Here's the detailed **line-by-line explanation** of the provided Python code:

Imports

```
import os
import shutil
```

- `os`: Provides functions to interact with the operating system (e.g., file paths, checking directories, listing files).
- `shutil`: Provides high-level file operations like copying, moving, or removing files and directories.

Global Dictionary: `FILE_CATEGORIES`

```
FILE_CATEGORIES = {
    "Images": [".jpg", ".jpeg", ".png", ".gif"],
    "Videos": [".mp4", ".mkv", ".flv", ".avi"],
    "Documents": [".pdf", ".docx", ".doc", ".txt"],
    "Music": [".mp3", ".wav", ".aac", ".flac"],
    "Archives": [".zip", ".rar", ".7z", ".tar"],
    "Code": [".py", ".java", ".cpp", ".js", ".html"],
    "Others": []  # Any other file types will be moved here
}
```

- This dictionary maps file categories (e.g., Images, Videos) to a list of corresponding file extensions.
- Files will be sorted into these categories based on their extensions.
- The `"Others"` key is for any files that don't match the listed extensions.

Function: `organize_directory`

```
def organize_directory(directory_path):
```

- This function takes a **directory path** as input and organizes all files in that directory into subfolders based on their file type.

Check if the Directory Exists

```
    if not os.path.exists(directory_path):
        print(f"Error: The directory '{directory_path}' does not exist.")
        return
```

- Checks if the provided directory path exists using `os.path.exists()`.
- If the directory does not exist, it prints an error message and exits the function.

Create Category Folders

```
    for category in FILE_CATEGORIES.keys():
        category_folder = os.path.join(directory_path, category)
        if not os.path.exists(category_folder):
            os.mkdir(category_folder)
```

- Loops through all the category names (keys of `FILE_CATEGORIES`).
- Constructs the path for each category folder using `os.path.join()`.
- Checks if the folder already exists using `os.path.exists()`.
- If the folder does not exist, creates it using `os.mkdir()`.

Iterate Through Files in the Directory

```
    for item in os.listdir(directory_path):
        item_path = os.path.join(directory_path, item)
```

- `os.listdir(directory_path)` returns a list of files and folders in the given directory.
- `item_path` is the full path of each item (file or folder).

Skip Folders

```
        if os.path.isdir(item_path):
            continue
```

- Checks if the current `item_path` is a directory using `os.path.isdir()`.
- If it is a directory, the code skips to the next item in the loop.

Get File Extension

```
        _, file_extension = os.path.splitext(item)
        file_extension = file_extension.lower()
```

- `os.path.splitext(item)` splits the file name into two parts: the name and the extension.
- `file_extension` stores the extension of the file (e.g., `.jpg`).
- `.lower()` ensures the extension is in lowercase for consistent comparison.

Move Files to Appropriate Folders

```
        moved = False
        for category, extensions in FILE_CATEGORIES.items():
            if file_extension in extensions:
                shutil.move(item_path, os.path.join(category_folder, item))
                moved = True
                break
```

- `moved = False`: A flag to track whether the file has been moved to a category folder.
- Iterates through each `category` and its `extensions` in `FILE_CATEGORIES`.
- If the `file_extension` matches any extension in the category, the file is moved to that category folder using `shutil.move()`.
- `moved = True`: Indicates the file was successfully moved, and the loop breaks.

Move Unmatched Files to "Others"

```
        if not moved:
            shutil.move(item_path, os.path.join(others_folder, item))
```

- If the file was not moved to any category (i.e., no matching extension), the file is moved to the "Others" folder.

Print Completion Message

```
    print(f"Files in '{directory_path}' have been organized.")
```

- Prints a confirmation message that the files have been organized successfully.

Main Block

```
if __name__ == "__main__":
    dir_path = input("Enter the path of the directory to organize: ")
    organize_directory(dir_path)
```

- `if __name__ == "__main__"`: ensures the code runs only when executed directly (not when imported as a module).
- Prompts the user to enter a directory path.
- Passes the entered path to the `organize_directory` function.

How It Works:

1. **Input**:
   - The user provides the directory path they want to organize.
2. **Folder Creation**:
   - Subfolders named after categories (Images, Videos, etc.) are created if they don't already exist.
3. **File Organization**:
   - The program iterates through all the files in the directory.
   - Based on the file extension, files are moved into the corresponding category folder.
   - Files with unrecognized extensions are moved to the "Others" folder.
4. **Output**:
   - The directory is organized into subfolders, and a success message is displayed.

Example

Before:

```
/Downloads
    photo.jpg
    movie.mp4
    document.pdf
    song.mp3
    archive.zip
    script.py
    unknownfile.xyz
```

After Running the Script:

```
/Downloads
    Images/
        photo.jpg
    Videos/
        movie.mp4
    Documents/
        document.pdf
    Music/
        song.mp3
    Archives/
        archive.zip
    Code/
        script.py
    Others/
        unknownfile.xyz
```

Uses of the Script

1. **Organizing Files**:
   - Useful for cleaning up cluttered directories like `Downloads`, `Desktop`, or project folders.
2. **Automation**:
   - Saves time by automating the manual task of sorting files into categories.
3. **Customizable**:
   - Easily add more categories or extensions to tailor the script to your needs.

This script is efficient, simple, and a great example of using Python for file automation tasks!