```
import os
import subprocess

def run_exiftool(file_path):
    print("\n[+] Running ExifTool:")
    try:
        result = subprocess.run(["exiftool", file_path], capture_output=True,
text=True)
        print(result.stdout)
    except Exception as e:
        print(f"[-] Error running exiftool: {e}")

def run_strings(file_path):
    print("\n[+] Extracting strings:")
    try:
        result = subprocess.run(["strings", file_path], capture_output=True,
text=True)
        print(result.stdout)
    except Exception as e:
        print(f"[-] Error running strings: {e}")

def run_binwalk(file_path):
    print("\n[+] Running binwalk -e to extract embedded files:")
    try:
        subprocess.run(["binwalk", "-e", file_path])
    except Exception as e:
        print(f"[-] Error running binwalk: {e}")

def run_foremost(file_path):
    print("\n[+] Running foremost to recover hidden files:")
    try:
        output_dir = "output"
        subprocess.run(["foremost", "-i", file_path, "-T", "-o", output_dir])
        print(f"[+] Foremost results saved to ./{output_dir}/")
    except Exception as e:
        print(f"[-] Error running foremost: {e}")

def brute_force_steghide(file_path, wordlist_path):
    print("\n[+] Brute-forcing steghide password (this may take time)...")
    try:
        with open(wordlist_path, "r", encoding="latin-1") as f:
            passwords = f.read().splitlines()

        for pw in passwords:
            result = subprocess.run(
                ["steghide", "extract", "-sf", file_path, "-p", pw, "-xf",
"extracted_secret.txt"],
                capture_output=True, text=True
            )
            if "wrote extracted data to" in result.stdout.lower():
                print(f"[+] Password found: {pw}")
                print("[+] Extracted file: extracted_secret.txt")
                return
        print("[-] Password not found in wordlist.")
    except Exception as e:
```

```
        print(f"[-] Error in brute-force: {e}")

def main():
    file_path = input("Enter path to stego file (e.g., /home/user/stego.jpg):
").strip()
    if not os.path.isfile(file_path):
        print("[-] File not found.")
        return

    run_exiftool(file_path)
    run_strings(file_path)
    run_binwalk(file_path)
    run_foremost(file_path)

    choice = input("\nDo you want to brute-force steghide password? (y/n):
").strip().lower()
    if choice == 'y':
        wordlist_path = input("Enter path to wordlist (e.g.,
/usr/share/wordlists/rockyou.txt): ").strip()
        if os.path.isfile(wordlist_path):
            brute_force_steghide(file_path, wordlist_path)
        else:
            print("[-] Wordlist file not found.")

if __name__ == "__main__":
    main()
```

---

python
```
import os
import subprocess
```

Explanation: These are standard Python modules:

- `os`: Provides functions to interact with the operating system (e.g., checking if a file exists).
- `subprocess`: Allows execution of shell commands from within Python.

python
```
def run_exiftool(file_path):
    print("\n[+] Running ExifTool:")
```

Explanation: Defines the function `run_exiftool()` to extract metadata from a file using `ExifTool`. It takes `file_path` as an argument and prints a message indicating that the tool is being executed.

python
```
    try:
        result = subprocess.run(["exiftool", file_path], capture_output=True,
text=True)
        print(result.stdout)
```

Explanation: Uses `subprocess.run()` to execute `exiftool` with the provided file path. The `capture_output=True` ensures output is captured, and `text=True` converts it into a readable string. It then prints the metadata extracted from the file.

python
```
    except Exception as e:
        print(f"[-] Error running exiftool: {e}")
```

Explanation: Catches any errors that occur while running `exiftool` and prints an error message.

python
```
def run_strings(file_path):
    print("\n[+] Extracting strings:")
```

Explanation: Defines the `run_strings()` function, which extracts readable strings from the given file.

python
```
    try:
        result = subprocess.run(["strings", file_path], capture_output=True,
text=True)
        print(result.stdout)
```

Explanation: Runs the `strings` command on the file, capturing readable text that may indicate hidden data or embedded messages.

python
```
    except Exception as e:
        print(f"[-] Error running strings: {e}")
```

Explanation: Handles errors that may occur while executing the `strings` command.

python
```
def run_binwalk(file_path):
    print("\n[+] Running binwalk -e to extract embedded files:")
```

Explanation: Defines `run_binwalk()` to search for embedded files within the provided file using `binwalk`.

python
```
    try:
        subprocess.run(["binwalk", "-e", file_path])
```

Explanation: Executes `binwalk` with the `-e` flag, which attempts to extract hidden embedded files.

python
```
    except Exception as e:
```

```
        print(f"[-] Error running binwalk: {e}")
```

Explanation: Handles errors that may occur when running `binwalk`.

python
```
def run_foremost(file_path):
    print("\n[+] Running foremost to recover hidden files:")
```

Explanation: Defines `run_foremost()`, which recovers hidden or deleted files using the `foremost` tool.

python
```
    try:
        output_dir = "output"
        subprocess.run(["foremost", "-i", file_path, "-T", "-o", output_dir])
        print(f"[+] Foremost results saved to ./{output_dir}/")
```

Explanation: Runs `foremost` on the file, saving recovered files to the "output" directory.

python
```
    except Exception as e:
        print(f"[-] Error running foremost: {e}")
```

Explanation: Handles errors that occur when executing `foremost`.

python
```
def brute_force_steghide(file_path, wordlist_path):
    print("\n[+] Brute-forcing steghide password (this may take time)...")
```

Explanation: Defines `brute_force_steghide()` to attempt a brute-force attack on a Steghide-protected file using a wordlist.

python
```
    try:
        with open(wordlist_path, "r", encoding="latin-1") as f:
            passwords = f.read().splitlines()
```

Explanation: Opens the wordlist file, reading its contents line by line, storing passwords in a list.

python
```
        for pw in passwords:
            result = subprocess.run(
                ["steghide", "extract", "-sf", file_path, "-p", pw, "-xf",
"extracted_secret.txt"],
                capture_output=True, text=True
            )
```

Explanation: Iterates through the passwords, trying each one with `steghide extract`. If successful, it extracts a hidden file (`extracted_secret.txt`).

python
```
            if "wrote extracted data to" in result.stdout.lower():
                print(f"[+] Password found: {pw}")
                print("[+] Extracted file: extracted_secret.txt")
                return
```

Explanation: Checks if the output indicates successful extraction. If a correct password is found, it prints the password and exits the function.

python
```
        print("[-] Password not found in wordlist.")
```

Explanation: If no valid password is found, prints a failure message.

python
```
    except Exception as e:
        print(f"[-] Error in brute-force: {e}")
```

Explanation: Catches errors related to file access or command execution.

python
```
def main():
    file_path = input("Enter path to stego file (e.g., /home/user/stego.jpg):
").strip()
```

Explanation: Defines `main()`, prompts the user for a file path, and removes extra whitespace.

python
```
    if not os.path.isfile(file_path):
        print("[-] File not found.")
        return
```

Explanation: Checks if the provided file exists. If not, prints an error message and exits.

python
```
    run_exiftool(file_path)
    run_strings(file_path)
    run_binwalk(file_path)
    run_foremost(file_path)
```

Explanation: Calls all previously defined analysis functions.

python
```
    choice = input("\nDo you want to brute-force steghide password? (y/n):
").strip().lower()
```

Explanation: Asks the user if they want to perform brute-force password cracking.

python
```
    if choice == 'y':
```

```
        wordlist_path = input("Enter path to wordlist (e.g.,
/usr/share/wordlists/rockyou.txt): ").strip()
        if os.path.isfile(wordlist_path):
            brute_force_steghide(file_path, wordlist_path)
        else:
            print("[-] Wordlist file not found.")
```

Explanation: If the user selects brute-force, asks for a wordlist file. If the file exists, runs `brute_force_steghide()`.

python

```
if __name__ == "__main__":
    main()
```

Explanation: Ensures the script runs only when executed directly, not when imported as a module.