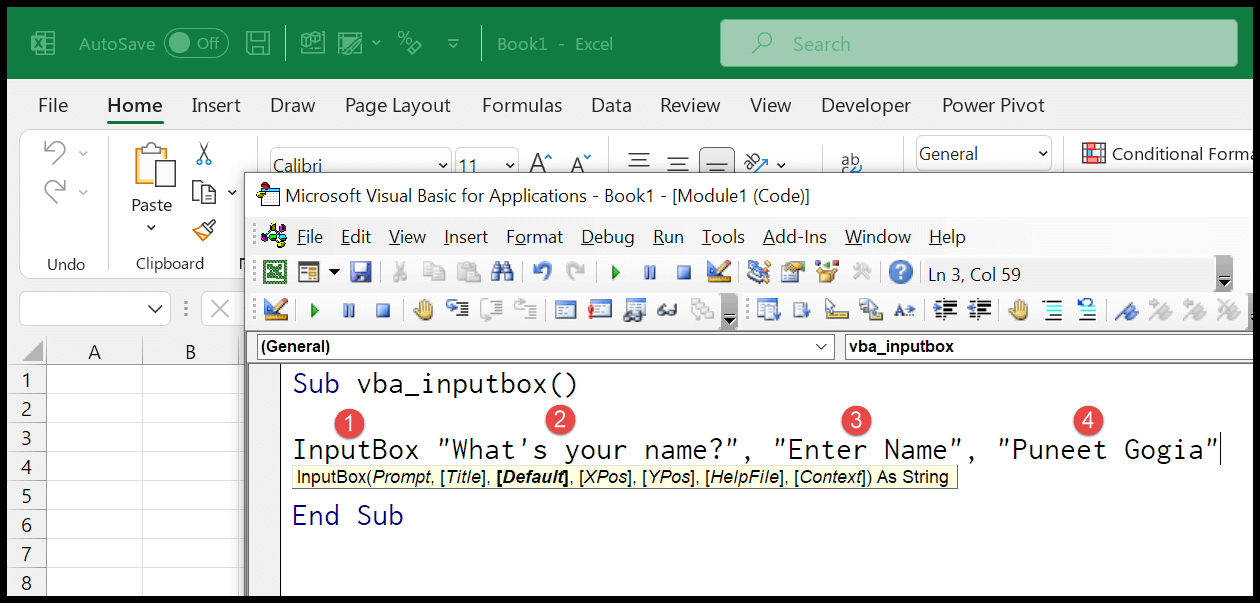
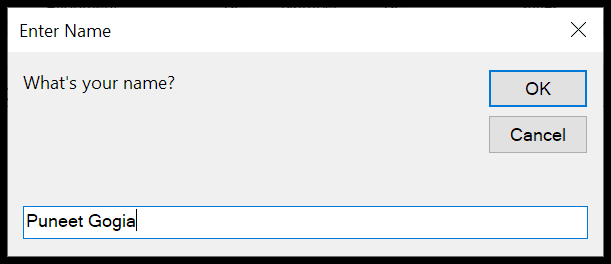
1. Write a VBA code to enter your name in A1 Cell using Input Box and

once you enter the name display a message box that says the name

has been entered.

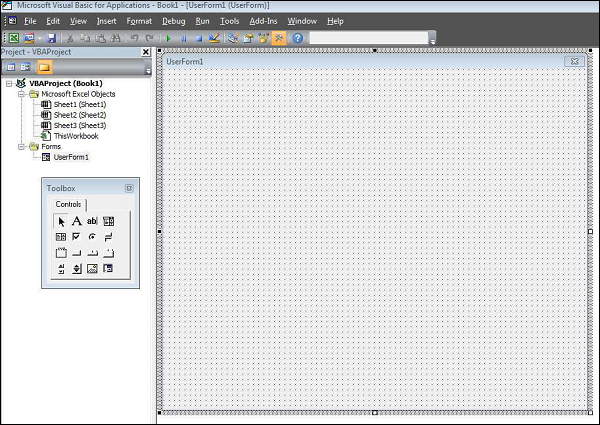
* Type “Input Box” and enter a space and you will get a tool for the arguments you need to define.
* Specify the “Prompt”, message that you want to show to the user.
* Define a title for the input box, otherwise, it will show the default title.
* Mention the text that you want to have in the input bar by default.
* 
* And when you run this code, it will show you an input box just like the following:
* 

2. What are User forms? Why are they used? How to fill a list box using

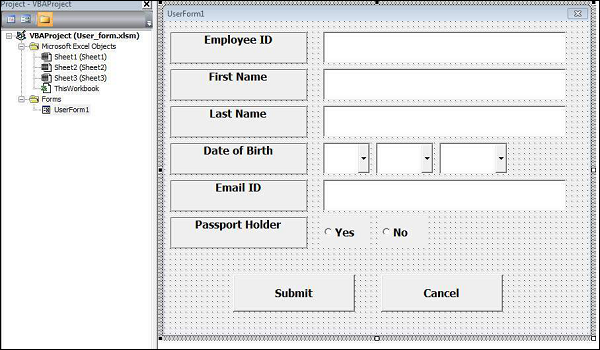
for loop.

A **User Form** is a custom-built dialog box that makes a user data entry more controllable and easier to use for the user. In this chapter, you will learn to design a simple form and add data into excel.

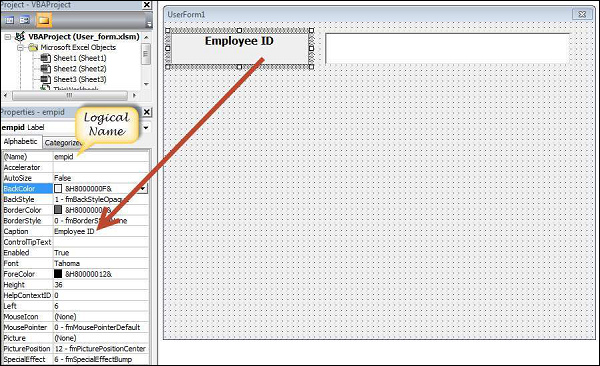
**Step 1** − Navigate to VBA Window by pressing Alt+F11 and Navigate to "Insert" Menu and select "User Form". Upon selecting, the user form is displayed as shown in the following screenshot.



**Step 2** − Design the forms using the given controls.



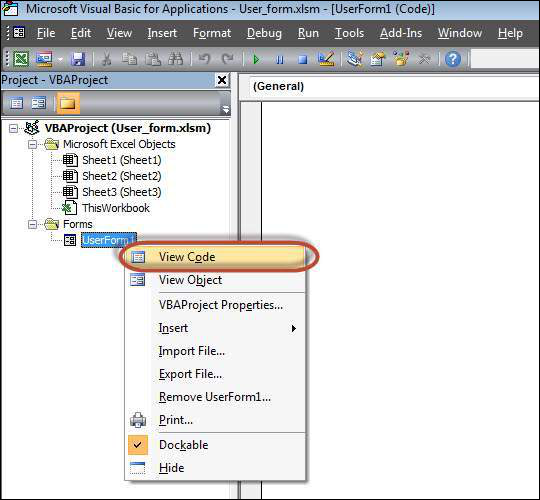
**Step 3** − After adding each control, the controls have to be named. Caption corresponds to what appears on the form and name corresponds to the logical name that will be appearing when you write VBA code for that element.



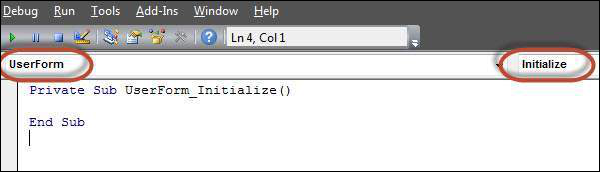
**Step 4** − Following are the names against each one of the added controls.

|  |  |  |
| --- | --- | --- |
| **Control** | **Logical Name** | **Caption** |
| From | frmempform | Employee Form |
| Employee ID Label Box | empid | Employee ID |
| firstname Label Box | firstname | First Name |
| lastname Label Box | lastname | Last Name |
| dob Label Box | dob | Date of Birth |
| mailid Label Box | mailid | Email ID |
| Passportholder Label Box | Passportholder | Passport Holder |
| Emp ID Text Box | txtempid | NOT Applicable |
| First Name Text Box | txtfirstname | NOT Applicable |
| Last Name Text Box | txtlastname | NOT Applicable |
| Email ID Text Box | txtemailid | NOT Applicable |
| Date Combo Box | cmbdate | NOT Applicable |
| Month Combo Box | cmbmonth | NOT Applicable |
| Year Combo Box | cmbyear | NOT Applicable |
| Yes Radio Button | radioyes | Yes |
| No Radio Button | radiono | No |
| Submit Button | btnsubmit | Submit |
| Cancel Button | btncancel | Cancel |

**Step 5** − Add the code for the form load event by performing a right-click on the form and selecting 'View Code'.



**Step 6** − Select ‘Userform’ from the objects drop-down and select 'Initialize' method as shown in the following screenshot.



**Step 7** − Upon Loading the form, ensure that the text boxes are cleared, drop-down boxes are filled and Radio buttons are reset.

Private Sub UserForm\_Initialize()

'Empty Emp ID Text box and Set the Cursor

txtempid.Value = ""

txtempid.SetFocus

'Empty all other text box fields

txtfirstname.Value = ""

txtlastname.Value = ""

txtemailid.Value = ""

'Clear All Date of Birth Related Fields

cmbdate.Clear

cmbmonth.Clear

cmbyear.Clear

'Fill Date Drop Down box - Takes 1 to 31

With cmbdate

.AddItem "1"

.AddItem "2"

.AddItem "3"

.AddItem "4"

.AddItem "5"

.AddItem "6"

.Add Item "7"

.AddItem "8"

.AddItem "9"

.AddItem "10"

.AddItem "11"

.AddItem "12"

.AddItem "13"

.AddItem "14"

.AddItem "15"

.AddItem "16"

.AddItem "17"

.AddItem "18"

.AddItem "19"

.AddItem "20"

.AddItem "21"

.AddItem "22"

.AddItem "23"

.AddItem "24"

.AddItem "25"

.AddItem "26"

.AddItem "27"

.AddItem "28"

.AddItem "29"

.AddItem "30"

.AddItem "31"

End With

'Fill Month Drop Down box - Takes Jan to Dec

With cmbmonth

.AddItem "JAN"

.AddItem "FEB"

.AddItem "MAR"

.AddItem "APR"

.AddItem "MAY"

.AddItem "JUN"

.AddItem "JUL"

.AddItem "AUG"

.AddItem "SEP"

.AddItem "OCT"

.AddItem "NOV"

.AddItem "DEC"

End With

'Fill Year Drop Down box - Takes 1980 to 2014

With cmbyear

.AddItem "1980"

.AddItem "1981"

.AddItem "1982"

.AddItem "1983"

.AddItem "1984"

.AddItem "1985"

.AddItem "1986"

.AddItem "1987"

.AddItem "1988"

.AddItem "1989"

.AddItem "1990"

.AddItem "1991"

.AddItem "1992"

.AddItem "1993"

.AddItem "1994"

.AddItem "1995"

.AddItem "1996"

.AddItem "1997"

.AddItem "1998"

.AddItem "1999"

.AddItem "2000"

.AddItem "2001"

.AddItem "2002"

.AddItem "2003"

.AddItem "2004"

.AddItem "2005"

.AddItem "2006"

.AddItem "2007"

.AddItem "2008"

.AddItem "2009"

.AddItem "2010"

.AddItem "2011"

.AddItem "2012"

.AddItem "2013"

.AddItem "2014"

End With

'Reset Radio Button. Set it to False when form loads.

radioyes.Value = False

radiono.Value = False

End Sub

**Step 8** − Now add the code to the Submit button. Upon clicking the submit button, the user should be able to add the values into the worksheet.

Private Sub btnsubmit\_Click()

Dim emptyRow As Long

'Make Sheet1 active

Sheet1.Activate

'Determine emptyRow

emptyRow = WorksheetFunction.CountA(Range("A:A")) + 1

'Transfer information

Cells(emptyRow, 1).Value = txtempid.Value

Cells(emptyRow, 2).Value = txtfirstname.Value

Cells(emptyRow, 3).Value = txtlastname.Value

Cells(emptyRow, 4).Value = cmbdate.Value & "/" & cmbmonth.Value & "/" & cmbyear.Value

Cells(emptyRow, 5).Value = txtemailid.Value

If radioyes.Value = True Then

Cells(emptyRow, 6).Value = "Yes"

Else

Cells(emptyRow, 6).Value = "No"

End If

End Sub

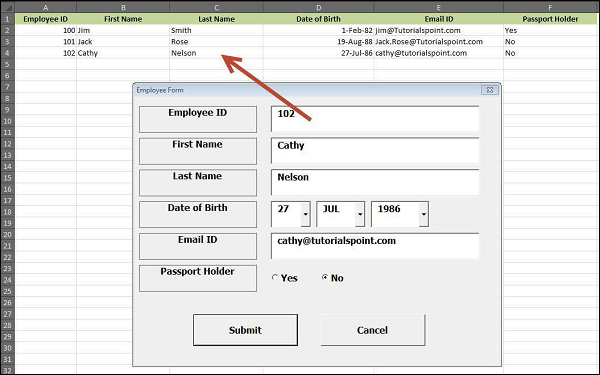
**Step 9** − Add a method to close the form when the user clicks the Cancel button.

Private Sub btncancel\_Click()

Unload Me

End Sub

**Step 10** − Execute the form by clicking the "Run" button. Enter the values into the form and click the 'Submit' button. Automatically the values will flow into the worksheet as shown in the following screenshot.



3. What is an array? Write a VBA code to enter students and their marks

from the below table.

**A VBA array** is a type of variable. It is used to store lists of data of the same type. An example would be storing a list of countries or a list of weekly totals.

In VBA a normal variable can store only one value at a time.

In the following example we use a variable to store the marks of a student:

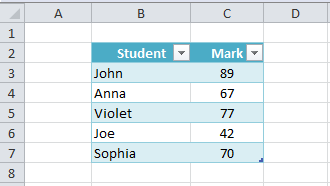
' Can only store 1 value at a time

**Dim** Student1 **As** **Long**

Student1 = 55

If we wish to store the marks of another student then we need to create a second variable.

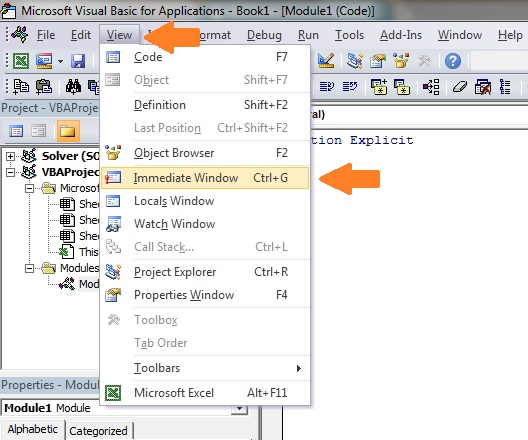
In the following example, we have the marks of five students:

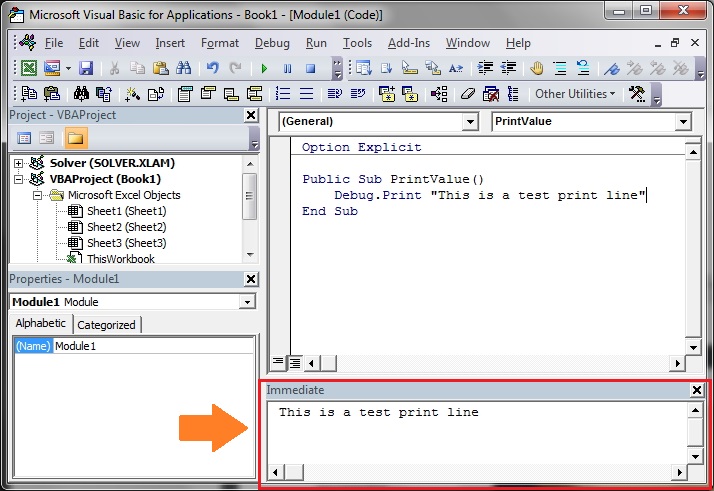
[](https://excelmacromastery.com/wp-content/uploads/2015/01/StudMarksSheet.png)

**Student Marks**

We are going to read these marks and write them to the Immediate Window.

**Note:** The function *Debug.Print* writes values to the Immediate  Window. To view this window select View->Immediate Window from the menu( Shortcut is Ctrl + G)

[](https://excelmacromastery.com/wp-content/uploads/2014/12/ImmediateWindow2.jpg)

[](https://excelmacromastery.com/wp-content/uploads/2014/12/ImmediateSampeText.jpg)

As you can see in the following example we are writing the same code five times – once for each student:

' https://excelmacromastery.com/

**Public** **Sub** StudentMarks()

' Get the worksheet called "Marks"

**Dim** sh **As** Worksheet

**Set** sh = ThisWorkbook.Worksheets("Marks")

' Declare variable for each student

**Dim** Student1 **As** **Long**

**Dim** Student2 **As** **Long**

**Dim** Student3 **As** **Long**

**Dim** Student4 **As** **Long**

**Dim** Student5 **As** **Long**

' Read student marks from cell

Student1 = sh.Range("C" & 3).Value

Student2 = sh.Range("C" & 4).Value

Student3 = sh.Range("C" & 5).Value

Student4 = sh.Range("C" & 6).Value

Student5 = sh.Range("C" & 7).Value

' Print student marks

**Debug.Print** "Students Marks"

**Debug.Print** Student1

**Debug.Print** Student2

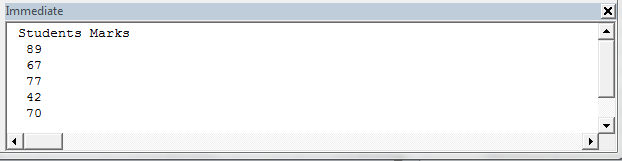
**Debug.Print** Student3

**Debug.Print** Student4

**Debug.Print** Student5

**End** **Sub**

The following is the output from the example:

[](https://excelmacromastery.com/wp-content/uploads/2015/01/StuMarksOutput.png)

**Output**

The problem with using one variable per student is that you need to add code for each student. Therefore if you had a thousand students in the above example you would need three thousand lines of code!

Luckily we have arrays to make our life easier. Arrays allow us to store a list of data items in one structure.

The following code shows the above student example using an array:

' ExcelMacroMastery.com

' https://excelmacromastery.com/excel-vba-array/

' Author: Paul Kelly

' Description: Reads marks to an Array and write

' the array to the Immediate Window(Ctrl + G)

' TO RUN: Click in the sub and press F5

**Public** **Sub** StudentMarksArr()

' Get the worksheet called "Marks"

**Dim** sh **As** Worksheet

**Set** sh = ThisWorkbook.Worksheets("Marks")

' Declare an array to hold marks for 5 students

**Dim** Students(1 **To** 5) **As** **Long**

' Read student marks from cells C3:C7 into array

' Offset counts rows from cell C2.

' e.g. i=1 is C2 plus 1 row which is C3

' i=2 is C2 plus 2 rows which is C4

**Dim** i **As** **Long**

**For** i = 1 **To** 5

Students(i) = sh.Range("C2").Offset(i).Value

**Next** i

' Print student marks from the array to the Immediate Window

**Debug.Print** "Students Marks"

**For** i = LBound(Students) **To** UBound(Students)

**Debug.Print** Students(i)

**Next** i

**End** **Sub**

The advantage of this code is that it will work for any number of students. If we have to change this code to deal with 1000 students we only need to change the **(1 To 5)** to **(1 To 1000)** in the declaration. In the prior example we would need to add approximately five thousand lines of code.

Let’s have a quick comparison of variables and arrays. First we compare the declaration:

' Variable

**Dim** Student **As** **Long**

**Dim** Country **As** **String**

' Array

**Dim** Students(1 **To** 3) **As** **Long**

**Dim** Countries(1 **To** 3) **As** **String**

Next we compare assigning a value:

' assign value to variable

Student1 = .Cells(1, 1)

' assign value to first item in array

Students(1) = .Cells(1, 1)

Finally we look at writing the values:

' Print variable value

**Debug.Print** Student1

' Print value of first student in array

**Debug.Print** Students(1)

As you can see, using variables and arrays is quite similar.

The fact that arrays use an index(also called a subscript) to access each item is important. It means we can easily access all the items in an array using a For Loop

6.Write step by step procedure to protect your workbook using a

password.

* Select **File** > **Info**.
* Select the **Protect Workbook**box and choose **Encrypt with Password.**
* Enter a password in the **Password**box, and then select **OK**.
* Confirm the password in the **Reenter Password**box, and then select **OK**.