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

Ans - In Python, the shutil module provides two functions for copying files and directories: shutil.copy() and shutil.copytree().

shutil.copy(src, dst):

Used for copying a single file from the source (src) to the destination (dst).

If dst is a directory, the file is copied into that directory with the same base name.

If dst is a file path, the file is copied with the specified new name.

import shutil

# Copy a file to a new location

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

shutil.copytree(src, dst):

Used for recursively copying an entire directory tree from the source (src) to the destination (dst).

Creates a new directory at the destination with the same base name as the source directory and copies all files and subdirectories into the new directory.

import shutil

# Copy an entire directory and its contents to a new location

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

shutil.copy() is used for copying individual files.

shutil.copytree() is used for copying entire directory trees.

1. What function is used to rename files??

Ans - In Python, the os.rename() function is used to rename files. The os.rename() function takes two arguments: the current name of the file and the new name that you want to assign to it.

import os

# Specify the current and new names of the file

current\_name = 'old\_name.txt'

new\_name = 'new\_name.txt'

# Rename the file

os.rename(current\_name, new\_name)

In this example, the file named 'old\_name.txt' is renamed to 'new\_name.txt'. Keep in mind that both the current and new names should include the path if the file is not in the current working directory.

It's important to note that if a file with the new name already exists, the existing file will be overwritten without warning. Therefore, it's recommended to check for the existence of the destination file before performing the rename operation if you want to avoid accidental overwrites.

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

Ans - In Python, the send2trash and shutil modules provide functions for deleting files and directories, but they have some differences in terms of their behavior and purpose.

send2trash module:

The send2trash module is primarily used for moving files and directories to the operating system's trash or recycle bin, instead of permanently deleting them.

It sends files and directories to the system trash, allowing for easy recovery if needed.

This module provides a send2trash() function.

from send2trash import send2trash

# Move a file to the trash

send2trash('file\_to\_delete.txt')

This function sends the specified file or directory to the trash/recycle bin, making it a safer option when you want to provide a way to recover deleted items.

shutil module:

The shutil module provides the shutil.rmtree() function, which is used to recursively delete a directory and its contents. It doesn't move items to the trash but permanently deletes them.

It can be used to delete files using os.remove() or os.unlink() for individual files.

import shutil

# Permanently delete a file

os.remove('file\_to\_delete.txt')

# Permanently delete a directory and its contents

shutil.rmtree('directory\_to\_delete/')

when using shutil functions, especially shutil.rmtree(), as it permanently removes the specified files or directories without the option of recovery.

the primary difference lies in the purpose and behavior of the modules. send2trash is focused on sending items to the trash for potential recovery, while shutil provides functions for permanent deletion.

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

Ans - In Python, the ZipFile method equivalent to the File objects' open() method is ZipFile() itself. To open a zip file for reading or writing, you use the ZipFile() constructor.

example of how to open a zip file for reading:

import zipfile

# Open an existing zip file for reading

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

# Perform operations on the zip file

# ...

# The zip file is automatically closed when the 'with' block is exited

And for writing:

import zipfile

# Create a new zip file for writing

with zipfile.ZipFile('new\_archive.zip', 'w') as zip\_file:

# Add files to the zip file

zip\_file.write('file1.txt')

zip\_file.write('file2.txt')

# ...

# The zip file is automatically closed when the 'with' block is exited

In these examples, the ZipFile() constructor is used to open a zip file either for reading ('r' mode) or writing ('w' mode).

The with statement is used to ensure that the zip file is properly closed after the operations are performed.

1. 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 - use the os and shutil modules to achieve this task

import os

import shutil

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

# Create the destination folder if it doesn't exist

if not os.path.exists(destination\_folder):

os.makedirs(destination\_folder)

# Traverse the source folder tree

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

for file in files:

# Check if the file has the specified extension

if file.endswith(file\_extension):

# Construct source and destination paths

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

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

# Copy the file to the destination folder

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

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

# Example usage

source\_directory = '/path/to/source'

destination\_directory = '/path/to/destination'

file\_extension\_to\_copy = '.pdf' # Change this to the desired file extension

copy\_files\_with\_extension(source\_directory, destination\_directory, file\_extension\_to\_copy)

Replace /path/to/source with the path to the folder where you want to start searching, and /path/to/destination with the path to the folder where you want to copy the matching files. Adjust the file\_extension\_to\_copy variable to the desired file extension.

This script uses os.walk() to traverse the source folder and its subdirectories, and for each file with the specified extension, it copies the file to the destination folder using shutil.copy2().