Sehr gerne! Ich erkläre dir das jetzt **ganz langsam, Schritt für Schritt**, sodass du **jeden Teil wirklich verstehst**, auch ohne Vorkenntnisse in E-Mail-Parsing oder Filterlogik.

🎯 Ziel des Ganzen:

Du möchtest deinem Programm **beibringen**, zu entscheiden:

- ❌ Auf welche E-Mails **NICHT** geantwortet werden soll (Ausschlussfilter / „Blacklist“)
- ✅ Auf welche E-Mails **geantwortet** werden soll (Einschlussfilter / „Whitelist“)

➡️ Das heißt, das Programm muss **jede E-Mail untersuchen**, und dann **entscheiden**, ob es antwortet oder nicht.

🧱 Was du brauchst

Um das umzusetzen, brauchst du diese 5 Dinge im Code:

---

1. Zugriff auf die E-Mail-Daten

Du musst aus der Mail herauslesen:

- Wer hat sie geschickt? 👉 `From`
- Was steht im Betreff? 👉 `Subject`

Dafür brauchst du die E-Mail als `EmailMessage`-Objekt (die bekommst du nach dem `fetch()` und `email.message_from_bytes()`).

➡️ Code-Dummy:

```
# msg ist dein E-Mail-Objekt
from_ = msg.get("From")
subject = msg.get("Subject")
```

---

2. Die Filterregeln laden

Du speicherst deine Regeln in einer JSON-Datei, z. B. `filters.json`.

Beispiel `filters.json`:

```
{
  "include": {
    "from": ["chef@firma.de", "teamlead@"],
    "subject_contains": ["Dringend", "Projekt", "Update"]
  },
  "exclude": {
    "from": ["newsletter@", "noreply@"],
    "subject_contains": ["Rabatt", "Werbung"]
  }
```

```
}
```

➡️ Diese Datei musst du **mit `json` laden**:

```
import json

with open("json/filters.json", "r") as f:
    filters = json.load(f)
```

Jetzt hast du z. B.:

```
filters["exclude"]["from"]  # → ["newsletter@", "noreply@"]
filters["include"]["subject_contains"]  # → ["Dringend", "Projekt"]
```

---

3. Filter anwenden (Vergleichen)

Jetzt prüfst du, ob `from_` oder `subject` Dinge enthalten, die in den Filtern stehen.

Dazu brauchst du:

- Eine **`for`-Schleife**, um alle Filter durchzugehen
- Einen **`if ... in ...`-Vergleich**, um zu schauen, ob z. B. `"newsletter@" in from_"`

➡️ **Dummycode zum Ausschließen:**

```
for wort in filters["exclude"]["subject_contains"]:
    if wort.lower() in subject.lower():
        print("Diese E-Mail wird ignoriert.")
```

Genauso mit `"from"`.

➡️ **Dummycode zum Einschließen:**

```
for erlaubter_absender in filters["include"]["from"]:
    if erlaubter_absender.lower() in from_.lower():
        print("Diese E-Mail darf beantwortet werden.")
```

---

4. Entscheidungsfunktion bauen

Jetzt packst du die Filterlogik in eine Funktion wie:

```
def should_reply(from_, subject, filters):
    # Erst prüfen: Ist die E-Mail gesperrt?
    for blockiert in filters["exclude"]["from"]:
        if blockiert.lower() in from_.lower():
            return False  # NICHT antworten

    for verboten in filters["exclude"]["subject_contains"]:
        if verboten.lower() in subject.lower():
            return False

    # Dann prüfen: Ist sie erlaubt?
    includes_set = filters["include"]["from"] or filters["include"]
["subject_contains"]
```

```
    if includes_set:
        erlaubt = any(w.lower() in from_.lower() for w in filters["include"]
["from"]) \
                  or any(w.lower() in subject.lower() for w in
filters["include"]["subject_contains"])
        return erlaubt

    # Wenn keine Einschlussregeln gesetzt sind → immer antworten (sofern nicht
ausgeschlossen)
    return True
```

Diese Funktion kannst du später in deinem Code so verwenden:

```
if should_reply(from_, subject, filters):
    antworte_auf_email()
```

---

5. Sonderfall: „From“-Adresse parsen

Absender wie:

```
"Max Mustermann <max@firma.de>"
```

musst du **parsen**, damit du nur die Adresse rausbekommst.

➡️ Dafür gibt’s diese Hilfsfunktion:

```
from email.utils import parseaddr

adresse = parseaddr(msg.get("From"))[1]  # → gibt nur "max@firma.de"
```

💡 Empfohlen: Nutze **immer `parseaddr`**, bevor du die Filter vergleichst.

🧪 Beispiel-Ablauf

```
msg = email.message_from_bytes(raw_email)

from_ = parseaddr(msg.get("From"))[1]   # Absenderadresse extrahieren
subject = msg.get("Subject")            # Betreff extrahieren

if should_reply(from_, subject, filters):
    print("Ja, antworten!")
else:
    print("Nein, ignorieren.")
```

✅ Zusammenfassung

Schritt	Was passiert?
`msg.get("From")`	Absender auslesen
`parseaddr(...)`	Nur E-Mail-Adresse rausziehen
`msg.get("Subject")`	Betreff auslesen
`json.load(...)`	Filterregeln aus Datei laden

Schritt	Was passiert?
`for ... in ...:` + `if ... in ...:`	Regeln prüfen
Funktion `should_reply(...)`	Automatische Entscheidung

Wenn du willst, kann ich dir als nächstes helfen:

- Die Filterdatei dynamisch über `prompt_toolkit` zu erstellen
- Einen Testmodus zu machen, der die Filter überprüft
- Oder eine Konfigurationsoberfläche im Terminal

Sag einfach, was du selbst machen möchtest oder wo du noch unsicher bist.