CHAPTER 3

Input, Variables, and Calculations

Storing Data with Variables

- A *variable* is a name that represents a value stored in the computer's memory.
- So far, apps that you have created have stored data only in component properties.
- For instance, a component's `Text` property is used to hold data that you want to display.

Storing Data with Variables

Local Variables and Global Variables

- A *local variable* is created inside a method or function, and it can be accessed only by blocks that are also in that method or function.
- A *global variable* is created outside of all methods and functions in the workspace. It can be accessed by any blocks in the workspace, regardless of which method or function they belong to.

Storing Data with Variables

Creating a Local Variable

- To create a local variable, you must *initialize* it.
- To create and initialize a local variable, open the *Variables* drawer In the *Built-in* section of the Blocks column.
- Notice that in Figure 3-34 there are two blocks that are shaped differently. For now you want to use the one that is circled.

Storing Data with Variables

Figure 3-34 Creating a Variable Initialization Block (*Source:* MIT App Inventor 2)

Storing Data with Variables

Creating a Local Variable

When you create an *`initialize local name to`* block, place it inside the method or function that it will belong to.

Figure 3-36 An `initialize local name to` Block Placed Inside a Button's `Click` Event Handler (*Source:* MIT App Inventor 2)

The variable initialization block isn't complete yet. We need to:

- Change the variables name to something that describes the variables purpose.
- Assign an initial value to the variable.

Storing Data with Variables

Changing the Variable's Name

The following rules apply to variable names in App Inventor:

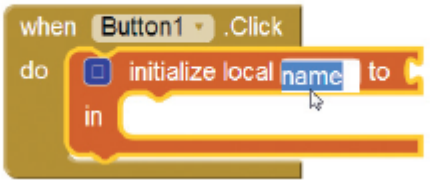
- The variable name must begin with an alphabetic letter.
- After the first letter, the remaining characters can be alphabetical letters, numbers, or underscore characters (_).
- You cannot have spaces in a variable name.
- Variable names must be unique within a project.

Storing Data with Variables

Changing the Variable's Name

To change a variable's name, click the word *name* on the *initialize local name to* block.

Figure 3-37 Changing the Variable Name (*Source:* MIT App Inventor 2)

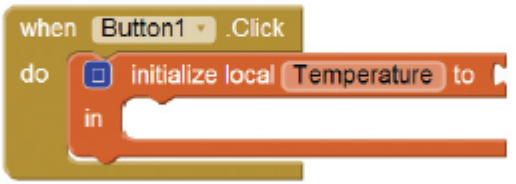


Storing Data with Variables

Assigning an Initial Value to the Variable

- When we set a variable to a value, we are assigning a value to the variable.
- Noticed that the variable initialization block in figure 3-38 has a socket label too.
- This socket requires a value.

Figure 3-38 The Variable Name Changed to `Temperature` (*Source:* MIT App Inventor 2)



Storing Data with Variables

Assigning an Initial Value to the Variable

The blocks that you can plug into this socket are:

- *number* blocks
- text string blocks
- Boolean blocks
- List blocks
- *Color* blocks

Storing Data with Variables

Assigning an Initial Value to the Variable

- Figure 3-39 shows two variable initialization blocks. The upper block defines a variable named *`Age`* and sets its initial value to the number 25.
- The lower block defines a variable named *`FirstName`* and sets its initial value to the text *Johnny*.

Figure 3-39 Two Complete Variable Initialization Blocks (*Source:* MIT App Inventor 2)

Storing Data with Variables

Creating a Local Variable That Holds a Number

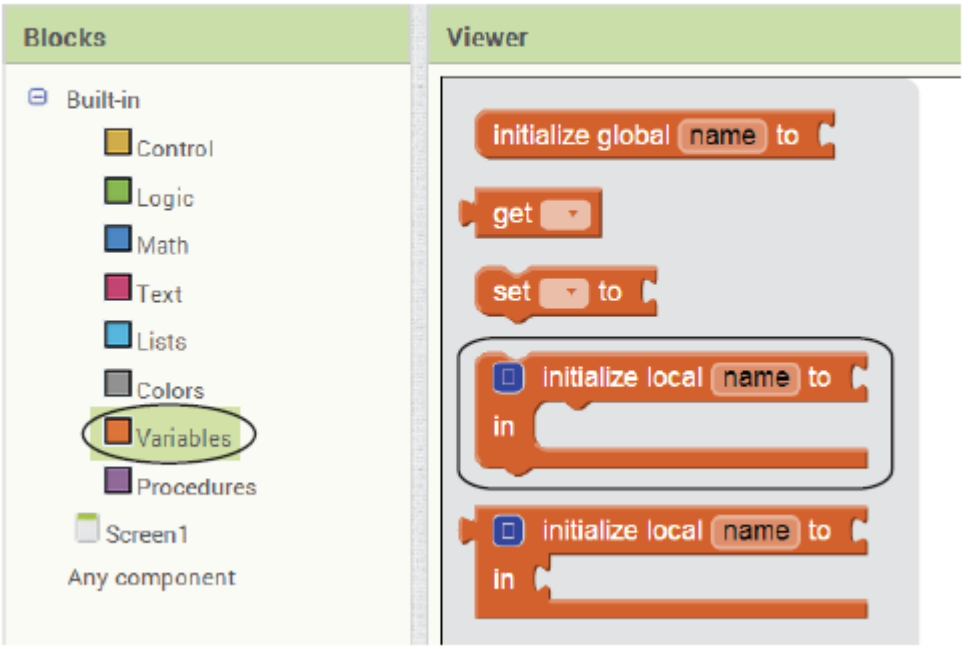
- Suppose we have a *`Click`* event handler for a button.
- We want to create a local variable to hold a car's speed.
- We initially assign the number zero to the variable.
- Here are the steps:
  - In the Blocks Editor's *Built-in* section, click *Variables*.
  - Select the initialize *`local name to`* block as shown in figure 3-40.

Storing Data with Variables

Creating a Local Variable That Holds a Number

This creates an *initialize local name to* block in your workspace.

Figure 3-40 Creating a Variable Initialization Block (*Source:* MIT App Inventor 2)

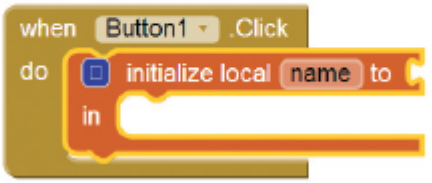


Storing Data with Variables

Creating a Local Variable That Holds a Number

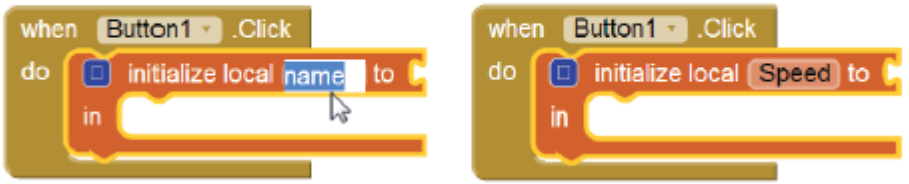
Place the block inside the desired event handler as shown in figure 3-41.

Figure 3-41 Insert the `initialize local name to` Block Inside the Desired Event Handler (*Source:* MIT App Inventor 2)



Click the word *name* and change the name to *Speed* as shown in Figure 3-42.

Figure 3-42 Renaming the Variable (*Source:* MIT App Inventor 2)



Storing Data with Variables

Creating a Local Variable That Holds a Number

- Create a *number* block to assign to the *Speed* variable.
- In the *Built-in* Section of the Blocks column, click *Math*, then click the *number* block (0).
- Plug the block into the *to* socket of the *Speed* variable initialization block as shown in figure 3-43.

Figure 3-43 Assigning the Number 0 (*Source:* MIT App Inventor 2)

Storing Data with Variables

Creating a Variable That Holds Text

Suppose we have a `Click` event handler for a button and we want to create a variable that holds the text *Dark Roast Coffee*. Here are the steps:

- In the Blocks Editor, go to the *Built-in* section of the Blocks column, click *Variables*.
- Select the `initialize local name to` block as shown in Figure 3-44.

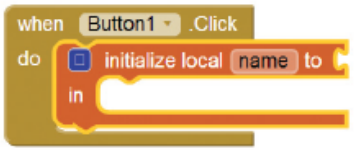
Storing Data with Variables

Creating a Variable That Holds Text

- This creates an *initialize local name to* block in your workspace.
- Place the block inside the desired event handler as seen in Figure 3-45.

Figure 3-44 Creating a Variable Initialization Block (*Source:* MIT App Inventor 2)

Figure 3-45 Insert the `initialize local` *name* `to` Block Inside the Desired Event Handler (*Source:* MIT App Inventor 2)

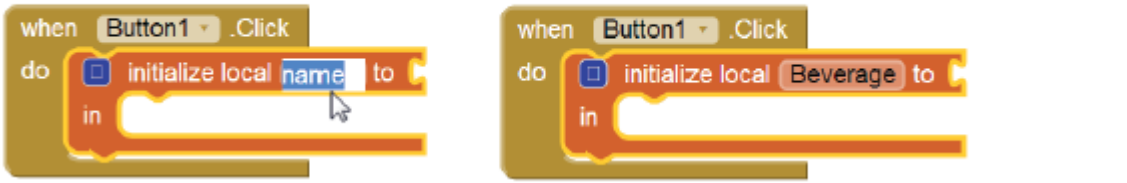


Storing Data with Variables

Creating a Variable That Holds Text

- Change the variable's name to *`Beverage`*.
- Click the word *name* that appears on the block as in Figure 3-46.

Figure 3-46 Renaming the Variable (*Source:* MIT App Inventor 2)



Storing Data with Variables

Creating a Variable That Holds Text

- Create a text string block to assign to the *Beverage* variable.
- In the *Built-in* section of the Blocks column, click *Text*.
- Click the text string block  .
- Plug the block into the *to* socket of the *Beverage* variable.
- Click the empty space between the quotation marks, as shown on the left in Figure 3-47.

Storing Data with Variables

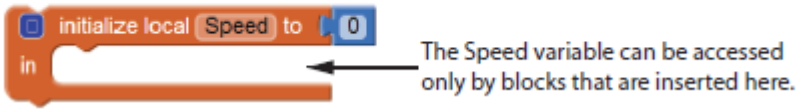
Figure 3-47 Assigning the Text *Dark Roast Coffee* (*Source:* MIT App Inventor 2)

Storing Data with Variables

Working with a Local Variable

The blocks that work with a local variable must be inserted inside the variables initialization block, as shown in figure 3-48.

Figure 3-48 Where to Insert Blocks that Work with a Variable (*Source:* MIT App Inventor 2)

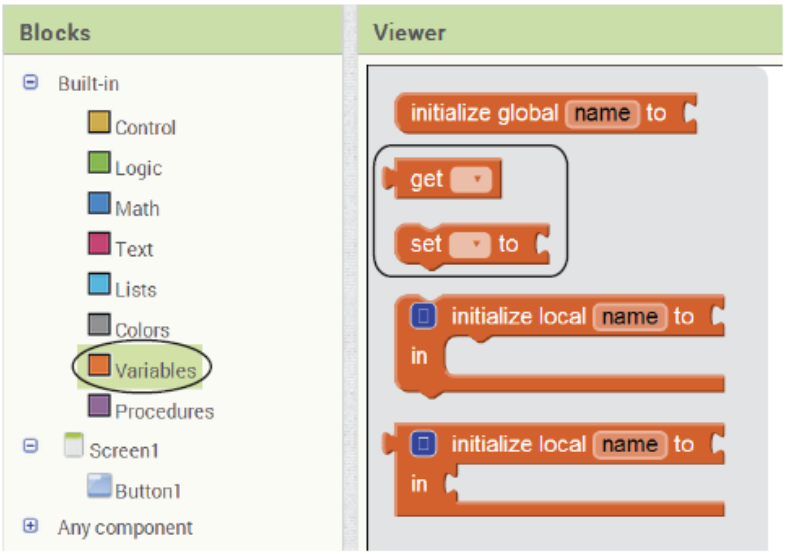


Storing Data with Variables

Working with a Local Variable

- Use the *get* instruction to get a variables value.
- Use a *set* instruction to store the value in the variable.
- You will find the *get* and *set* blocks in the *Variables* drawer as shown in Figure 3-49.

Figure 3-49 Blocks for Setting and Getting the Value of the Beverage Variable
(*Source:* MIT App Inventor 2)



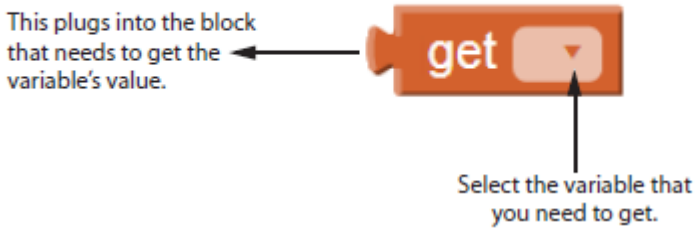
Storing Data with Variables

Working with a Local Variable

When you create a *get* block, you do two things:

1. You plug the *get* block into the block that needs to get the value.
2. On the *get* block, you select the variable that you need to get.

Figure 3-50 Using the get Block (*Source:* MIT App Inventor 2)



Storing Data with Variables

Working with a Local Variable

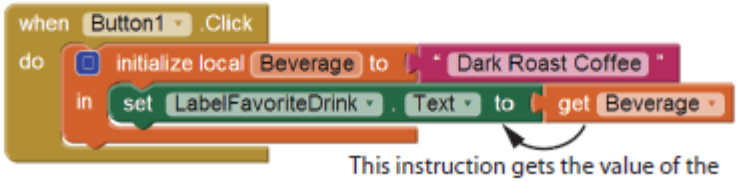
- Figure 3-51 shows that we have created a *get* block and we are going to plug it into the *setLabelFavoriteDrink to* block.
- Next complete the *get* blocked by selecting the *Beverage* variable.
- As shown in figure 3-52 click the down arrow on the *get* block and select *Beverage*.
- Figure 3-53 shows the completed instruction.

Storing Data with Variables

Figure 3-51 Plugging a `get` Block into Another Block (*Source:* MIT App Inventor 2)

Figure 3-52 Selecting the `Beverage` Variable for the `get` Block
(*Source:* MIT App Inventor 2)

Figure 3-53 The Completed Instruction (*Source:* MIT App Inventor 2)



This instruction gets the value of the Beverage variable and assigns it to the Text property of the LabelFavoriteDrink component.

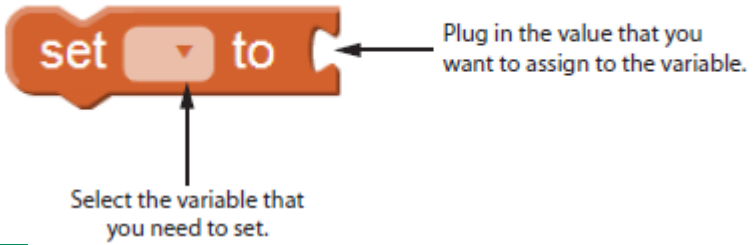
Storing Data with Variables

Working with a Local Variable

When you create a *set* block for a local variable, you do the following things:

- Insert the *set* block into the desired variable's initialization block.
- On the *set* block, select the name of the variable that you want to set.
- Plug a value into the *to* socket of the *set* block.

Figure 3-54 Using the `set` Block (*Source:* MIT App Inventor 2)



Storing Data with Variables

Working with a Local Variable

Suppose we have a local variable named *Speed*, initialized to the value zero, and we want to change its value to 75. Do the following things:

- Create a *set* block and insert into the speed variables in initialization block as shown in Figure 3-55.
- On the *set* block, select the *Speed* variable as shown in Figure 3-56.
- Create a number block for the value 75 and plug it into the *set* block as shown in Figure 3-57.

Storing Data with Variables

Figure 3-55 The `set` Block Created (*Source:* MIT App Inventor 2)

Figure 3-56 Selecting the `Speed` Variable on the `set` Block (*Source:* MIT App Inventor 2)

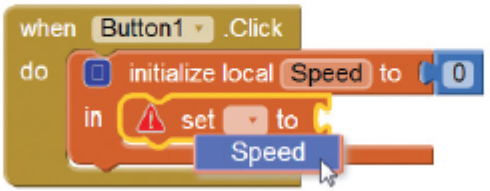


Figure 3-57 Plugging the Value 75 into the `set` Block (*Source:* MIT App Inventor 2)

Storing Data with Variables

Working with a Local Variable

Note: When you create a *`set`* block, you cannot select the name of the local variable until you plug the set block somewhere inside that local variable's initialization block.

Storing Data with Variables

Variable Scope

- A variable's *scope* is described as part of the program in which a variable may be accessed.
- A variable is visible only to instructions inside the variable's scope.
- The variable can be accessed only by the instructions that are inside the *`initialize local name to`* block.
- Figure 3-63 shows an example.

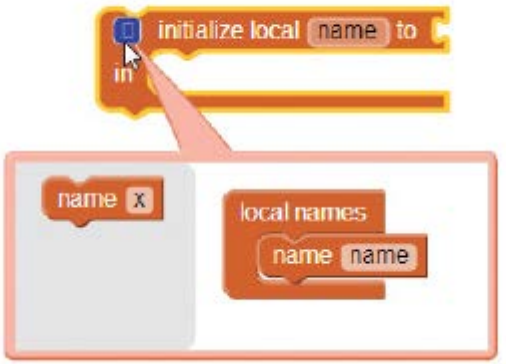
Figure 3-63 The Scope of a Local Variable (*Source:* MIT App Inventor 2)

Storing Data with Variables

Creating Multiple Local Variables

- The *`initialize local name to`* block can be modified to create and initialize multiple variables.
- Click the blue box that appears in the block's upper-left corner to display the mutator bubble as shown in figure 3-64.

Figure 3-64 Mutator Bubble (*Source:* MIT App Inventor 2)

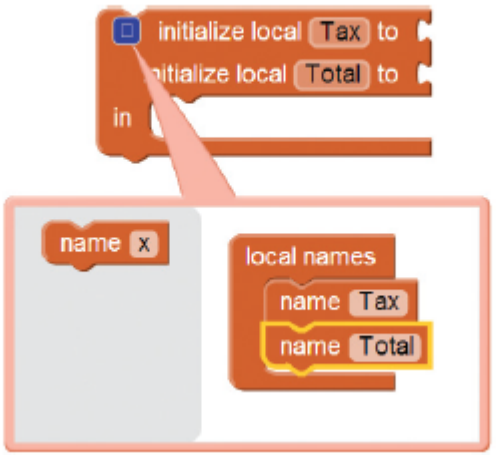


Storing Data with Variables

Creating Multiple Local Variables

- Double click the variable name to change it to something more descriptive.
- Figure 3-67 shows an initialization block that creates two variables named *Tax* and *Total*.

Figure 3-67 The Variable Names Changed to Tax and Total (*Source:* MIT App Inventor 2)

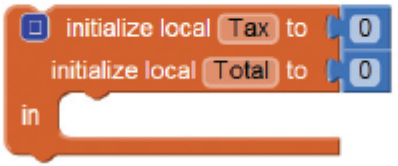


Storing Data with Variables

Creating Multiple Local Variables

The last step is to plug initialization values into each variable as in Figure 3-68.

Figure 3-68 The `Tax` and `Total` Variables Initialized to the Value 0
(*Source:* MIT App Inventor 2)



Storing Data with Variables

Creating Multiple Local Variables

Here is an example that uses to local variables in an event handler.

Figure 3-69 The `SalesTaxCalculator` Project (Source: MIT App Inventor 2)

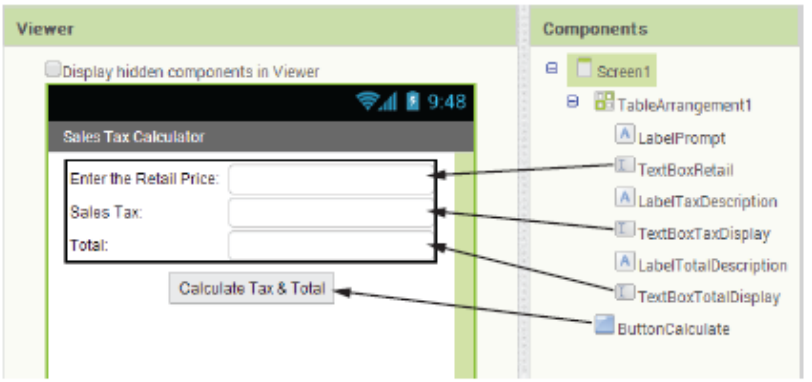
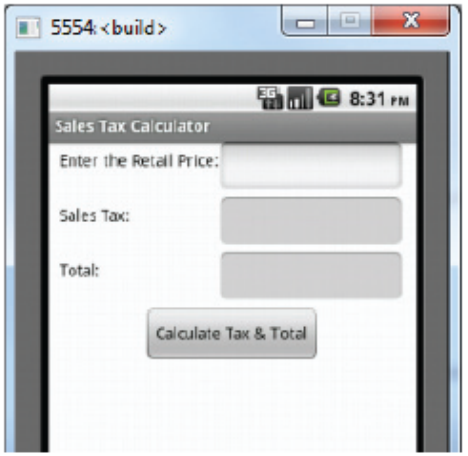


Figure 3-70 The `SalesTaxCalculator` App Running in the Emulator (Source: MIT App Inventor 2)

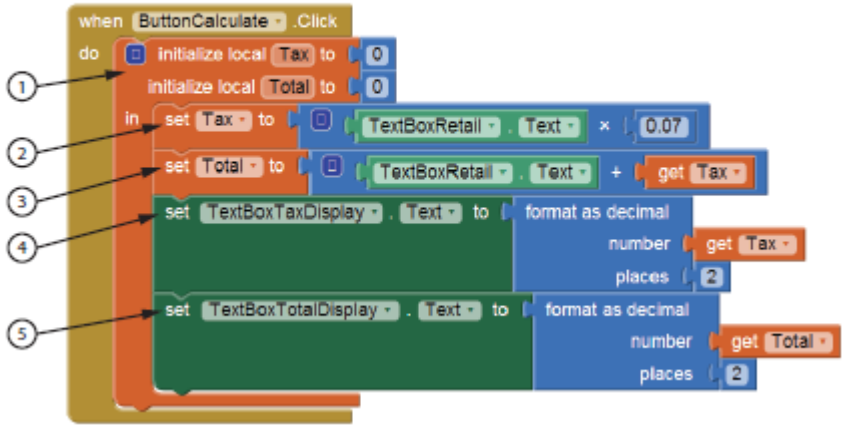


Storing Data with Variables

Creating Multiple Local Variables

The *`Click`* event handler for the *`ButtonReadInput`* component is shown in Figure 3-72.

Figure 3-72 `Click` Event Handler for the `ButtonCalculate` Component
(Source: MIT App Inventor 2)



Storing Data with Variables

Figure 3-72 Described

1. This is an initialization block that creates and initializes two local variables: *Tax* and *Total*.
2. This block sets the *Tax* variable to the value of the *TextBoxRetail.Text* x *0.07*.
3. This block sets the *Tax* variable to the value of the *TextBoxRetail.Text* + the value of the *Tax* variable.
4. This block sets the *TextBoxTaxDisplay.Text* to the value of the *Tax* variable, rounded to two decimal places.
5. This block sets the *TextBoxTotalDisplay.Text* to the value of the *Total* variable, rounded to two decimal places.

Storing Data with Variables

Global Variables

- A global variable is created outside of all methods and functions. It is accessible to all of the code in the workspace.
- Create and initialize a global variable by opening the *Variables* drawer found in the *Built-in* section of the Blocks column.

Figure 3-73 Creating a Global Variable Initialization Block *(Source: MIT App Inventor 2)*

Storing Data with Variables

Global Variables

When you create an *`initialize global name to block`*, you can place it anywhere in the workspace that is not inside a method or function.

Figure 3-75 Global Variable Initialization Block Outside of All Methods
(*Source:* MIT App Inventor 2)

Storing Data with Variables

Global Variables

Once you have created a global variable's initialization block, you need to do two more things:

1. Change the variables name to something that describes the variable's purpose.
2. Assign an initial value to the variable.

To change a variable's name, click the word *name* on the *Initialize global name to* block as shown in Figure 3-76

Figure 3-76 Changing the Name of a Global Variable (Source: MIT App Inventor 2)

Storing Data with Variables

Global Variables

Figure 3-77 has a socket labeled *to*. This socket requires a value to be plugged in. The value that you plug into this socket is the variable's initial value.

Figure 3-77 A Global Variable Named `Population` (Source: MIT App Inventor 2)

Storing Data with Variables

Global Variables

Figure 3-78 shows to global variable initialization blocks.

Figure 3-78 Two Complete Global Variable Initialization Blocks

(*Source:* MIT App Inventor 2)

Once you have created and initialized a global variable, you can use the get block to get the variable's value and the set block to assign value to the variable.

Storing Data with Variables

A Word of Caution About Global Variables

- Most programmers agree that you should restrict the use of global variables.
- Here are some reasons:
  - Global variables make debugging difficult.
  - Global variables make a program hard to understand. A global variable can be modified by any instruction in the program.