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

In Python’s shutil module, shutil.copy() and shutil.copytree() are used for copying files and directories respectively. Here’s how they differ:

* shutil.copy(src, dst, \*, follow\_symlinks=True): This function copies the file src to the file or directory dst. src and dst are path names given as strings. If dst specifies a directory, the file will be copied into dst using the base filename from src. Returns the path to the newly created file.
* shutil.copytree(src, dst, symlinks=False, ignore=None, copy\_function=copy2, ignore\_dangling\_symlinks=False): This function recursively copies an entire directory tree rooted at src to a directory named dst and returns the destination directory. It will raise an Error if the destination directory already exists. To ignore certain files and directories, ignore can be used. It must be a callable that takes a directory name and filenames as input parameters and returns a list of names to ignore.

In summary, use shutil.copy() when you want to copy a single file, and shutil.copytree() when you want to copy an entire directory tree. Please note that shutil.copytree() will throw an error if the destination directory already exists, unlike shutil.copy().

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

In Python, the os.rename() function is used to rename files. This function is part of the os module. Here’s a basic example:

import os

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

[In this example, ‘old\_filename.txt’ is the current name of the file, and ‘new\_filename.txt’ is the new name you want to give to the file](https://stackoverflow.com/questions/2491222/how-to-rename-a-file-using-python)[1](https://stackoverflow.com/questions/2491222/how-to-rename-a-file-using-python)[2](https://pythongeeks.org/rename-files-in-python/)[3](https://datagy.io/python-rename-file/)[4](https://blog.enterprisedna.co/how-to-rename-a-file-in-python-4-easy-ways/)[5](https://www.geeksforgeeks.org/python-os-rename-method/).

Please note that you need to provide the full path of the file if the file is not in the same directory as your Python script. For example:

import os

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

[Also, be aware that os.rename() will raise a FileExistsError if the destination file already exists3](https://datagy.io/python-rename-file/). [If you want to ensure that the destination file is replaced even if it exists, you can use os.replace()](https://stackoverflow.com/questions/2491222/how-to-rename-a-file-using-python)[1](https://stackoverflow.com/questions/2491222/how-to-rename-a-file-using-python).

Remember to handle exceptions appropriately in your code to avoid any runtime errors.

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

The send2trash and shutil modules in Python both provide functions to delete files and folders, but they do so in different ways:

* send2trash: This module provides a safer way to delete files and folders. [When you use send2trash, the files and folders are moved to the recycle bin or trash rather than being permanently deleted1](https://www.geeksforgeeks.org/how-to-delete-files-in-python-using-send2trash-module/). This means you can recover them from the recycle bin if you deleted them accidentally. Here’s an example of how to use it:
* import send2trash
* send2trash.send2trash("/path/to/file")
* shutil: This module provides functions to perform high-level operations on files and collections of files. It includes functions to copy, move, rename, and delete files and directories. [When you use shutil to delete files or directories, they are permanently deleted](https://www.geeksforgeeks.org/how-to-delete-files-in-python-using-send2trash-module/)[2](http://automatetheboringstuff.com/chapter9/). Here’s an example of how to use it:
* import shutil
* shutil.rmtree("/path/to/directory")

In summary, if you want to delete files or directories permanently and immediately, you can use shutil. If you want to move them to the recycle bin (providing an option for recovery), you can use send2trash. [Please remember to handle exceptions appropriately in your code to avoid any runtime errors1](https://www.geeksforgeeks.org/how-to-delete-files-in-python-using-send2trash-module/)[2](http://automatetheboringstuff.com/chapter9/)[3](https://www.realcode4you.com/post/python-interview-questions-and-practice-set-5-hire-python-expert-to-get-help)[4](https://github.com/smselby/Answers-to-Chapter-9).

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

The `ZipFile` method equivalent to the `open()` method for file objects is `ZipFile()` itself. You use the `ZipFile()` constructor to create a new `ZipFile` object, analogous to how you use the `open()` function to open a file and obtain a file object.

Here's an example:

```python

from zipfile import ZipFile

# Equivalent to opening a zip file

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

# Perform operations on the ZipFile object

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

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

# The 'with' statement automatically closes the ZipFile object when done

```

In this example, `ZipFile('example.zip', 'w')` is equivalent to opening a zip file named 'example.zip' in write mode (`'w'`). The `with` statement is used to ensure that the `ZipFile` object is automatically closed when the block is exited.

Just like the `open()` function for file objects, the `ZipFile()` constructor takes the file name and the mode as arguments. The modes for a `ZipFile` include:

- `'r'`: Read (default)

- `'w'`: Write (creates a new zip file or overwrites an existing one)

- `'a'`: Append (adds files to an existing zip file)

So, `ZipFile()` is the equivalent of `open()` when working with zip files.

**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.**

Sure, here’s a Python script that does that. This script uses the os and shutil modules to walk through a directory tree, find files with a certain extension, and copy them to a new directory.

import os

import shutil

def copy\_files\_with\_extension(src\_folder, dst\_folder, extension):

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

os.makedirs(dst\_folder)

for foldername, subfolders, filenames in os.walk(src\_folder):

for filename in filenames:

if filename.endswith(extension):

shutil.copy(os.path.join(foldername, filename), dst\_folder)

# Usage

copy\_files\_with\_extension('/path/to/source/folder', '/path/to/destination/folder', '.pdf')

In this script, copy\_files\_with\_extension is a function that takes three arguments: src\_folder (the root of the directory tree to search), dst\_folder (the destination folder where the files will be copied to), and extension (the file extension to search for). The function first checks if the destination folder exists, and creates it if it doesn’t. Then it walks through the directory tree rooted at src\_folder. For each file in the directory tree, if the file’s extension matches extension, the file is copied to dst\_folder.

Please replace '/path/to/source/folder', '/path/to/destination/folder', and '.pdf' with your actual source folder, destination folder, and file extension respectively.

Remember to handle exceptions appropriately in your code to avoid any runtime errors.