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

Ans - In Python's `shutil` module, both `shutil.copy()` and `shutil.copytree()` are used for file and directory operations, but they have different purposes and usage patterns. Here's how you can distinguish between them:

1. shutil.copy(src, dst):

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

- It requires that the `src` path points to an existing file, and the `dst` path specifies the target file name or a directory where the file will be copied.

- If the `dst` path specifies an existing file, it will be overwritten.

- Example usage:

```python

import shutil

# Copy a file from source to destination

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

```

2. shutil.copytree(src, dst):

- This function is used to recursively copy an entire directory from the source path (`src`) to the destination path (`dst`).

- It requires that the `src` path points to an existing directory, and the `dst` path specifies the target directory where the source directory will be copied.

- If the `dst` path already exists, it will raise a `FileExistsError`.

- Example usage:

```python

import shutil

# Copy a directory from source to destination

shutil.copytree('source/dir', 'destination/dir')

```

In summary, `shutil.copy()` is used for copying individual files, while `shutil.copytree()` is used for recursively copying entire directories.

1. **What function is used to rename files?**

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

The `os.rename()` function takes two arguments: the current name or path of the file, and the new name or path that the file should be renamed to. Here's the general syntax:

```python

import os

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

```

It's important to note that `os.rename()` can be used to rename both files and directories. However, if you are renaming a directory, it should be empty, otherwise, an error will occur.

Here's an example that demonstrates how to use `os.rename()` to rename a file:

```python

import os

current\_name = 'old\_name.txt'

new\_name = 'new\_name.txt'

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

```

In this example, the file named `'old\_name.txt'` will be renamed to `'new\_name.txt'` in the same directory.

Keep in mind that `os.rename()` will raise an exception if the file doesn't exist or if you don't have permission to rename it. Therefore, it's a good practice to handle potential exceptions using a `try-except` block when using this function.

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. Here's the difference between the delete functions in these two modules:

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 permanently deleting them.

- It is useful when you want to provide a way for users to recover deleted files or if you want to avoid accidental deletions.

- The primary function in this module is `send2trash.send2trash()`, which takes a file or directory path as an argument and sends it to the trash or recycle bin.

- Example usage:

```python

import send2trash

# Delete a file or directory by sending it to the trash

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

```

2. `shutil` module:

- The `shutil` module provides various functions for file and directory operations, including deleting files and directories.

- The deletion functions in `shutil` permanently remove files and directories, bypassing the trash or recycle bin. They cannot be recovered using typical methods like undo or restore.

- The `shutil` module provides multiple deletion functions such as `shutil.rmtree()` for deleting directories and `os.remove()` for deleting files.

- Example usage:

```python

import shutil

import os

# Delete a file using shutil

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

# Delete a directory using shutil

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

```

In summary, the `send2trash` module moves files and directories to the trash or recycle bin, allowing for potential recovery, while the deletion functions in the `shutil` module permanently delete files and directories without moving them to the trash. The choice between them depends on whether you want to provide a safety net for recoverability or if you need to ensure 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 - The equivalent method in the `ZipFile` class to the `open()` method of file objects is the `ZipFile()` constructor method.

When working with file objects, the `open()` method is used to open a file and create a file object that allows various operations such as reading, writing, or appending to the file. Similarly, in the `ZipFile` class, the `ZipFile()` constructor method is used to open a ZIP file and create a `ZipFile` object that provides methods to work with the contents of the ZIP file.

Here's an example that demonstrates the equivalent usage:

```python

import zipfile

# Open a file using open()

with open('file.txt', 'r') as file\_obj:

# Perform operations on the file object

# ...

# Open a ZIP file using ZipFile()

with zipfile.ZipFile('archive.zip', 'r') as zip\_obj:

# Perform operations on the ZipFile object

# ...

```

In the above example, the `open()` method is used to open the file `'file.txt'` in read mode, creating a file object (`file\_obj`). Similarly, the `ZipFile()` constructor method is used to open the ZIP file `'archive.zip'` in read mode, creating a `ZipFile` object (`zip\_obj`).

Once the file or ZIP file is opened, you can use the respective object (`file\_obj` or `zip\_obj`) to perform operations like reading, writing, or extracting data, depending on the purpose.

**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 a Python program that searches a folder tree for files with a specific file extension and copies them to a new folder:

```python

import os

import shutil

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

# Create the destination folder if it doesn't exist

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

# Traverse the folder tree

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

for file in files:

# Check if the file has the desired extension

if file.endswith(file\_extension):

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} -> {destination\_path}")

# Example usage

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

destination\_folder = '/path/to/destination/folder'

file\_extension = '.pdf'

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

```

In this example, the `search\_and\_copy\_files()` function takes three parameters: the `source\_folder` (the root folder to start the search from), `destination\_folder` (the folder where the matching files will be copied), and `file\_extension` (the desired file extension to search for).

The program uses the `os.walk()` function to traverse the folder tree starting from the `source\_folder`. For each file encountered, it checks if the file extension matches the desired `file\_extension`. If so, it constructs the source and destination paths and uses `shutil.copy2()` to copy the file to the `destination\_folder`. The `shutil.copy2()` function preserves the original file metadata such as timestamps.

Make sure to replace `'/path/to/source/folder'` with the actual path of the folder you want to search, and `'/path/to/destination/folder'` with the actual path of the folder where you want to copy the files. Also, modify `file\_extension` to the desired file extension you want to search for (e.g., `'.pdf'`, `'.jpg'`, etc.).

Note: The program assumes that the destination folder already exists or will be created if it doesn't exist. The `os.makedirs()` function is used to create the destination folder if it doesn't exist (`exist\_ok=True` ensures that no error is raised if the folder already exists).