Sehr gerne! Hier ist ein **ausführlicher, einsteigerfreundlicher Lernzettel zu `argcomplete`**, der wie dein gewünschter `concurrent.futures`-Lernzettel aufgebaut ist – **für Leute mit Python-Grundkenntnissen**, aber ohne Erfahrung mit CLI-Autovervollständigung.

🧠 Lernzettel: `argcomplete` – Autovervollständigung für `argparse`

✅ Ziel

Du lernst, wie du mit **`argcomplete`** deine Python-Programme so ausstattest, dass Nutzer im **Terminal per `Tab`-Taste Befehle automatisch vervollständigen** können – genau wie bei `git`, `docker` oder `pip`.

🧩 Was ist `argcomplete`?

`argcomplete` ist ein Zusatzmodul für Python, das dein bestehendes `argparse`-System um **Shell-Autovervollständigung** erweitert – **ohne** dein Code-Design komplett zu verändern.

🔍 Typischer Nutzen

- Vorschlagen von Optionen wie `--mode`, `--config`, `--email`
- Vervollständigen von Argumentwerten (z. B. `--mode run`)
- Anzeigen von möglichen Auswahlwerten bei gedrückter `Tab`-Taste

---

📦 Installation

```
pip install argcomplete
```

Empfohlen: globale Installation oder in einem venv.

🔧 Setup im Script (Basics)

```
import argparse
import argcomplete  # Wichtig!

parser = argparse.ArgumentParser(description="Beispielprogramm mit
Autovervollständigung")
parser.add_argument("--mode", choices=["setup", "run", "edit"])
parser.add_argument("--loglevel", choices=["info", "debug", "warning"])
```

```
# Aktiviert Autovervollständigung
argcomplete.autocomplete(parser)

args = parser.parse_args()
print(args)
```

🖥️ Terminal-Vorbereitung (nur 1x nötig)

🟢 Variante 1: Global aktivieren (empfohlen für wiederholte Nutzung)

```
activate-global-python-argcomplete --user
```

Dann funktioniert `argcomplete` für **alle Python-Skripte automatisch**, wenn Shell richtig konfiguriert ist.

🟡 Variante 2: Nur für 1 Skript (lokal aktiv)

```
eval "$(register-python-argcomplete python3 main.py)"
```

Danach kannst du `main.py` wie folgt nutzen:

```
python3 main.py --[Tab]
# Vorschläge: --mode, --loglevel, --help, ...
```

🔍 Wichtige Begriffe

Begriff	Bedeutung
argparse	Standardmodul für CLI-Argumente
argcomplete	Fügt argparse Tab-Autovervollständigung hinzu
register-python-argcomplete	Erstellt Autovervollständigungsfunktion
eval ...	Aktiviert Bash-Funktion vorübergehend

💡 Vorteile gegenüber reiner `input()`-Eingabe

Feature	`argcomplete`	`input()` / `prompt_toolkit`
Tab-Autovervollständigung	✅	✅ (`prompt_toolkit`)
Shell-nativ	✅	❌
Auswahl durch Optionen	✅	nur manuell
Sicher gegen Tippfehler	✅ (`choices=`)	❌
Für technische Nutzer	✅	✅

🔐 Erweiterte Beispiele

📁 Beispiel 1: Dateinamen automatisch vorschlagen

```
from argparse import ArgumentParser
from argcomplete.completers import FilesCompleter
import argcomplete

parser = ArgumentParser()
parser.add_argument("--file", help="Wähle eine Datei",
completer=FilesCompleter())

argcomplete.autocomplete(parser)
args = parser.parse_args()
print(args)
```

Jetzt kannst du mit `--file [TAB]` durch Dateisystem navigieren!

🧠 Beispiel 2: Eigene dynamische Vorschläge

```
def custom_completer(prefix, parsed_args, **kwargs):
    return ["info", "warnung", "kritisch", "fatal"]

parser.add_argument("--stufe", help="Wähle eine Stufe",
completer=custom_completer)
```

Bei `--stufe [TAB]` erscheinen deine eigenen Vorschläge.

🧠 Fehler vermeiden

Problem	Lösung
Kein Tab-Vervollständigen funktioniert	`eval` vergessen oder Shell nicht neu gestartet
Keine Vorschläge sichtbar	`choices=` oder `completer=` fehlt
Terminal zeigt Python-Code an	`argcomplete.autocomplete()` nicht eingefügt
Funktioniert nicht in Windows CMD	Nutze Bash, Git Bash oder WSL

🧯 So beendest du ein Programm sauber

`argcomplete` beeinflusst nicht deine Programm-Logik – aber dein Code kann z. B. per `KeyboardInterrupt` (STRG+C) sauber abbrechen:

```
try:
    while True:
        ...
except KeyboardInterrupt:
    print("Beendet.")
```

✅ Zusammenfassung – Warum du `argcomplete` verwenden solltest

Vorteil	Beschreibung
🚀 Schneller	Nutzer müssen keine Optionen auswendig wissen
🤝 Benutzerfreundlich	Besonders für technisch unerfahrene Nutzer angenehm
⚙️ Einfach zu integrieren	Ergänzt `argparse` ohne viel Mehraufwand
🛠️ Sehr anpassbar	Eigene Vorschläge, Dateiauswahl, Optionen
🧩 Gut kombinierbar	mit `prompt_toolkit` für interaktive Details

🔚 Fazit

Wenn du ein CLI-Tool entwickelst, ist `argcomplete` **ideal für den Einstieg in professionelle Terminal-Nutzung**. Es wirkt dezent, ist einfach zu integrieren und erhöht sofort den Bedienkomfort.

Wenn du möchtest, kann ich dir helfen:

- die **`argcomplete`-Shell-Integration** bei dir lokal einzurichten,
- oder **dynamische Vorschläge** für z. B. E-Mail-Konten, Domains etc. zu definieren.

🔄 **Nächster Schritt für dich**
👉 Baue dein erstes CLI-Argument mit `choices=`
👉 Dann: aktiviere `argcomplete` per `eval`
👉 Teste `Tab` und passe an!

Möchtest du danach ein Beispiel kombinieren mit `prompt_toolkit`, um Eingaben + Argumente zu mischen?