can never change. These are called **strictly typed** languages.

Other languages are not as strict about declaring data types and are called **loosely typed**.

We will declare our data types in our pseudocode.

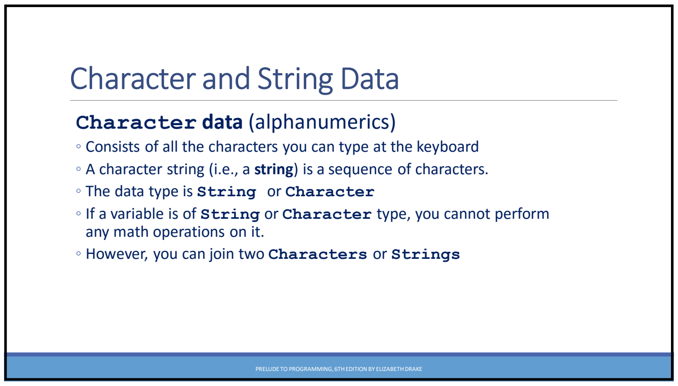


The **`Declare`** Statement

Variables should be **declared** to identify their data type.

In this text, a statement such as the following will set aside a space in the computer's memory to store a variable of a given data type:

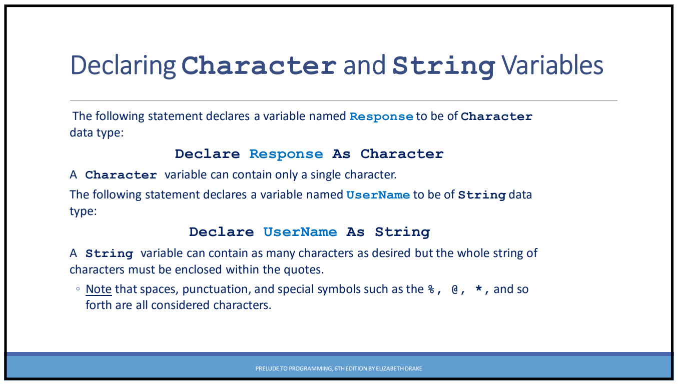
```
Declare VariableName As DataType
```



Character and String Data

`Character` data (alphanumerics)

- Consists of all the characters you can type at the keyboard
- A character string (i.e., a **string**) is a sequence of characters.
- The data type is **`String`** or **`Character`**
- If a variable is of **`String`** or **`Character`** type, you cannot perform any math operations on it.
- However, you can join two **`Characters`** or **`Strings`**



Declaring **`Character`** and **`String`** Variables

The following statement declares a variable named **`Response`** to be of **`Character`** data type:

```
Declare Response As Character
```

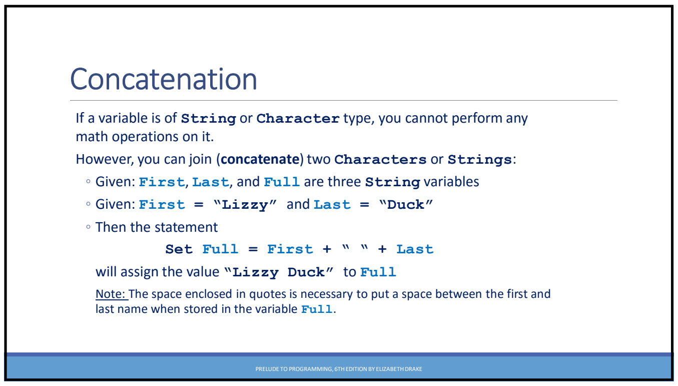
A **`Character`** variable can contain only a single character.

The following statement declares a variable named **`UserName`** to be of **`String`** data type:

```
Declare UserName As String
```

A **`String`** variable can contain as many characters as desired but the whole string of characters must be enclosed within the quotes.

- Note that spaces, punctuation, and special symbols such as the `%, @, *`, and so forth are all considered characters.



Concatenation

If a variable is of `String` or `Character` type, you cannot perform any math operations on it.

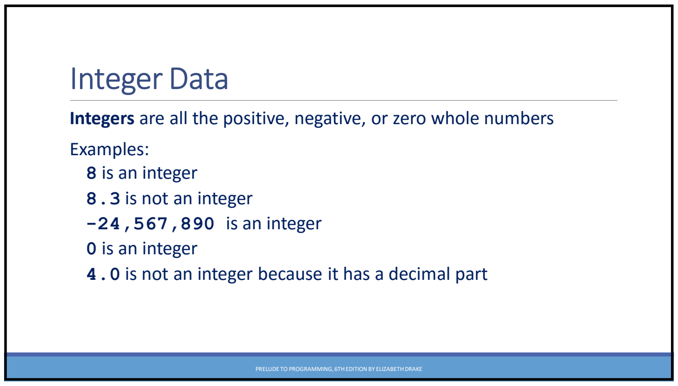
However, you can join (**concatenate**) two `Characters` or `Strings`:

- Given: `First`, `Last`, and `Full` are three `String` variables
- Given: `First = "Lizzy"` and `Last = "Duck"`
- Then the statement

```
Set Full = First + " " + Last
```

will assign the value `"Lizzy Duck"` to `Full`

<u>Note:</u> The space enclosed in quotes is necessary to put a space between the first and last name when stored in the variable `Full`.

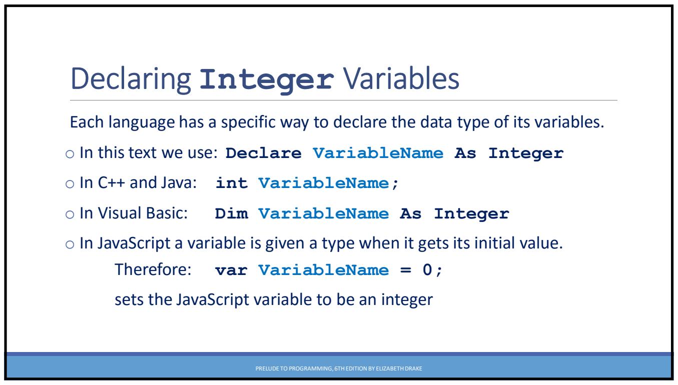


Integer Data

Integers are all the positive, negative, or zero whole numbers

Examples:

`8` is an integer
`8.3` is not an integer
`-24,567,890` is an integer
`0` is an integer
`4.0` is not an integer because it has a decimal part



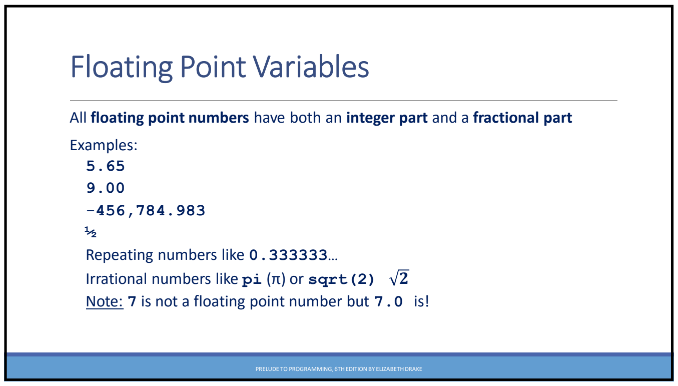
Declaring `Integer` Variables

Each language has a specific way to declare the data type of its variables.

- In this text we use: `Declare VariableName As Integer`
- In C++ and Java: `int VariableName;`
- In Visual Basic: `Dim VariableName As Integer`
- In JavaScript a variable is given a type when it gets its initial value.

  Therefore: `var VariableName = 0;`

  sets the JavaScript variable to be an integer



Floating Point Variables

All **floating point numbers** have both an **integer part** and a **fractional part**

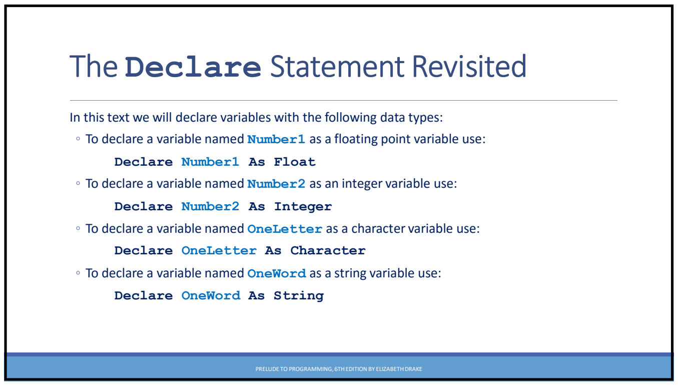
Examples:

```
5.65
9.00
-456,784.983
```
½

Repeating numbers like `0.333333`…

Irrational numbers like `pi` (π) or `sqrt(2)` $\sqrt{2}$

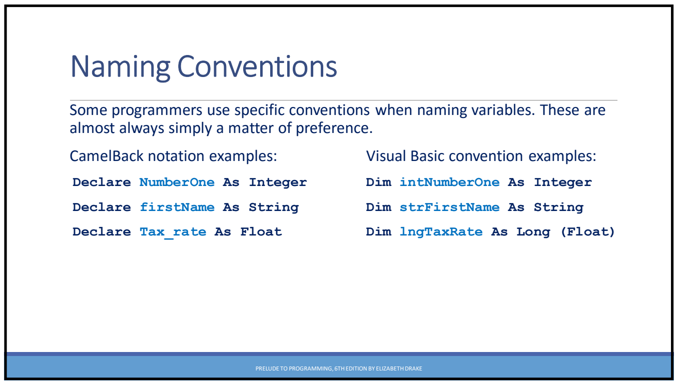
Note: `7` is not a floating point number but `7.0` is!



The `Declare` Statement Revisited

In this text we will declare variables with the following data types:

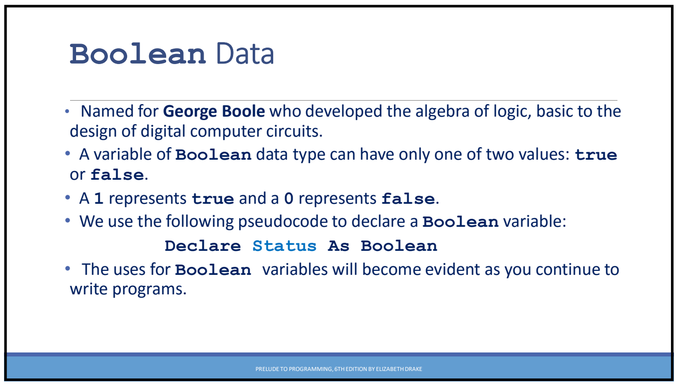
- To declare a variable named `Number1` as a floating point variable use:
  ```
  Declare Number1 As Float
  ```
- To declare a variable named `Number2` as an integer variable use:
  ```
  Declare Number2 As Integer
  ```
- To declare a variable named `OneLetter` as a character variable use:
  ```
  Declare OneLetter As Character
  ```
- To declare a variable named `OneWord` as a string variable use:
  ```
  Declare OneWord As String
  ```



Naming Conventions

Some programmers use specific conventions when naming variables. These are almost always simply a matter of preference.

CamelBack notation examples:	Visual Basic convention examples:
`Declare NumberOne As Integer`	`Dim intNumberOne As Integer`
`Declare firstName As String`	`Dim strFirstName As String`
`Declare Tax_rate As Float`	`Dim lngTaxRate As Long (Float)`



Boolean Data

- Named for **George Boole** who developed the algebra of logic, basic to the design of digital computer circuits.
- A variable of `Boolean` data type can have only one of two values: `true` or `false`.
- A `1` represents `true` and a `0` represents `false`.
- We use the following pseudocode to declare a `Boolean` variable:

```
Declare Status As Boolean
```

- The uses for `Boolean` variables will become evident as you continue to write programs.