Let's recapitulate the full code of the proxy.py file:

- First, given a class called SensitiveInfo, which holds sensitive information, below:

```
class SensitiveInfo:
    def __init__(self):
        self.users = ['nick', 'tom', 'ben', 'mike']

    def read(self):
        nb = len(self.users)
        print(f"There are {nb} users: {' '.join(self.users)}")

    def add(self, user):
        self.users.append(user)
        print(f'Added user {user}')
```

- We define another class (called Info), to provide a protection proxy of the SensitiveInfo class, as shown here:

```
class Info:
    '''protection proxy to SensitiveInfo'''

    def __init__(self):
        self.protected = SensitiveInfo()
        self.secret = '0xdeadbeef'

    def read(self):
        self.protected.read()

    def add(self, user):
        sec = input('what is the secret? ')
        self.protected.add(user) if sec == self.secret else print("Tha
t's wrong!")
```

- Finally, here is the main() function where we exploit the Info class, and the end of the code where main() is called:

```
def main():
    info = Info()

    while True:
        print('1. read list |==| 2. add user |==| 3. quit')
        key = input('choose option: ')
        if key == '1':
            info.read()
        elif key == '2':
            name = input('choose username: ')
            info.add(name)
        elif key == '3':
            exit()
        else:
            print(f'unknown option: {key}')

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

We can see a sample output of the program when executing the command: python proxy.py