**Module 4: Working with Procedures & Functions**

* Sub Procedures vs Function Procedures

In **VBA (Visual Basic for Applications)**, both **Sub Procedures** and **Function Procedures** are used to organize reusable code. But they serve different purposes:

**1. Sub Procedures**

* Perform an action (like calculations, formatting, displaying messages).
* Do not return a value.
* Called using Call statement or just the procedure name.

Example:

Sub GreetUser()

Dim userName As String

userName = Range("B1").Value ' Read value from Excel cell B1

MsgBox "Hello, " & userName & "!"

End Sub

👉 When run, this **displays a greeting** but doesn’t return anything.

**2. Function Procedures**

* Perform an action and return a value.
* Used when you need a result for further calculations.
* Called from VBA code or directly in Excel cells.

Example:

Function SquareNumber(num As Double) As Double

SquareNumber = num \* num

End Function

Sub TestSquare()

Dim result As Double

result = SquareNumber(5) ' Call function

MsgBox "Square is: " & result

End Sub

**Summary:**

* Use a **Sub** when you just want to perform an action.
* Use a **Function** when you need a **value returned** (like formulas).

* **Calling Procedures**

**1. Calling a Sub Procedure**

**(a) Directly by its name**

Sub DisplayMessage()

MsgBox "Hello, VBA!"

End Sub

**(b) Using Call keyword**

Sub TestCall2()

Call DisplayMessage

End Sub

**(c) From Excel’s Macro Menu**

* Press **Alt + F8** → Select Sub name → Run.
* Example: Running DisplayMessage directly.

**2. Calling a Function Procedure**

Unlike Subs, Functions **return values**, so you call them in an **expression**.

**(a) From VBA code**

Function AddNumbers(x As Double, y As Double) As Double

AddNumbers = x + y

End Function

Sub TestFunction()

Dim result As Double

result = AddNumbers(10, 20) ' Function call

MsgBox "The sum is: " & result

End Sub

**(b) Directly in an Excel cell**

If the above Function AddNumbers is in a standard VBA module, you can type in a cell:

=AddNumbers(10,20)

and Excel will display **30**.

**Steps to Create a User Defined Function (UDF)**

1. Open Excel.
2. Press **Alt + F11** → opens the VBA editor.
3. Go to **Insert → Module** (very important – it must be in a **standard module**, not inside Sheet1 or ThisWorkbook).
4. Paste this code:

Function AddNumbers(x As Double, y As Double) As Double

AddNumbers = x + y

End Function

Close the VBA editor.

Go back to Excel → In a cell, type:

=AddNumbers(10,20)

* **User Defined Functions (UDFs)**

A **User Defined Function (UDF)** is a **custom formula** you create in VBA when Excel’s built-in functions are not enough.

* Like built-in functions (=SUM(A1:A5)), but you **define** them yourself.
* Written in VBA inside a **standard Module**.
* Can be used directly in Excel cells.

**Creating a UDF – Step by Step**

1. Open Excel → Press **Alt + F11** (VBA Editor).
2. Go to **Insert → Module**.
3. Write your function. Example:
4. **Add Number**

=AddNumbers(10,20)

➝ Result: 30

**2.) Find Square of a Number**

Function SquareNumber(num As Double) As Double

SquareNumber = num \* num

End Function

=SquareNumber(5) 'Result = 25

**3.) Return Student Result (Pass/Fail)**

Function CheckResult(marks As Double) As String

If marks >= 40 Then

CheckResult = "Pass"

Else

CheckResult = "Fail"

End If

End Function

=CheckResult(A1)

If A1 = 55 → Pass; if A1 = 32 → Fail.

**4.) Calculate Age from DOB**

Function GetAge(DOB As Date) As Integer

GetAge = DateDiff("yyyy", DOB, Date)

If Date < DateSerial(Year(Date), Month(DOB), Day(DOB)) Then

GetAge = GetAge - 1

End If

End Function