Natürlich! Hier ist dein **ausführlicher Lernzettel zur Eingabe-Validierung mit `prompt_toolkit`**, inklusive nützlicher **Stringfunktionen**, mit denen du Eingaben effektiv prüfen kannst.

🧾 Lernzettel: Validierung von Benutzereingaben mit `prompt_toolkit`

🎯 Ziel

Du lernst, **wie man Eingaben validiert**, also prüft, ob der Benutzer sinnvolle Daten eingegeben hat (z. B. gültige E-Mail-Adresse, nicht-leeres Passwort, gültiger Port etc.). Außerdem zeige ich dir **wichtige Python-Stringmethoden**, die dir helfen, diese Prüfungen sauber umzusetzen.

🧱 1. Grundlagen der Validierung

Was ist Validierung?

Validierung heißt: **Prüfen von Eingaben**, bevor sie weiterverarbeitet werden.

Beispiele:

- Ist die Eingabe **nicht leer**?
- Ist das eine **gültige E-Mail-Adresse**?
- Ist der eingegebene **Port eine Zahl zwischen 1–65535**?

🧰 2. Zwei Wege der Validierung

Methode	Wann sinnvoll?
Direkt bei der Eingabe (live)	z. B. bei `prompt_toolkit` mit `Validator`
Nach der Eingabe (nach `Enter`)	z. B. mit `if not email.endswith("@...")`

In der Praxis werden **beide Methoden** kombiniert:

1. Direktes Feedback für offensichtliche Fehler.
2. Tiefere Prüfung nach dem Ausfüllen.

📦 3. Validierung mit `prompt_toolkit.Validator`

Wie funktioniert das?

Du definierst eine **Validator-Klasse**, die überprüft, ob der eingegebene Text gültig ist. Sie wird automatisch beim Tippen (oder nach Enter) aufgerufen.

```
from prompt_toolkit.validation import Validator, ValidationError

class EmailValidator(Validator):
    def validate(self, document):
        text = document.text
        if "@" not in text or "." not in text:
            raise ValidationError(
                message="Bitte gib eine gültige E-Mail-Adresse ein.",
                cursor_position=len(text)  # setzt den Cursor ans Ende
            )
```

Verwendung mit `TextArea`:

```
from prompt_toolkit.widgets import TextArea

email_input = TextArea(
    prompt="E-Mail: ",
    validator=EmailValidator(),
    validate_while_typing=True
)
```

🧠 4. Wichtige Stringfunktionen für Validierung

Funktion	Beispiel	Zweck
`str.strip()`	`" abc ".strip()` → `"abc"`	Leerzeichen entfernen
`str.lower()`	`"ABC".lower()` → `"abc"`	Kleinbuchstaben vergleichen
`str.isdigit()`	`"123".isdigit()` → `True`	Prüfen ob nur Ziffern
`str.isalpha()`	`"abc".isalpha()` → `True`	Nur Buchstaben?
`str.isalnum()`	`"abc123".isalnum()` → `True`	Buchstaben + Zahlen
`str.startswith()`	`"email@...".startswith("mail")`	Beginnt mit ...
`str.endswith()`	`"email@...".endswith(".de")`	Endet auf ...
`in`-Operator	`"@" in text`	Enthält Zeichen
`try/except int(text)`	Port prüfen → `int("abc")` → Error	String in Zahl konvertieren

🧪 5. Validierung nach Eingabe

```
# Beispiel: Port prüfen
port_str = "587"
if not port_str.isdigit():
    print("Kein gültiger Port!")
else:
```

```
    port = int(port_str)
    if port < 1 or port > 65535:
        print("Port muss zwischen 1 und 65535 liegen.")
```

🧰 6. Tipps zur Fehlerbehandlung

- **Nie blind weiterverarbeiten!** Immer prüfen.
- **Dem Benutzer helfen:** Konkrete Hinweise anzeigen, z. B. "Bitte gib eine gültige Zahl ein".
- **Felder wiederholen lassen:** z. B. mit Schleife oder erneutem Prompt.

🧪 7. Beispielanwendung: Benutzername + Port

```
from prompt_toolkit.application import Application
from prompt_toolkit.widgets import TextArea, Label, Button
from prompt_toolkit.layout import Layout, HSplit
from prompt_toolkit.validation import Validator, ValidationError

# Benutzername-Validator
class UsernameValidator(Validator):
    def validate(self, document):
        text = document.text.strip()
        if not text or not text.isalnum():
            raise ValidationError(message="Nur Buchstaben/Zahlen erlaubt.")

# Port-Validator
class PortValidator(Validator):
    def validate(self, document):
        if not document.text.isdigit():
            raise ValidationError(message="Bitte eine Zahl eingeben.")
        port = int(document.text)
        if not (0 < port < 65536):
            raise ValidationError(message="Port muss 1–65535 sein.")

# Felder definieren
username_input = TextArea(prompt="Benutzername: ",
validator=UsernameValidator())
port_input = TextArea(prompt="Port: ", validator=PortValidator())

def speichern():
    print("✅ Benutzer:", username_input.text)
    print("✅ Port:", port_input.text)
    app.exit()

save_btn = Button("Speichern", handler=speichern)

# Layout und App
form = HSplit([username_input, port_input, save_btn])
app = Application(layout=Layout(form), full_screen=False)
app.run()
```

✅ Zusammenfassung

🔑 Wichtigste Punkte

- Verwende `Validator` in `prompt_toolkit`, um **Eingaben live zu prüfen**.
- Nutze Stringmethoden wie `.strip()`, `.isdigit()`, `.endswith()` etc.
- Kombiniere **Live-Prüfung** mit **abschließender Logikprüfung**, wenn nötig.
- Zeige **klare Fehlermeldungen**.
- Wiederhole Eingaben nur bei Fehlern, nicht pauschal.

Wenn du willst, kann ich dir als Nächstes zeigen:

- 🧠 Wie man Felder wiederholt, bis sie gültig sind (z. B. in einer Funktion)
- 💾 Wie du validierte Daten speicherst (z. B. JSON, INI, ENV)
- 🔄 Dynamische Validierung (je nach anderem Feld)

Sag einfach Bescheid!