Perfekt, das ist eine sehr sinnvolle Idee:
Du bekommst jetzt einen **detaillierten Lernzettel**, mit dem du **Eingaben mit `prompt_toolkit`** wie ein Profi umsetzen kannst – **ohne Internet**, nur mit deinem Editor und deinen eigenen Händen.

📘 Lernzettel: Eingabe & Validierung mit `prompt_toolkit`

> Ziel: Du lernst, wie du Benutzereingaben (z. B. E-Mail-Adresse, Passwort, Portnummern etc.) **interaktiv, geprüft, professionell** abfragen kannst – mit *`prompt_toolkit`*.

🧠 1. Grundidee: Wie läuft eine Eingabe mit `prompt_toolkit` ab?

Statt `input()` nutzt du:

```
from prompt_toolkit import prompt
```

Dieser `prompt()`-Aufruf unterstützt viele Profi-Funktionen:

Funktion	Zweck
`prompt()`	Textbasiertes Eingabefeld im Terminal
`Validator`	Gültigkeit der Eingabe prüfen
`Completer`	Tab-Vervollständigung
`is_password=True`	Eingabe maskieren (z. B. für Passwörter)
`bottom_toolbar`	Zusätzliche Hinweise anzeigen

🛠️ 2. Eingabestruktur für dein Projekt

Du wirst **pro Eingabe eine eigene Funktion** schreiben, z. B.:

- `ask_email()`
- `ask_imap_server()`
- `ask_smtp_server()`
- `ask_port()`
- `ask_password()`
- `ask_default_reply_message()`

Diese Funktionen nutzen **`prompt()`** mit den gewünschten Features.

🧪 3. Eingabe validieren mit `Validator`

a) Was ist ein Validator?

Ein Objekt, das prüft, ob der eingegebene Text gültig ist.
Du implementierst dazu eine eigene Klasse, die von `Validator` erbt.

Beispiel-Schema:

```
from prompt_toolkit.validation import Validator, ValidationError

class EmailValidator(Validator):
    def validate(self, document):
        if not ...:  # Bedingung
            raise ValidationError(
                message='Das ist keine gültige E-Mail.',
                cursor_position=...  # Optional: wo der Fehler ist
            )
```

b) Typische Validierungen, die du brauchst:

Eingabe	Prüfidee
E-Mail	Regex + „@“ enthalten + bekannte Domain
Portnummer	Muss Zahl sein + 1 ≤ Port ≤ 65535
Servernamen	Darf kein Leerzeichen enthalten
Passwort	Nicht leer

⚡ 4. Autovervollständigung mit `Completer`

a) Was ist ein Completer?

Ein Objekt, das **Tab-Vervollständigungen** anbietet.

```
from prompt_toolkit.completion import WordCompleter

server_completer = WordCompleter(
    ['imap.gmail.com', 'imap.mailbox.org', 'imap.strato.de'],
    ignore_case=True
)
```

Dann bei `prompt()`:

```
prompt("IMAP-Server: ", completer=server_completer)
```

🔐 5. Passworteingabe (Maskierung)

Einfach mit:

```
prompt("Passwort: ", is_password=True)
```

Das ersetzt Eingaben mit `****`.

📋 6. Zusätzliche Hinweise anzeigen (`bottom_toolbar`)

Du kannst unten im Terminal Hinweise einblenden:

```
prompt(
    "E-Mail: ",
    bottom_toolbar="Beispiel: name@gmail.com"
)
```

🧩 7. Strukturierung deiner Eingabelogik

Du baust ein eigenes Modul: z. B. `input_flow.py`.

Darin:

- Funktionen für jede Eingabe
- Gemeinsamer Funktionsaufruf `run_initial_setup()` z. B. in `ResetInit.py`
- Rückgabe aller Daten als `dict` oder speichern in `.env` / `settings.json`

🧹 8. Eingaben speichern (außerhalb dieses Lernzettels)

- Nutze z. B. das `json`-Modul oder `dotenv`
- Speichere Daten in Datei, sobald sie eingegeben wurden
- Beachte: Passwort ggf. verschlüsseln oder sichern

🧠 9. Best Practices

Technik	Wann einsetzen?
`Validator`	Immer, bei prüfbaren Eingaben
`Completer` (WordCompleter)	Wenn feste Liste vorliegt
`is_password=True`	Bei geheimen Daten
`bottom_toolbar`	Bei Hilfe-/Hinweistexte
Eigene Funktion je Eingabe	Sauberer, wartbarer Code

📌 Beispiel-Ablauf deiner Eingabesequenz (ohne Code):

1. Starte `ResetInit.py`
2. Rufe `run_initial_setup()` aus `input_flow.py` auf
3. `run_initial_setup()` fragt:

- E-Mail (validiert + Domainprüfung)
- IMAP/SMTP Server (automatisch gesetzt oder manuell)
- Portnummern (per `Validator`)
- Passwort (maskiert)
- Standardantwort (beliebiger Text, kein Validator)

4. Bei Erfolg → speichere alles

🎓 Bonus-Tipp: Wiederverwendbare Bausteine

Wenn du dir z. B. einen **`IntegerValidator`**, `EmailValidator`, `PortValidator`** etc. anlegst, kannst du diese immer wieder in anderen Projekten verwenden – du baust dir damit dein eigenes kleines CLI-Toolkit.

❓ Möchtest du weiterführende Lernzettel?

- `prompt_toolkit` Multiline-Eingaben
- `prompt_toolkit` mit `asyncio`
- `prompt_toolkit` Formulareingabe mit mehreren Feldern gleichzeitig (z. B. tabbar)

Sag einfach Bescheid – ich begleite dich gern weiter!