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

Ans: The `shutil.copy()` and `shutil.copytree()` functions in Python's `shutil` module are used for file and directory copying, respectively. They differ in the level of copying they perform and the types of inputs they accept.

Here's how you can distinguish between `shutil.copy()` and `shutil.copytree()`:

1. `shutil.copy(src, dst)`: This function is used to copy an individual file from the source (`src`) to the destination (`dst`). It creates a new file at the destination with the same contents as the source file.

Example usage:

```python

import shutil

# Copy a single file

shutil.copy('source/file.txt', 'destination/file.txt')

```

In this example, `shutil.copy()` is used to copy the file `source/file.txt` to the destination directory, creating a new file `destination/file.txt` with the same content as the source file.

2. `shutil.copytree(src, dst)`: This function is used to copy an entire directory tree from the source directory (`src`) to the destination directory (`dst`). It recursively copies all files and directories under the source directory, preserving the directory structure.

Example usage:

```python

import shutil

# Copy a directory and its contents

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

```

In this example, `shutil.copytree()` is used to copy the entire directory tree rooted at `source\_dir` to `destination\_dir`. It creates a new directory at the destination and recursively copies all files and directories from the source directory.

To summarize, `shutil.copy()` is used for copying individual files, while `shutil.copytree()` is used for recursively copying entire directory trees, including all files and subdirectories.

1. What function is used to rename files??

Ans: The `shutil.move()` function is commonly used to rename files in Python. While its primary purpose is to move files or directories, it can also be utilized for renaming files by specifying a new name or path for the file.

Here's an example of using `shutil.move()` to rename a file:

```python

import shutil

# Rename a file

shutil.move('old\_name.txt', 'new\_name.txt')

```

In this example, the file 'old\_name.txt' is renamed to 'new\_name.txt' by specifying the new name as the destination in the `shutil.move()` function.

It's worth noting that the `shutil.move()` function can also be used to move files to a different directory. If the destination path provided in `shutil.move()` differs from the source path in terms of the file name, it effectively renames the file.

Make sure to exercise caution when using `shutil.move()` to rename files, as it will overwrite any existing file with the same name in the destination location.

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

Ans: The `send2trash` and `shutil` modules in Python provide different approaches for deleting files and directories, and they have distinct differences in functionality and behavior.

1. `send2trash` module:

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

- The `send2trash.send2trash()` function is used to send files or directories to the trash.

- This approach allows for potential recovery of deleted items from the trash or recycle bin, providing a level of safety when deleting files or directories.

- Example:

```python

from send2trash import send2trash

# Send a file to the trash

send2trash('path/to/file.txt')

```

2. `shutil` module:

- The `shutil` module provides a variety of file operations, including the ability to delete files and directories permanently.

- The `shutil.rmtree()` function is commonly used to remove a directory and all its contents recursively.

- The `os.remove()` function from the `os` module, which is also commonly used, can be used to delete individual files.

- Unlike `send2trash`, the deletion performed by `shutil.rmtree()` and `os.remove()` is permanent and bypasses the operating system's trash or recycle bin.

- Example:

```python

import shutil

# Permanently delete a directory and its contents

shutil.rmtree('path/to/directory')

# Permanently delete a file

os.remove('path/to/file.txt')

```

To summarize, the `send2trash` module allows files and directories to be sent to the trash or recycle bin, providing a way to potentially recover them, while the `shutil` module provides functions for permanently deleting files and directories without moving them to the trash or recycle bin. The choice between these modules depends on the desired behavior and the level of safety required for file and directory 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: The equivalent method in the `ZipFile` class to the `open()` method of file objects is the `ZipFile()` constructor method.

The `ZipFile()` constructor is used to create a new `ZipFile` object, which represents a ZIP archive file. It allows you to open and work with ZIP files, extract their contents, and perform various operations on the archive.

Here's an example of using the `ZipFile()` constructor to open a ZIP archive:

```python

from zipfile import ZipFile

# Open a ZIP archive file

zip\_file = ZipFile('archive.zip', 'r')

```

In this example, `ZipFile('archive.zip', 'r')` creates a new `ZipFile` object by opening the ZIP archive file named 'archive.zip' in read mode (`'r'`). This allows you to access and manipulate the contents of the ZIP file.

Once you have finished working with the `ZipFile` object, you should call its `close()` method to release any resources associated with the ZIP file and close the file.

```python

# Close the ZIP file

zip\_file.close()

```

The `close()` method is equivalent to the `close()` method of file objects, as it releases the resources held by the `ZipFile` object, such as the underlying file handle, and closes the ZIP archive file.

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: Certainly! Here's an example program that searches a folder tree for files with a specified file extension and copies them to a new folder using the `shutil` module in Python:

```python

import os

import shutil

def search\_and\_copy\_files(source\_folder, target\_folder, file\_extension):

# Create the target folder if it doesn't exist

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

# Traverse the folder tree and copy matching files

for foldername, \_, filenames in os.walk(source\_folder):

for filename in filenames:

if filename.endswith(file\_extension):

source\_file = os.path.join(foldername, filename)

target\_file = os.path.join(target\_folder, filename)

shutil.copy(source\_file, target\_file)

print(f"Copied: {source\_file} --> {target\_file}")

# Example usage

source\_folder = 'path/to/source\_folder'

target\_folder = 'path/to/target\_folder'

file\_extension = '.pdf'

search\_and\_copy\_files(source\_folder, target\_folder, file\_extension)

```

In this example, the `search\_and\_copy\_files()` function takes three parameters: `source\_folder` (the root folder to search for files), `target\_folder` (the destination folder to copy the files), and `file\_extension` (the file extension to search for, e.g., '.pdf', '.jpg', etc.).

The function recursively traverses the folder tree using `os.walk()` and checks if each file's extension matches the specified `file\_extension`. If a match is found, the file is copied from its source location to the target folder using `shutil.copy()`. The program prints the source and target file paths to provide feedback on the copying process.

You can modify the `source\_folder`, `target\_folder`, and `file\_extension` variables according to your specific requirements.