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

Ans: `shutil.copy()` and `shutil.copytree()` are both functions provided by the `shutil` module in Python, but they serve different purposes:

i. `shutil.copy(src, dst)`:

- `shutil.copy()` is used to copy a single file from the source (`src`) to the destination (`dst`).

- It copies the file, creating a new file at the destination with the same content as the source file.

- If the destination file already exists, `shutil.copy()` will overwrite it.

Example using `shutil.copy()`:

import shutil

# Copy a single file from source to destination

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

ii. `shutil.copytree(src, dst)`:

- `shutil.copytree()` is used to recursively copy an entire directory tree from the source (`src`) to the destination (`dst`).

- It copies the entire directory structure along with all files and subdirectories under the source directory to the destination directory.

- If the destination directory already exists, `shutil.copytree()` will raise an error.

Example using `shutil.copytree()`:

import shutil

# Copy an entire directory tree from source to destination

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

1. What function is used to rename files??

Ans: To rename files in Python, you can use the `os.rename()` function from the built-in `os` module. This function allows you to change the name of a file by providing the current file path and the new file path (with the new name).

Here's the syntax for using `os.rename()`:

import os

# Rename the file 'old\_filename.txt' to 'new\_filename.txt'

os.rename('old\_filename.txt', 'new\_filename.txt')

In the above example, the file named `'old\_filename.txt'` will be renamed to `'new\_filename.txt'`. Note that both the old and new filenames should include the full or relative path to the file if it is not in the current working directory.

Keep in mind that the `os.rename()` function will raise an error if the file with the old name doesn't exist or if a file with the new name already exists. To handle these situations gracefully, you can use exception handling.

If you only want to change the name of the file and keep it in the same directory, you can use the `os.path` module to manipulate the filenames and paths. For example:

import os

old\_path = '/path/to/old\_filename.txt'

new\_path = os.path.join(os.path.dirname(old\_path), 'new\_filename.txt')

os.rename(old\_path, new\_path)

In this case, `os.path.dirname()` extracts the directory path from the original file's full path, and `os.path.join()` combines it with the new filename to create the new file path. This approach allows you to rename the file while keeping it in the same directory.

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

Ans: The `send2trash` and `shutil` modules in Python both provide functions to delete files or directories, but they have different behaviors and purposes:

A} `send2trash` module:

- The `send2trash` module provides a safe way to delete files or directories by moving them to the operating system's trash or recycle bin instead of permanently deleting them.

- The `send2trash.send2trash()` function is used to send files or directories to the trash, and it takes the path to the file or directory as its argument.

Example using `send2trash`:

import send2trash

# Send the file 'file.txt' to the trash instead of permanently deleting it

send2trash.send2trash('file.txt')

The `send2trash.send2trash()` function is useful when you want to provide a safety net and allow users to recover deleted files from the trash in case they were deleted accidentally.

B} `shutil` module:

- The `shutil` module provides functions for file operations, including the `shutil.rmtree()` function, which is used to delete directories and their contents recursively. It permanently deletes the specified directory and all its subdirectories and files.

Example using `shutil`:

import shutil

# Delete the directory 'my\_folder' and all its contents permanently

shutil.rmtree('my\_folder')

Unlike `send2trash`, the `shutil.rmtree()` function deletes the files and directories without moving them to the trash. This operation is not reversible, and the data will be lost permanently.

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 to File objects' `open()` method for ZipFile objects is `ZipFile()` from the `zipfile` module.

To work with ZIP files in Python, you can use the `zipfile` module, and the `ZipFile()` method is used to create a ZipFile object, similar to how the `open()` method is used to create a file object.

Here's an example of using `ZipFile()` to create a ZipFile object:

import zipfile

# Create a ZipFile object for writing (to create a new ZIP file)

with zipfile.ZipFile('example.zip', 'w') as myzip:

# Add files to the ZIP archive

myzip.write('file1.txt')

myzip.write('file2.txt')

# Later, in a different program run or session:

# Open an existing ZIP file in read mode

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

# Extract files from the ZIP archive or perform other operations

file\_list = myzip.namelist()

print(file\_list) # Output: ['file1.txt', 'file2.txt']

In the above example, the `ZipFile()` method is used with the mode `'w'` to create a new ZIP file called `'example.zip'`. Then, the `write()` method is used to add files to the ZIP archive. In the second part, the same ZIP file is opened in read mode (`'r'`), and we can access its contents using the `ZipFile` object.

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: To create a program that searches a folder tree for files with a specific file extension and then copies these files to a new folder, you can use the `os` and `shutil` modules in Python. The `os` module is used to traverse the folder tree and find files with the desired extension, while the `shutil` module is used to copy the files to the new folder. Here's a sample program to achieve this:

import os

import shutil

def find\_and\_copy\_files(src\_folder, dst\_folder, file\_extension):

# Ensure the source folder exists

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

print(f"Source folder '{src\_folder}' does not exist.")

return

# Ensure the destination folder exists, create if not

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

# Traverse the source folder tree and find files with the desired extension

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

for file in files:

if file.endswith(file\_extension):

# Get the full path of the source and destination files

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

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

# Copy the file to the destination folder

shutil.copy(src\_file\_path, dst\_file\_path)

print(f"Copied: {src\_file\_path} -> {dst\_file\_path}")

if \_\_name\_\_ == "\_\_main\_\_":

source\_folder = "/path/to/source\_folder" # Replace with the actual source folder path

destination\_folder = "/path/to/destination\_folder" # Replace with the actual destination folder path

extension\_to\_copy = ".pdf" # Specify the desired file extension

find\_and\_copy\_files(source\_folder, destination\_folder, extension\_to\_copy)

Replace `/path/to/source\_folder` with the path of the folder where you want to start the search, and `/path/to/destination\_folder` with the path of the folder where you want to copy the matching files. Set `extension\_to\_copy` to the desired file extension (e.g., `".pdf"` or `".jpg"`).When you run the program, it will traverse the folder tree starting from the source folder and copy all files with the specified extension to the destination folder. If the destination folder does not exist, the program will create it.