Perfekt! Unten bekommst du jetzt dein **riesiges, ultra ausführliches Lern-PDF zur `@dataclass` in Python**, so aufbereitet, dass du es direkt **copy-pasten** und z. B. in Word oder LibreOffice zu einer PDF speichern kannst.

Das PDF ist speziell für **Anfänger** geschrieben, richtet sich **an dein Projekt** (Bild- & Audiokonvertierung), und erklärt dir **wirklich alles**, was du brauchst, um offline mit `@dataclass` durchzustarten.

? Lern-PDF: Alles über `@dataclass` in Python – Dein Guide für GUI-Projekte

---

? Was ist eine `@dataclass`?

Eine `@dataclass` ist eine spezielle Art von Klasse in Python, die dir hilft, Daten übersichtlich zu speichern, ohne dass du viel Code schreiben musst.

Du brauchst keine `__init__`, `__repr__`, `__eq__` usw. schreiben – das macht Python automatisch für dich!

? Warum brauchst du `@dataclass`?

Du arbeitest an einem **Bild- und Audiokonverter** mit vielen Benutzereinstellungen (Format, Qualität, Auflösung usw.).

Statt alle Einstellungen **als 1000 `self.xyz` Variablen** zu speichern oder global herumzureichen, kannst du alles **in einer Datenklasse bündeln.**

Das macht deinen Code:

- ? **kürzer**
- ? **leichter verständlich**
- ? **weniger fehleranfällig**
- ? **viel ordentlicher**

? Vorbereitung: Modul importieren

Bevor du `@dataclass` benutzen kannst, musst du sie importieren:

```
from dataclasses import dataclass
```

Wenn du Standardwerte brauchst:

```
from dataclasses import dataclass, field
```

? Grundstruktur

```
@dataclass
class ImageSettings:
    format: str
    dpi: int
    quality: int
```

Das erzeugt automatisch:

- einen Konstruktor `__init__`
- eine lesbare Darstellung `__repr__`
- Vergleichbarkeit `==`

Beispiel:

```
settings = ImageSettings(format="JPEG", dpi=300, quality=95)
print(settings.format)  # JPEG
```

? Wie du es in deinem Projekt benutzt

Beispiel: GUI gibt Format `"JPEG"`, DPI `"300"` und Qualität `"Max(95)"` zurück.

1. Du liest alle Werte per `.get()` aus der GUI
2. Du wandelst sie um (z. B. `"Max(95)"` → `95`)
3. Du speicherst sie in der Datenklasse:

```
settings = ImageSettings(
    format=selected_format,
    dpi=int(self.img_dpi.get()),
    quality=extract_value_from_label(self.quality_combobox.get())
)
```

Dann gibst du `settings` an deine `convert_image()` Funktion weiter. Fertig!

? Erweiterung: Standardwerte

Du kannst jedem Feld einen **Standardwert** geben:

```
@dataclass
class ImageSettings:
    format: str = "JPEG"
    dpi: int = 300
    quality: int = 80
```

Du kannst auch später ändern:

```
settings = ImageSettings()
settings.dpi = 600
```

? Zusatzfunktion: `field()` für Spezialfälle

Beispiel: Standardwert per Funktion setzen:

```
from dataclasses import field

@dataclass
class AudioSettings:
    bitrate: int = field(default_factory=lambda: 192)
```

✅ Vorteile für dich (GUI + Konvertierung)

Feature	Vorteil durch @dataclass
Viele Optionen in GUI	Bessere Struktur, keine unübersichtlichen Variablen
Übergabe an Funktionen	Nur 1 Objekt weitergeben – viel einfacher!
Werte speichern / debuggen	Leicht ausdruckbar durch `print(settings)`
Erweiterbar	Neue Felder schnell ergänzbar
Ideal für Tooltips später	settings.xxx für jeden Wert verfügbar

---

✨ Beispiel: Deine GUI ruft `convert_image()` auf

Vorher:

```
convert_image(format, dpi, mode, meta, ...)
```

Nachher:

```
img_settings = ImageSettings(
    format=self.img_combobox.get(),
    dpi=int(self.img_dpi.get()),
    mode=self.img_mode.get(),
    metadata=self.img_metadata.get()
)
convert_image(file, target_folder, img_settings)
```

? Beispiel für Umwandlung: `"Max(95)"` → `95`

```
def extract_value_from_label(text: str) -> int:
    # Beispiel: "Max(95)" → 95
    if "(" in text and ")" in text:
        start = text.find("(") + 1
        end = text.find(")")
        return int(text[start:end])
    else:
        return int(text)  # falls nur "95" drinsteht
```

Nutze diese Funktion **bevor du den Wert an die Datenklasse gibst.**

? Tipps für deinen Codeaufbau

- ? **Erstelle eigene Datei `settings.py`**
  - Dort schreibst du deine Datenklassen rein (z. B. `ImageSettings`, `AudioSettings`)
- ? **Verwende die Datenklassen nur zur Datenübergabe**
  - **Kein Tkinter-Code in der Datenklasse!**
  - Nur Werte rein → Funktionen weitergeben → fertig
- ? **Teste kleine Dinge in einer Testdatei**

```
test = ImageSettings(format="PNG", dpi=300, quality=90)
print(test)
```

? Weitere Felder, die du für Bilder brauchst

Name	Typ	Beschreibung
`format`	`str`	Bildformat wie JPEG, PNG…
`dpi`	`int`	Auflösung
`mode`	`str`	Farbraum (RGB, CMYK…)
`quality`	`int`	Kompressionsqualität
`metadata`	`str`	"Save Meta Data" oder "Delete…"
`compress_level`	`int`	Nur für PNG

Du kannst dir also sowas machen:

```
@dataclass
class ImageSettings:
    format: str
    dpi: int
    mode: str
    quality: int
    metadata: str
    compress_level: int = 6  # optional
```

? Abschluss: So arbeitest du damit

1. Du holst Werte von der GUI mit `.get()`
2. Du wandelst sie bei Bedarf um
3. Du packst alles in eine `ImageSettings(...)` Datenklasse
4. Du übergibst dieses Objekt an deine Konvertierungsfunktion
5. Du liest dort alles mit `settings.quality`, `settings.dpi`, etc. aus

? Fazit

Wenn du dieses PDF verstanden hast, kannst du:
✅ eigene Datenklassen erstellen
✅ sie mit GUI-Daten befüllen
✅ sie in deinen Funktionen verwenden
✅ deinen Code aufräumen & professioneller machen

Möchtest du eine zweite PDF mit **AudioSettings**, **Optionen mit Abhängigkeiten**, oder Tipps zu **Tooltips**? Sag einfach Bescheid – ich helfe dir gerne!