سوال اول:

.۱

1. با استفاده از `@dataclass`، کلاسی به نام `User` بسازید که شامل فیلدهای زیر باشد:
   - `name: str`
   - `email: str`
   - `password_hash: str`
   - `created_at: datetime` (به صورت خودکار مقداردهی شود)
2. رمز عبور هنگام ایجاد کاربر نباید به‌صورت خام ذخیره شود؛ باید قبل از ذخیره، هش شود.

.۲

کلاسی به نام `UserManager` بسازید که وظیفه‌ی افزودن، حذف، جستجو و نمایش کاربران را دارد.

این کلاس باید:

- کاربران را در یک فایل با فرمت `pickle` ذخیره و بازیابی کند.
- از ماژول `pprint` برای نمایش خوانای داده‌ها استفاده کند.
- هنگام بارگذاری، بررسی کند اگر فایل کاربران وجود نداشت، خودش ایجادش کند.

.۳

یک **decorator** به نام `require_password_check` بنویسید که:

- پیش از اجرای هر متد حساس (مثل حذف کاربر)، از کاربر بخواهد رمز عبور خود را وارد کند.
- هش ورودی را با هش ذخیره‌شده مقایسه کند.
- فقط در صورت تطابق، متد مورد نظر را اجرا کند.

.۴

برنامه باید بتواند به‌صورت کنترل‌شده با خطاهای احتمالی برخورد کند **استثناء مناسب** موارد زیر را در نظر بگیرید و برای هرکدام تعریف یا مدیریت کنید :

اگر فایل pickle :

1. خراب یا خالی بود برنامه نباید کرش کند باید با پیام مناسب("فایل داده ها اسیب دیده است درحال ایجاد نسخه جدید...")فایل را دوباره بسازد

2. اگر کاربر جدیدی با ایمیل تکراری ثبت شود، برنامه باید با یک **Custom Exception** مثل `DuplicateEmailError`جلوی ثبت را بگیرد.
3. اگر فایل هنگام ذخیره‌سازی در دسترس نبود (مثلاً مسیر اشتباه)، خطا را بگیرید و پیام کاربرپسند نمایش دهید.
4. در متدهای مهم مثل حذف کاربر، اگر ایمیل واردشده وجود نداشت، باید خطا را مدیریت کنید (بدون توقف برنامه).

💡نکته‌ی مهم: برای تمرین بیشتر، کلاس اختصاصی `UserError` به‌عنوان پایه‌ی تمام استثناهای کاربر بسازید و سایر خطاها را از آن ارث‌بری دهید.

۵.

1. پروژه را در یک محیط مجازی (`venv`) ایجاد کنید.
2. پکیج‌های مورد استفاده را نصب کنید.
3. یک فایل `requirements.txt` بسازید که بتوان محیط را بازسازی کرد.

۶.

1. با استفاده از ماژول `hashlib.md5` از کل داده‌ی کاربرها یک امضای دیجیتال (checksum) بسازید تا در هر بار اجرا بتوانید تشخیص دهید داده‌ها تغییر کرده‌اند یا نه.
2. اگر فایل تغییر کرده باشد، پیام هشدار چاپ شود.
3. برنامه را طوری بنویسید که اگر هش فعلی با هش فایل قبلی برابر بود، از بارگذاری مجدد صرف‌نظر کند.

خروجی نهایی:

در پایان اجرای برنامه باید:

- لیست کاربران با `pprint` نمایش داده شود.
- کاربر بتواند با رمز صحیح عملیات حساس انجام دهد.
- داده‌ها به‌صورت امن و خودکار در فایل ذخیره شوند.
- برنامه قابل اجرا در محیط مجازی باشد.

سوال دوم:

با استفاده از @dataclass دو کلاس زیر را بسازید:

Product با فیلدهای id, name, price

Order با فیلدهای order_id, customer_email, items (لیستی از محصولات), total_price, created_at

در هنگام ساخت هر سفارش (Order)، total_price باید به‌صورت خودکار محاسبه شود (جمع قیمت تمام محصولات).

برای تولید order_id از هش (hashlib.md5) استفاده کنید تا هر سفارش شناسه‌ی منحصربه‌فرد داشته باشد.

۲.

کلاسی به نام StoreManager بسازید که وظیفه‌ی زیر را داشته باشد:

افزودن محصول جدید

حذف محصول

ثبت سفارش جدید

نمایش لیست سفارش‌ها و محصولات

تمام داده‌ها باید با استفاده از pickle در فایل ذخیره و هنگام اجرای مجدد برنامه بازیابی شوند.

از pprint برای نمایش زیبا و مرتب داده‌ها استفاده کنید.

۳.

یک decorator به نام admin_required بسازید که:

فقط به ادمین (با رمز مشخص) اجازه‌ی انجام عملیات حساس مثل حذف محصول را بدهد.

اگر رمز اشتباه وارد شد، تابع اجرا نشود و پیام مناسبی نمایش دهد.

رمز باید قبل از مقایسه با هش شود (برای امنیت بیشتر).

۴. مدیریت خطاها (Exception Handling)

در برنامه، خطاها را کنترل‌شده مدیریت کنید.

نمونه‌ها:

اگر کاربر بخواهد محصولی را حذف کند که وجود ندارد، خطای ProductNotFoundError مدیریت شود.

اگر فایل داده‌ها پیدا نشود یا خراب باشد، برنامه نباید متوقف شود؛ باید فایل جدید ساخته شود.

اگر کاربری بخواهد سفارشی ثبت کند ولی هیچ محصولی اضافه نکرده باشد، استثناء EmptyOrderError باید پرتاب و مدیریت شود.

💡 نکته: همه‌ی خطاهای خاص فروشگاه می‌توانند از کلاس پایه‌ای به نام StoreError ارث‌بری کنند.

۵.

پروژه را در محیط مجازی (venv) ایجاد کنید.

پکیج‌های مورد استفاده را نصب کنید و فایل requirements.txt بسازید تا دیگران بتوانند محیط را بازسازی کنند.

۶.

برای هر سفارش یک فایل فاکتور بسازید که شامل نام محصولات، قیمت‌ها و مجموع باشد.

نام فایل فاکتور از روی هش سفارش (order_id) ساخته شود (مثلاً invoice_<order_id>.txt).

اگر هنگام ساخت فاکتور خطایی رخ داد (مثل نداشتن دسترسی نوشتن)، برنامه باید آن را با پیام مناسب مدیریت کند و از کرش جلوگیری کند.

خروجی نهایی:

در پایان اجرای برنامه باید:

داده‌های محصولات و سفارش‌ها در فایل pickle ذخیره شوند.

عملیات‌های حساس فقط با رمز ادمین انجام شوند.

خطاها به شکل کاربرپسند مدیریت شوند.

فاکتور سفارش‌ها (در صورت فعال بودن بخش اختیاری) تولید شود.

محیط پروژه با requirements.txt بازتولیدپذیر باشد.