Name :- Deepawali . B. Mhaisagar
Assignment no 10

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

The shutil module in Python provides functions for file and directory operations. To distinguish between shutil.copy() and shutil.copytree(), consider the following:

shutil.copy(src, dst): This function is used to copy a single file from the source (src) to the destination (dst) directory. It creates a new file with the same contents as the source file.
Example usage:

```
import shutil

source_file = '/path/to/source/file.txt'

destination_dir = '/path/to/destination/'

shutil.copy(source_file, destination_dir)
```

In this case, file.txt from source_file is copied to destination_dir. The destination can be a directory, and the copied file will retain its original name in the destination directory.
shutil.copytree(src, dst): This function is used to recursively copy an entire directory tree from the source (src) to the destination (dst) directory. It copies all files and subdirectories within the source directory.
Example usage:

```
import shutil

source_dir = '/path/to/source/directory/'

destination_dir = '/path/to/destination/'

shutil.copytree(source_dir, destination_dir)
```

In this case, the entire directory tree rooted at source_dir is copied to destination_dir. The destination directory should not exist before calling shutil.copytree(), as it will be created and populated with the source directory's contents.

To summarize:

- shutil.copy(src, dst): Copies a single file from the source to the destination directory.

- shutil.copytree(src, dst): Copies an entire directory tree, including all files and subdirectories, from the source to the destination directory.

It's important to note that both functions overwrite existing files with the same names in the destination directory. If you want to preserve the original file attributes, such as permissions or timestamps, you can use the shutil.copy2() or shutil.copytree() functions instead.

2. What function is used to rename files??

ANS

In Python, the os.rename() function is used to rename files or directories. It allows you to change the name of an existing file or directory to a new name.

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

```
import os

source = 'old_filename.txt'

destination = 'new_filename.txt'

os.rename(source, destination)
```

In the example above, the file with the name 'old_filename.txt' will be renamed to 'new_filename.txt'.

3. 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 to deleting files or directories. Here's the difference between the delete functions in each module:

send2trash module:

- The send2trash module provides a safer alternative for deleting 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 delete files or directories and sends them to the trash.
- This module is useful when you want to have a chance to recover the deleted files from the trash before they are permanently removed.
- Example usage:
- python

- Copy code

```
import send2trash

file_path = 'path/to/file.txt'

send2trash.send2trash(file_path)
```

- In this case, the file at file_path will be moved to the trash or recycle bin, depending on the operating system.

shutil module:

- The shutil module provides the shutil.rmtree() function, which is used to remove an entire directory tree, including all files and subdirectories.
- The shutil.rmtree() function permanently deletes the specified directory and its contents without the option for recovery from the trash.
- This module is useful when you want to remove directories and their contents entirely without the need for a trash or recycle bin.
- Example usage:
- python
- Copy code

```
import shutil

directory_path = 'path/to/directory'

shutil.rmtree(directory_path)
```

- In this case, the directory at directory_path and all its contents will be permanently deleted.

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 ZipFile objects to the open() method in File objects is the ZipFile.open() method.

The ZipFile.open() method is used to open a file contained within a ZIP archive. It allows you to access the contents of a specific file within the archive, similar to how the open() method is used to open a file in the file system.

Here's an example of using ZipFile.open():

```
import zipfile

archive_path = 'archive.zip'
```

```
# Open the ZIP archive

with zipfile.ZipFile(archive_path, 'r') as archive:

    # Open a file within the archive

    with archive.open('file.txt') as file:

        # Perform operations on the opened file

        content = file.read()

        print(content)
```

In the example above, the ZipFile object is created using zipfile.ZipFile(), specifying the path to the ZIP archive and the mode ('r' for reading). Then, using the archive.open() method, a specific file named 'file.txt' within the archive is opened. Operations can be performed on the opened file, such as reading its contents using file.read().

The ZipFile.open() method provides access to files within a ZIP archive, allowing you to interact with the contents of the archive as if they were regular files in the file system.

Remember to close the ZipFile object after you finish working with the archive, either by using the close() method explicitly or by using a with statement, as shown in the example.

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

```
import os

import shutil

def search_and_copy_files(source_folder, target_folder, file_extension):

    for root, dirs, files in os.walk(source_folder):

        for file in files:

            if file.endswith(file_extension):
```

```
            source_path = os.path.join(root, file)

            target_path = os.path.join(target_folder, file)

            shutil.copy(source_path, target_path)

            print(f"File '{file}' copied to '{target_folder}'")



# 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)
```

Here's how the program works:

- The search_and_copy_files() function takes three parameters: source_folder (the root folder to search in), target_folder (the destination folder to copy files to), and file_extension (the desired file extension to search for).
- The os.walk() function is used to iterate through all the directories and files within the source_folder. It returns a generator that yields the current directory path, a list of subdirectories, and a list of filenames in the current directory.
- For each file encountered, the program checks if it has the specified file_extension using the file.endswith() method.
- If a file matches the desired extension, the source path and target path are constructed using os.path.join() by combining the root path with the file name.
- The shutil.copy() function is used to copy the file from the source path to the target path.
- Finally, a message is printed to indicate that the file has been copied to the target folder.

You can customize the source_folder, target_folder, and file_extension variables to match your specific requirements. Ensure that the target folder exists before running the program.