Python-String-Methoden & Funktionen für Fortgeschrittene (Teil 2)

1. removeprefix() und removesuffix()

Diese Methoden entfernen gezielt einen Anfangs- oder Endtext aus einem String.

Beispiel:

text = "Max(9)"

wert = text.removeprefix("Max(").removesuffix(")")

ergibt: "9"

Nutzen:

- ideal, um strukturierte Texte zu zerlegen (z.B. "Modus(5)")

- sehr einfach zu lesen

Hinweis:

- nur ab Python 3.9 verfügbar!

2. zfill()

Füllt einen String von links mit Nullen auf eine gewünschte Länge.

Beispiel:

"7".zfill(3) -> "007"

Nutzen:

- praktisch bei Dateinamen, IDs, Sortierung

Python-String-Methoden & Funktionen für Fortgeschrittene (Teil 2)

3. partition()

Teilt den String in drei Teile: vor, das Trennzeichen, nach.

Beispiel:

"text:123".partition(":") -> ("text", ":", "123")

Nutzen:

- besser als split() für genau ein Trennzeichen

4. casefold()

Wie lower(), aber noch strenger. Gut für Sprachvergleiche.

Beispiel:

"Straße".casefold() -> "strasse"

Nutzen:

- Sprachunabhängige Kleinbuchstaben

5. lstrip(), rstrip()

Entfernt Leerzeichen (oder Zeichen) nur von links bzw. rechts.

Beispiel:

" text ".lstrip() -> "text "

" text ".rstrip() -> " text"

Python-String-Methoden & Funktionen für Fortgeschrittene (Teil 2)

Nutzen:

- für gezieltes Aufräumen

6. isalpha(), isalnum(), isspace()

Prüfen auf nur Buchstaben, Buchstaben+Zahlen oder nur Leerzeichen.

Beispiel:

"abc".isalpha() -> True

"abc123".isalnum() -> True

" ".isspace() -> True

Nutzen:

- für Validierung von Benutzereingaben

7. format() mit Platzhaltern

Erlaubt elegantes Einfügen von Variablen in Texte.

Beispiel:

"Name: {}, Alter: {}".format("Max", 12) -> "Name: Max, Alter: 12"

Nutzen:

- sauberer als viele + Zeichen

Python-String-Methoden & Funktionen für Fortgeschrittene (Teil 2)

8. startswith() und endswith()

Prüfen, ob ein String mit einem bestimmten Text beginnt oder endet.

Beispiel:

"bild.jpg".endswith(".jpg") -> True

Nutzen:

- wichtig z.B. bei Dateifiltern