VBA FILES & FOLDERS
COMPLETE GUIDE

1. What is File & Folder Handling in VBA?

When automating tasks in Excel, we often need to interact with files and folders on the computer. VBA lets us:

- Check if a file or folder exists
- Create, delete, or rename folders
- Loop through files in a folder
- Open or save files programmatically

VBA provides **two main approaches** for file & folder operations:

Approach	Description
Built-in VBA Functions Dir, MkDir, Kill, Name…	No extra reference needed. Simple and lightweight for basic tasks.
FileSystemObject (FSO) Scripting Runtime library	More powerful and object-oriented. Requires enabling Microsoft Scripting Runtime.

2. Key Built-in Functions

These functions are available in VBA without any extra library:

Function	Purpose
`Dir(path)`	Returns filename/folder name if it exists; "" if not found
`MkDir path`	Creates a new folder at the given path
`Kill path`	Permanently deletes a file (does NOT go to Recycle Bin)
`Name old As new`	Renames or moves a file
`FileCopy src, dest`	Copies a file from one location to another

3. VBA Files & Folders Examples

Example 1: Check if a File Exists

```
Sub CheckFileExists()
Dim filePath As String
filePath = "C:\Users\Akshay\OneDrive\Documents\Datafiles.xlsx"

If Dir(filePath) <> "" Then
```

```
MsgBox "File exists: " & filePath, vbInformation
Else
MsgBox "File not found!", vbExclamation
End If
End Sub
```

Explanation:

- **Dir(filePath)** — attempts to find the file; returns its name if found, "" if not
- **<> ""** — condition checks if Dir returned something (file exists)
- **vbInformation** — shows a blue ■ icon in the MsgBox
- **vbExclamation** — shows a yellow ■ warning icon in the MsgBox

Result: A message box tells the user whether the file was found or not.

Example 2: Loop Through All Files in a Folder

```
Sub ListFilesInFolder()
Dim folderPath As String, fileName As String
folderPath = "C:\Users\Akshay\OneDrive\Documents\"

fileName = Dir(folderPath & "*.xlsx") 'first Excel file
Do While fileName <> ""
Debug.Print fileName 'prints in Immediate Window
fileName = Dir 'next file
Loop
End Sub
```

Explanation:

- **Dir(folderPath & "*.xlsx")** — returns the first .xlsx file found in the folder
- **"*.xlsx"** — wildcard pattern; * means any name, .xlsx is the extension
- **Do While fileName <> ""** — keeps looping until no more files are found
- **Debug.Print fileName** — prints each file name in the Immediate Window (Ctrl+G)
- **fileName = Dir** — calling Dir() with no argument fetches the next matching file

Result: All Excel file names in the folder are printed in the Immediate Window.

Example 3: Create a New Folder

```
Sub CreateFolder()
Dim folderPath As String
folderPath = "C:\Users\Akshay\OneDrive\Documents\NewFolder"
```

```
If Dir(folderPath, vbDirectory) = "" Then
MkDir folderPath
MsgBox "Folder created successfully!"
Else
MsgBox "Folder already exists!"
End If
End Sub
```

Explanation:

- **Dir(path, vbDirectory)** — the vbDirectory flag makes Dir look for folders, not just files
- **MkDir folderPath** — creates the folder at the specified path
- Always check before creating to avoid a runtime error if folder already exists

Result: Folder is created only if it does not already exist.

Example 4: Delete a File

```
Sub DeleteFile()
Dim filePath As String
filePath = "C:\Users\Akshay\OneDrive\Documents\OldFile.xlsx"

If Dir(filePath) <> "" Then
Kill filePath
MsgBox "File deleted!"
Else
MsgBox "File not found!"
End If
End Sub
```

Explanation:

- Always check if the file exists before using **Kill**
- **Kill** permanently deletes the file — it does **NOT** go to the Recycle Bin

Result: File is permanently deleted if found. A message confirms the action.

4. Dir() Function – Key Behaviours

Usage	What it Returns
`Dir(filePath)`	Filename if found; "" if not found
`Dir(path & "*.xlsx")`	First .xlsx file in the folder
`Dir(path, vbDirectory)`	Folder name if exists; "" if not

`Dir() – no argument`	Next file matching the previous pattern

5. Best Practices for File & Folder Handling

1. Always check if a file or folder exists before accessing, deleting, or opening it
2. Use **On Error GoTo** to handle unexpected errors (e.g. file in use, no permission)
3. Make sure folder paths end with a backslash (****) when using Dir with wildcards
4. Use **Kill** carefully — deleted files **cannot** be recovered from the Recycle Bin
5. For complex operations, use **FileSystemObject (FSO)** for more power and control
6. Always close open workbooks before deleting or overwriting them

6. Quick Reference Table

Statement	Purpose
`Dir(path)`	Check if a file exists
`Dir(path, vbDirectory)`	Check if a folder exists
`Dir(path & "*.xlsx")`	Find first matching file in a folder
`Dir() – no argument`	Get next matching file in a loop
`MkDir path`	Create a new folder
`Kill path`	Delete a file permanently
`Name old As new`	Rename or move a file
`FileCopy src, dest`	Copy a file to a new location

Note: All code examples above use **Option Explicit** which requires all variables to be declared before use. File and folder handling is essential for automating real-world VBA tasks. Always validate paths and handle errors to make your macros reliable.