Alles klar! Hier kommt dein ausführlicher, gut kommentierter Lernzettel zum Python-Modul **`imaplib`** – für **E-Mails empfangen und lesen via IMAP**, praxisnah und Schritt für Schritt erklärt.

Lernzettel: Python-Modul `imaplib`

---

1. Was ist `imaplib`?

- `imaplib` ist das Standardmodul, um mit Python **per IMAP-Protokoll auf Mail-Server zuzugreifen** und E-Mails abzurufen.
- IMAP (Internet Message Access Protocol) erlaubt dir, Mails auf dem Server zu verwalten, ohne sie komplett herunterzuladen.
- Du kannst z.B. Mails suchen, lesen, als gelesen markieren, löschen etc.

---

2. Grundprinzip: Verbindung zum IMAP-Server aufbauen

- Verbinde dich per SSL oder unverschlüsselt mit dem Server.
- Melde dich mit Benutzername und Passwort an.
- Wähle einen Mail-Ordner (z.B. INBOX).
- Suche und hole Mails.
- Schließe die Verbindung.

---

3. Einfaches Beispiel: Einloggen und INBOX auswählen

```
import imaplib

imap_server = "imap.example.com"
username = "deine_email@example.com"
password = "dein_passwort"

# Verbindung per SSL aufbauen (Port 993)
mail = imaplib.IMAP4_SSL(imap_server)

# Login
mail.login(username, password)

# INBOX auswählen (nur lesen, readonly=False erlaubt auch Änderungen)
mail.select("INBOX")

print("Erfolgreich eingeloggt und INBOX geöffnet")
```

---

4. Mails suchen

Mails suchst du mit `search()`, z.B. alle ungelesenen:

```
status, messages = mail.search(None, 'UNSEEN')

# messages ist ein Byte-String mit IDs, z.B. b'1 2 3'
mail_ids = messages[0].split()

print("Ungelesene Mails:", mail_ids)
```

---

5. Mail-Inhalt holen

Eine Mail holst du mit `fetch()` anhand ihrer ID:

```
latest_email_id = mail_ids[-1]  # letzte Mail-ID

status, msg_data = mail.fetch(latest_email_id, '(RFC822)')

# msg_data ist eine Liste mit Tupeln, wir nehmen den ersten Eintrag
raw_email = msg_data[0][1]  # das ist der komplette Roh-Byte-String der Mail

print(raw_email[:100])  # Ausgabe der ersten 100 Byte zum Test
```

---

6. Mail parsen mit `email`-Modul

`raw_email` ist ein Byte-String, den du mit `email` in ein Message-Objekt umwandelst:

```
import email

msg = email.message_from_bytes(raw_email)

print("Von:", msg['From'])
print("Betreff:", msg['Subject'])

# Body auslesen (einfacher Fall)
if msg.is_multipart():
    for part in msg.walk():
        if part.get_content_type() == 'text/plain':
            body =
part.get_payload(decode=True).decode(part.get_content_charset())
            print("Body:", body)
else:
    body = msg.get_payload(decode=True).decode(msg.get_content_charset())
    print("Body:", body)
```

---

7. Beispiel: Alle ungelesenen Mails abrufen und Betreff ausgeben

```
status, messages = mail.search(None, 'UNSEEN')
mail_ids = messages[0].split()
```

```
for mail_id in mail_ids:
    status, msg_data = mail.fetch(mail_id, '(RFC822)')
    raw_email = msg_data[0][1]
    msg = email.message_from_bytes(raw_email)
    print("Mail von:", msg['From'])
    print("Betreff:", msg['Subject'])
    print("---")
```

---

8. Mail als gelesen markieren

```
mail.store(mail_id, '+FLAGS', '\\Seen')
```

---

9. Verbindung beenden

```
mail.close()  # schließt ausgewählten Mailbox-Ordner
mail.logout() # beendet Sitzung und Verbindung
```

---

10. Zusammenfassung & Tipps

- `imaplib` stellt Verbindung zum Mailserver per IMAP her.
- Login mit Benutzername/Passwort.
- Mailbox (z.B. INBOX) auswählen.
- Mails suchen (z.B. ungelesene, bestimmte Absender).
- Mails mit `fetch()` holen, dann mit `email`-Modul parsen.
- Mails als gelesen markieren oder löschen möglich.
- Verbindung mit `logout()` schließen.

---

11. Fehlerbehandlung (Grundgerüst)

```
try:
    mail = imaplib.IMAP4_SSL(imap_server)
    mail.login(username, password)
    mail.select("INBOX")
    # ... weitere Aktionen
except imaplib.IMAP4.error as e:
    print("IMAP Fehler:", e)
```

Wenn du magst, kann ich dir als nächstes zeigen, wie du mit `concurrent.futures` parallel mehrere Mails prüfst oder mehrere IMAP-Ordner gleichzeitig ausliest.

Willst du das?