Natürlich, ich erkläre es jetzt **ganz genau, Schritt für Schritt**, so als ob du völliger Python-Noob bist — aber mit dem Ziel, dass du am Ende wirklich verstehst, warum und wie du es so machst. Ich gehe dabei auf jedes Detail ein, damit es dir klar wird, was du tust und warum.

So bauen Profis (mit jahrzehntelanger Erfahrung) eine saubere, modulare Session-Struktur in Python – für einen Noob erklärt

---

1. Warum überhaupt Sessions und saubere Struktur?

Stell dir vor, dein Programm macht viele verschiedene Dinge:

- Passwort verwalten
- Filter verwalten
- E-Mail-Einstellungen ändern
- usw.

Wenn du alles in einer Datei und in vielen einzelnen Funktionen hast, verlierst du schnell den Überblick.

Deshalb teilst du dein Programm in kleine, überschaubare „Bausteine“ auf. Jeder Baustein kümmert sich um genau eine Aufgabe — so genannte **Sessions**.

So kannst du:

- Jede Session separat bauen, testen und verbessern.
- Verschiedene Sessions unabhängig voneinander nutzen.
- Texte (z.B. Eingabeaufforderungen) flexibel anpassen, ohne Code zu verändern.

---

2. Wie machst du das?

Schritt 1: Eine „Basisklasse“ erstellen

- Eine Basisklasse ist wie eine Vorlage, die sagt:
  „Jede Session, die du baust, muss eine Methode `run()` haben.“
- Außerdem kann sie schon Dinge vorbereiten, z. B. speichern, welche Texte du verwenden willst.

Warum? Weil so alle Sessions die gleiche Grundstruktur haben. Du weißt immer: „Ich kann `.run()` aufrufen, und die Session macht ihre Arbeit.“

Schritt 2: Konkrete Sessions bauen, die von der Basisklasse erben

- Eine Session für Passwörter z. B. fragt nach dem Passwort, validiert es und speichert es.
- Eine andere Session für Filter fragt nach Filterparametern.

Diese Sessions bekommen beim Start ihre eigenen Texte (Prompts) übergeben. So sind sie flexibel.

---

3. Was ist ein Prompt und warum auslagern?

- Prompts sind die Texte, die du dem Nutzer zeigst, z.B. „Bitte Passwort eingeben“.
- Du legst sie in einer eigenen Datei als Wörterbuch (dict) ab.
- So kannst du die Texte ändern, übersetzen oder anpassen, ohne am Programmcode herumzufummeln.

---

4. Wo kommen die Texte und Sessions zusammen?

- Du erstellst im Hauptprogramm (`main.py`) eine Session, z.B. `PasswordSession` und gibst ihr die passenden Texte mit.
- Dann rufst du `run()` auf, und die Session fragt den Nutzer nach Eingaben — mit den von dir übergebenen Texten.

---

5. Konkret umsetzen: Schritt für Schritt

5.1 Ordner & Dateien anlegen

- Erstelle einen Ordner `sessions/` für alle Session-Dateien.
- Erstelle eine Datei `sessions/base_session.py` für die Basisklasse.
- Erstelle eine Datei `sessions/password_session.py` für die Passwort-Session.
- Erstelle eine Datei `resources/prompts.py` für alle Texte.

5.2 Basisklasse in `base_session.py`

- Definiere eine Klasse `BaseSession`.
- Sie hat:
    - Einen Konstruktor, der ein Wörterbuch `prompts` annimmt und speichert.
    - Eine leere Methode `run()`, die du später in den Sessions überschreibst.

So weiß jede Session, welche Texte sie benutzen soll.

5.3 Passwort-Session in `password_session.py`

- Erstelle eine Klasse `PasswordSession`, die von `BaseSession` erbt.
- Im Konstruktor rufst du `super().__init__(prompts)` auf, um die Prompts zu übernehmen.
- In `run()` benutzt du `input(self.prompts['password_input'])` um den Nutzer zu fragen.
- Du speicherst das Ergebnis als `self.password`.

5.4 Prompts in `prompts.py`

- Lege ein Wörterbuch an, z.B.:

```
PASSWORD_PROMPTS = {
  "password_input": "Bitte geben Sie Ihr Passwort ein: ",
  "password_invalid": "Das Passwort ist zu kurz, bitte erneut eingeben.",
}
```

- Später kannst du hier weitere Texte für Filter oder andere Sessions hinzufügen.

5.5 Hauptprogramm `main.py`

- Importiere die `PasswordSession` und die `PASSWORD_PROMPTS`.
- Erstelle eine Instanz: `session = PasswordSession(PASSWORD_PROMPTS)`
- Starte die Session mit `session.run()`.
- Greife nachher auf `session.password` zu.

---

6. Warum so?

- **Klarer Code:** Du weißt sofort, welche Session was macht.
- **Wiederverwendbar:** Du kannst leicht neue Sessions hinzufügen, z.B. `FilterSession`.
- **Flexible Texte:** Du kannst Prompts jederzeit ändern, ohne die Sessions anzupassen.
- **Erweiterbar:** Du kannst in Sessions auch komplexere Logik unterbringen.
- **Testbar:** Du kannst jede Session separat testen.

---

7. Wie kannst du das in Zukunft für andere Aufgaben nutzen?

Wenn du z.B. eine Session für Filter erstellen willst:

- Erstelle `FilterSession`, die von `BaseSession` erbt.
- Lege in `prompts.py` ein `FILTER_PROMPTS` Wörterbuch an.

- Übergib diese Prompts beim Erstellen von `FilterSession`.
- Implementiere die Filterlogik in `FilterSession.run()`.

---

8. Tipps für dich als Noob:

- **Mach es Schritt für Schritt.** Fang mit einer Basisklasse an, dann eine einfache Session.
- **Teste jede Session einzeln.** Rufe `run()` auf und prüfe, ob sie Eingaben erwartet.
- **Schreibe Kommentare in deinen Code.** So verstehst du später, was du gemacht hast.
- **Mach die Prompts klar und verständlich.** So findest du Fehler schneller.
- **Speichere Ergebnisse als Instanzvariablen.** Dann kannst du später in `main.py` darauf zugreifen.

Zusammenfassung (ganz kurz):

- **Basisklasse für Struktur**
- **Konkrete Sessions mit eigener Logik**
- **Texte (Prompts) auslagern in dicts**
- **Hauptprogramm verbindet alles**
- **Sauber, flexibel, wartbar**

Wenn du das so machst, wirst du schon mit wenigen Zeilen Code ein sehr professionelles, strukturiertes Programm haben, das du später leicht erweitern und pflegen kannst.

Falls du möchtest, kann ich dir auch die genaue Reihenfolge geben, wie du die Dateien anlegst und was in jede Zeile gehört — aber so, dass du selbst den Code schreibst und verstehst.

Möchtest du das?