1. How do you distinguish between shutil.copy() and shutil.copytree()?

Ans:

shutil.copy() and shutil.copytree() are both functions from the shutil module in Python used for copying files and directories, but they serve different purposes:

shutil.copy(src, dst):

This function is used to copy a single file from the src path to the dst path.

It creates a new file at the destination path with the contents of the source file.

If the destination file already exists, it will be overwritten.

The src and dst arguments are strings representing the paths of the source file and the destination file, respectively.

Example:

import shutil

# Copy a file from source path to destination path

shutil.copy('source\_file.txt', 'destination\_file.txt')

shutil.copytree(src, dst):

This function is used to recursively copy an entire directory tree from the src path to the dst path.

It creates a new directory at the destination path and copies all files and subdirectories from the source directory into it.

If the destination directory already exists, it will raise a FileExistsError unless the dirs\_exist\_ok argument is set to True.

The src and dst arguments are strings representing the paths of the source directory and the destination directory, respectively.

Example:

import shutil

# Copy a directory tree from source path to destination path

shutil.copytree('source\_directory', 'destination\_directory')

2. What function is used to rename files??

Ans:

In Python, the os.rename() function is used to rename files and directories. It is part of the os module, which provides a way to interact with the operating system.

3. What is the difference between the delete functions in the send2trash and shutil modules?

Ans:

The send2trash and shutil modules both provide functions for deleting files and directories, but they differ in their behavior and purpose:

send2trash module:

The send2trash module is primarily used for sending files and directories to the system's trash or recycle bin, rather than permanently deleting them.

It provides the send2trash.send2trash() function, which moves the specified file or directory to the trash or recycle bin instead of permanently deleting it.

This module is safer than shutil for deleting files, as it allows users to recover accidentally deleted files from the trash.

It is cross-platform and works on Windows, macOS, and Linux systems.

Example:

import send2trash

# Send a file or directory to the trash

send2trash.send2trash('filename.txt')

shutil module:

The shutil module provides functions for file operations, including copying, moving, and deleting files and directories.

It provides the shutil.rmtree() function, which is used to recursively delete a directory and all its contents.

Unlike send2trash, shutil.rmtree() permanently deletes files and directories, bypassing the system's trash or recycle bin.

This module is more suitable for cases where you want to permanently remove files and directories from the filesystem.

Example:

import shutil

# Permanently delete a directory and all its contents

shutil.rmtree('directory\_name')

4.ZipFile objects have a close() method just like File objects’ close() method. What ZipFile method is equivalent to File objects’ open() method?

Ans:

The equivalent method in ZipFile objects to the open() method in file objects is the ZipFile() constructor itself.

When you want to work with a ZIP file, you typically create a ZipFile object using its constructor to open the ZIP file. This is similar to how you use the open() function to open a regular file.

Here's an example:

from zipfile import ZipFile

# Open a ZIP file in read mode

with ZipFile('example.zip', 'r') as zip\_file:

# Do something with the ZIP file

pass

In this example, ZipFile('example.zip', 'r') opens the file named 'example.zip' in read mode, similar to how open('example.txt', 'r') would open a text file in read mode. After you've finished working with the ZIP file, you can use the close() method or the context manager (with statement) to ensure that the file is properly closed and resources are released.

5. Create a programme that searches a folder tree for files with a certain file extension (such as .pdf or .jpg). Copy these files from whatever location they are in to a new folder.

Ans:

You can accomplish this task using the os module to traverse the folder tree and shutil module to copy files. Here's a Python program to search for files with a specific file extension (e.g., .pdf or .jpg) within a folder tree and copy them to a new folder:

import os

import shutil

def copy\_files\_with\_extension(source\_folder, destination\_folder, extension):

# Create the destination folder if it doesn't exist

os.makedirs(destination\_folder, exist\_ok=True)

# Walk through the source folder tree

for root, \_, files in os.walk(source\_folder):

for file in files:

# Check if the file has the specified extension

if file.endswith(extension):

# Get the full path of the source file

source\_file\_path = os.path.join(root, file)

# Create the destination file path

destination\_file\_path = os.path.join(destination\_folder, file)

try:

# Copy the file to the destination folder

shutil.copy2(source\_file\_path, destination\_file\_path)

print(f"Copied '{source\_file\_path}' to '{destination\_file\_path}'")

except IOError as e:

print(f"Error copying '{source\_file\_path}': {e}")

# Example usage:

source\_folder = '/path/to/source/folder' # Specify the source folder

destination\_folder = '/path/to/destination/folder' # Specify the destination folder

extension = '.pdf' # Specify the file extension to search for (e.g., '.pdf' or '.jpg')

copy\_files\_with\_extension(source\_folder, destination\_folder, extension)

Replace /path/to/source/folder with the path to the folder where you want to start the search, /path/to/destination/folder with the path to the destination folder where you want to copy the files, and .pdf with the desired file extension.