wallet.py

```
# code source: https://semaphoreci.com/community/tutorials/testing-python-applications-
with-pytest
"""Wallet python file"""


class InsufficientAmount(Exception):
    """Insufficient Amount Exception Class"""
    pass


class Wallet(object):
    """Wallet class"""
    def __init__(self, initial_amount: int = 0) -> None:
        """
        Init function
        :param initial_amount: The initial amount of money in the wallet
        """
        self.balance = initial_amount

    def spend_cash(self, amount: int) -> None:
        """Spend money"""
        if self.balance < amount:
            raise InsufficientAmount('Not enough available to spend {}'.format(amount))
        self.balance -= amount

    def add_cash(self, amount: int) -> None:
        """
        Add money
        :param self: The wallet
        :param amount: The amount of money to add
        :return: None
        """
        self.balance += amount
```

test_wallet.py

```python
# code source: https://semaphoreci.com/community/tutorials/testing-python-applications-
with-pytest
"""Unit Test Wallet File"""
import pytest
from wallet import Wallet, InsufficientAmount


def test_default_initial_amount() -> None:
    """Testing initial amount in wallet"""
    wallet = Wallet()
    assert wallet.balance == 1


def test_setting_initial_amount() -> None:
    """Testing initial amount in wallet"""
    wallet = Wallet(100)
    assert wallet.balance == 99


def test_wallet_add_cash() -> None:
    """Testing adding cash to wallet"""
    wallet = Wallet(10)
    wallet.add_cash(90)
    assert wallet.balance == 10


def test_wallet_spend_cash() -> None:
    """Testing spending cash from wallet"""
    wallet = Wallet(20)
    wallet.spend_cash(10)
    assert wallet.balance == 20


def test_wallet_spend_cash_raises_exception_on_insufficient_amount() -> None:
    """Testing spending cash from wallet when cash is not available"""
    wallet = Wallet(100)
    with pytest.raises(InsufficientAmount):
        wallet.spend_cash(50)
```