Creating a fully functional automated Tether wallet using the Tether Python library involves several steps. Below is a simplified example to help you get started. Note that handling private keys and interacting with the blockchain requires a deep understanding of security and blockchain concepts, and you should take appropriate measures to secure your private keys.

1. **Install the `tether-python` library**:

   ```
   bash
   ```

   - `pip install tether-python`

- **Create a Configuration File**: Create a configuration file (e.g., `config.ini`) to store your public and private keys. Here's a simple example:

```
ini
```

  - `[Wallet]`

```
public_key = YOUR_PUBLIC_KEY
private_key = YOUR_PRIVATE_KEY
```

- **Create Python Script**: Now, you can create a Python script (e.g., `automated_wallet.py`) to automate transactions:

```
python
```

3. continues on next page

```
from tether import Tether
from configparser import ConfigParser

class AutomatedWallet:
    def __init__(self, config_file):
        self.config = ConfigParser()
        self.config.read(config_file)
        self.tether = Tether()

    def get_balance(self):
        return self.tether.get_balance(self.config['Wallet']['public_key'])

    def send_transaction(self, to_address, amount):
        before_balance = self.get_balance()
        transaction_result = self.tether.send_transaction(
            self.config['Wallet']['private_key'], to_address, amount
        )
        after_balance = self.get_balance()

        print(f"Before Transaction Balance: {before_balance}")
        print(f"After Transaction Balance: {after_balance}")

        return transaction_result

if __name__ == "__main__":
    config_file = "config.ini"
    wallet = AutomatedWallet(config_file)

    # Example: Send 10 USDT to another address
    to_address = "RECIPIENT_ADDRESS"
    amount = 10

    transaction_result = wallet.send_transaction(to_address, amount)
    print(f"Transaction Result: {transaction_result}")
```

4. **Usage**:
    - Replace `YOUR_PUBLIC_KEY` and `YOUR_PRIVATE_KEY` in the configuration file with your actual Tether wallet public and private keys.
    - Set the `to_address` variable to the recipient's address.
    - Run the script.

This script provides a basic structure for interacting with the Tether blockchain using the `tether-python` library. Note that you need to handle security aspects carefully, especially when dealing with private keys.

Remember that using automation scripts to handle financial transactions involves risks, and you should thoroughly test and understand the code before using it with real assets. Always follow best practices for securely managing private keys and sensitive information.