MsgBox "I lost my mobile phone! :(", vbCritical, "Oh no!"

> msg, type, window name

Dim Name As String

> Explicitly set Name as type string

Name = InputBox("What is your name?")

Dim VariableName as String \*10

> limit str length to 10

Const Pi As Single=3.142

Const Temp As Double=37

CellContent = Range("B5").Value

MsgBox "The content of cell B5 is '" & CellContent & "'"

Range("A4:C10").ClearContents

Cells(4, 3).Value = “”

> Cell C4

Selection.Interior.Color = vbRed

Worksheets("Sheet2").Activate

Worksheets("Sheet2").Range("A1").Select

Worksheets("Sheet2").Cells(2, 2).Value = "Hello“

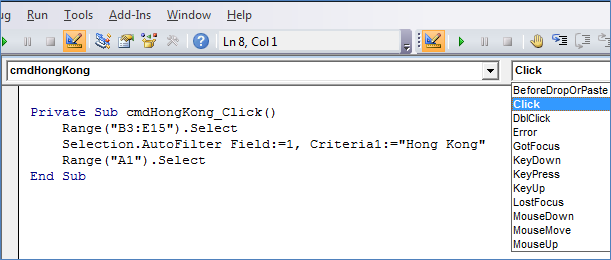
ActiveSheet.Cells.Font.Size = 20

**Range("C5:E14").Select**

**Selection.Copy**

**Range("C20").Select**

**ActiveSheet.Paste**



**Private Sub Workbook\_Open()**

**Worksheets("Welcome").Visible = True**

**Worksheets("Input").Visible = False**

**Worksheets("Analysis").Visible = False**

**Worksheets("Output").Visible = False**

**End Sub**



 A close up of a text

Description automatically generated A black text on a white background

Description automatically generated

A white text with black text

Description automatically generated

R1C1 => R[]C[]

Dim wb As Workbook

Dim ws As Worksheet

Set wb = ActiveWorkbook

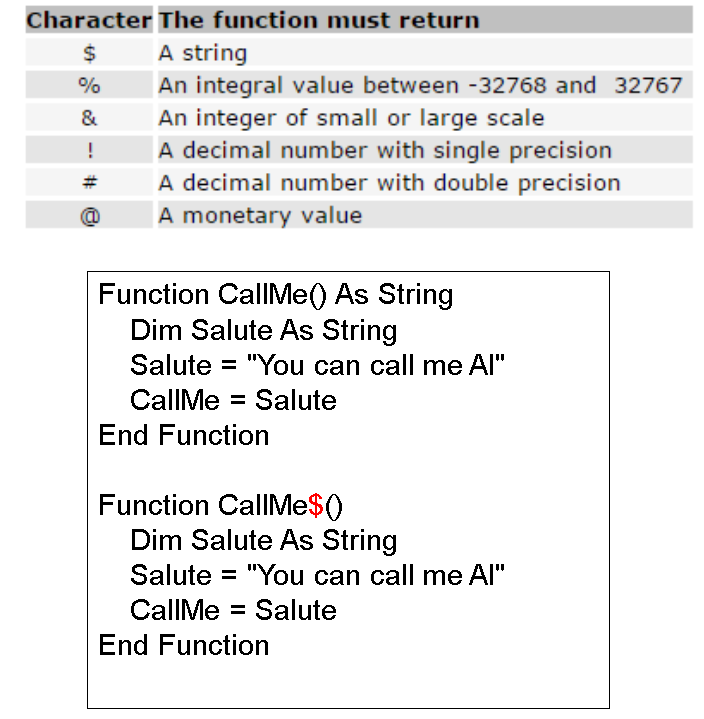
Set ws = Sheets("Sheet1")

wb.Activate

ws.Select

GetFullName = LastName & ", " & FirstName

& => Concatonation



A white background with black text and red and blue letters

Description automatically generated  
return sumNo cuz sumNo same var name as function name (???)

Sub MySubWithArguments(ByVal Arg1 As Integer, ByVal Arg2 As String)

MySubWithArguments 42, "Hello, World!"

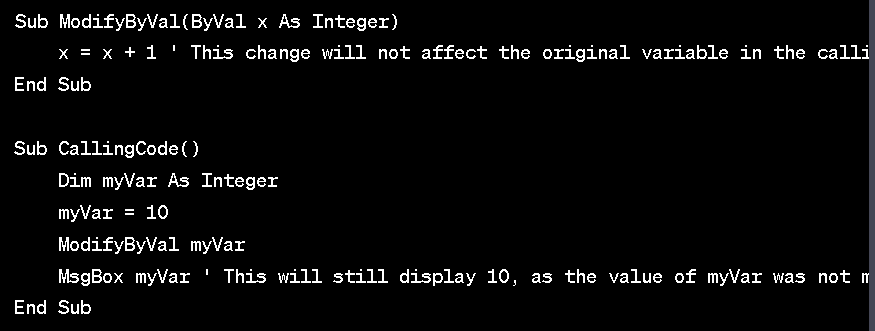
Sub newSub()

End Sub

If sub no argument, => Call newSub

By default, the argument will be passed as a reference (ByRef)

If you want to pass an argument by value (i.e., create a copy of the variable's value and work with that copy inside the Sub), you can specify "ByVal" explicitly:



Dim rng As range

Set rng = ThisWorkbook.Worksheets("Sheet1").range("C10")

rng.value = InputBox("Enter Sales for January")

rng.offset(-1, 0).value = "January Sales"

> “January Sales” will appear on C9

*Dim MyRange As Range*

*Set MyRange = Range(“A1”)*

Class Dog =>

Public Name As String

Public Color As Long

Public Weight As Double

Sub bark(ByVal Woof As String)

MsgBox Woof, , Name

End Sub

Sub eat()

Weight = Weight + 1

If Weight >= 35 Then

MsgBox "Oh dear, I'm full."

End If

End Sub

Sub createDogObject()

Dim snoopy As Dog

Set snoopy = New Dog

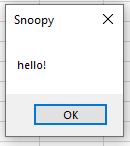
snoopy.Name = "Snoopy"

snoopy.Color = vbWhite

snoopy.Weight = 30

snoopy.bark "hello!"

End Sub



Dim DogToFeed As String

Do

DogToFeed = InputBox("Feed which dog?")

If DogToFeed = "Snoopy" Then

Snoopy.Eat

ElseIf DogToFeed = "Odie" Then

Odie.Eat

End If

Loop Until DogToFeed = "“

> Loop examples >

Sub GetInput()

Dim response As String

Dim isValid As Boolean

**Do**

isValid = True

response = InputBox("Enter the unit cost for the product.")

If Not IsNumeric(response) Then

isValid = False

MsgBox "Please enter a numeric value.", vbInformation, "Invalid entry"

ElseIf response <= 0 Then

isValid = False

MsgBox "Please enter a positive value.", vbInformation, "Invalid entry"

Else

MsgBox "Thanks."

End If

**Loop Until isValid**

End Sub

* **Sequence**
* **Selection**
  + If…Then
  + If…Then…Else
  + Select Case
* **Repetition**
  + For…Next
  + For Each…Next
  + Do While…Loop
  + Do Until…Loop

response = MsgBox("Do you want to enter another value?", vbYesNo)

If response = vbYes Then

RecordInput

End If

> vbYes or whatever it is

If x > -5 And x <= 5 Then

If x <= -5 Or x > 5 Then

> And / Or inside if statements

Select Case x

Case Is > 0: sig = "positive"

Case Is < 0: sig = "negative"

Case Else: sig = "zero"

End Select

> Switch case with < >Select Case age

Case 0 To 5, Is > 65: entry = 0

Case 6 To 15: entry = 2

Case 15 To 65: entry = 5

Case Else: entry = "Age not valid!"

End Select

> Switch case with range

Select Case product

Case "Mangoes": price = 2.5

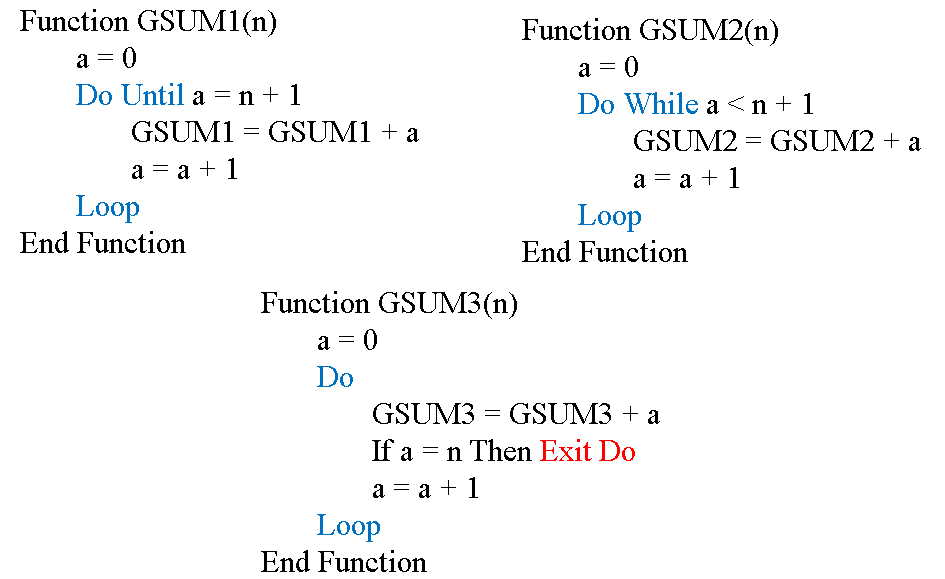
Case "Bananas": price = 1.8

Case "Pears", "Apples": price = 0.9

Case Else: price = "Fruit not in price list!"

End Select

> Switch case with str



Sub DoWhile\_Loop()

' Select the first cell in the 4th row

Cells(4, 1).Select

Do While ActiveCell.Row <= 10

' Put the row number into the first cell

ActiveCell.Value = "I am row " & ActiveCell.Row

' Move the selection down one cell

ActiveCell.Offset(1, 0).Select

Loop

End Sub

> Do while use case (better example than the one before)

Function GSUMNEXT1(n)

For a = 1 To n

GSUMNEXT = GSUMNEXT + a

Next a

End Function

Function GSUMNEXT2(n)

For a = 2 To 2\*n Step 2

GSUMNEXT2 = GSUMNEXT2 + a

Next a

End Function

> For loop

Dim Cell As Range

For Each Cell In ActiveSheet.Range("A1:A10")

Cell.Interior.Color = RGB(160, 251, 142)

Next Cell

> For each

Sub While\_Wend()

' Select the first cell in the 4th row

Cells(4, 1).Select

While ActiveCell.Row <= 10

' Put the row number into the first cell

ActiveCell.Value = "I am row " & ActiveCell.Row

' Move the selection down one cell

ActiveCell.Offset(1, 0).Select

Wend

End Sub

> Whatever the fuck a while wend is

Option Explicit

Option Base 1

Sub Days()

Dim Day(7) As String, iDay As Integer

Day(1) = "Monday"

Day(2) = "Tuesday"

Day(3) = "Wednesday"

Day(4) = "Thursday"

Day(5) = "Friday"

Day(6) = "Saturday"

Day(7) = "Sunday"

With Worksheets(1).Range("A1")

For iDay = 1 To 7

.Offset(0, iDay - 1).Value = Day(iDay)

Next

End With

End Sub

> Array

Option Explicit

Option Base 1

Sub DaysAlternative()

Dim Day As Variant, iDay As Integer

' Here's an alternative way to populate the array, but note that Day

' has to be declared as Variant type to do it this way.

Day = Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", \_

"Saturday", "Sunday")

With Worksheets(1).Range("A1")

For iDay = 1 To 7

.Offset(0, iDay - 1).Value = Day(iDay-1)

Next

End With

End Sub

> Creating Array with arguments

**Dim uInput() As Double**

Dim Size As Integer

Size = InputBox("How many values will you enter as input?")

**ReDim uInput(1 To Size) As Double**

> ReDim changes the size of the array

ReDim Preserve uInput(1 To Size + Size2)

> If you want to change the size of an array but do not want to reset its values, then use the ***ReDim Preserve*** statement.