Your SolanaWallet class looks good. It is well-organized and easy to read. The code is also efficient and uses the Solana Python library correctly.

Here are a few suggestions:

* In the `send_transaction()` method, you could add a `raise` statement to handle any errors that occur during the transaction process. For example:

```python
def send_transaction(self, to_address, amount):
    # Create a transaction
    transaction = Transaction(recent_blockhash=self.client.get_recent_blockhash())
    transaction.add(TransferParams(from_pubkey=self.account.public_key(),
                                   to_pubkey=to_address,
                                   lamports=amount))

    # Sign the transaction
    transaction.sign(self.account)

    # Send the transaction
    try:
        result = self.client.send_transaction(transaction)
    except Exception as e:
        raise e

    # Display before and after transaction balances
    self.display_balance("Before transaction balance")
    self.display_balance("After transaction balance")

    return result
```

* In the `display_balance()` method, you could convert the lamport balance to SOL before displaying it. This would make it easier for users to read and understand. For example:

```python
def display_balance(self, message):
    # Get balance of the wallet
    balance = self.get_balance()

    # Convert lamports to SOL
    balance_in_sol = balance / LAMPORTS_PER_SOL

    # Display balance with a custom message
    print(f"{message}: {balance_in_sol} SOL")
```

Overall, your SolanaWallet class is well-written and easy to use. Keep up the good work!