REGEX - Regular Expressions

Was sind Regular Expressions (Regex oder RE)?

Regular Expressions sind Muster, die verwendet werden, um im Text Zeichenfolgen zu suchen, zu extrahieren, zu ersetzen oder zu validieren. Sie sind ein wichtiges Werkzeug, um mit Textdaten in Programmierungen und Scripting-Sprachen wie Python zu arbeiten. In Python kann Regex über das `re`-Modul genutzt werden, das eine Schnittstelle zur Regex-Engine bietet.

Wichtige Funktionen im Python `re`-Modul

Funktion	**Beschreibung**
re.compile(*pattern*, *flags=0*)	erstellt aus Regex -Pattern ein Regex -Objekt, das wiederholt aufgerufen werden kann.
re.match(*pattern*, *string*, *flags=0*)	Überprüft, ob der Regex zu Beginn des Strings passt
re.search(*pattern*, *string*, *flags=0*)	Sucht den gesamten String nach einem Muster ab
re.findall(*pattern*, *string*, *flags=0*)	Gibt alle Vorkommen eines Musters im String als Liste zurück
re.sub(*pattern*, *repl*, *string*, *count=0*, *flags=0*)	Ersetzt Teile eines Strings, die einem Muster entsprechen
re.split(*pattern*, *string*, *maxsplit=0*, *flags=0*)	Teilt einen String an allen Stellen, die einem Muster entsprechen

Grundlegende Regex-Muster

Einfache Zeichen (Literal Characters)

Die meisten Zeichen entsprechen sich selbst. Beispiel: Der Ausdruck `test` matcht genau die Zeichenfolge "test". **Achtung**: Regex ist standardmäßig **case-sensitive**, d.h. `test` passt nur zu `test`, nicht zu `Test`.

Metacharacters (Sonderzeichen)

Metacharacters sind Zeichen mit spezieller Bedeutung in Regex. Einige der wichtigsten sind:

Metacharacter	**Bedeutung**	**Beispiel**
`.` (Punkt)	Passt zu jedem Zeichen (außer Zeilenumbruch)	`a.c` passt zu `abc`, `aXc`, aber nicht zu `ac`
`^` (Zirkumflex)	Passt nur am **Anfang** eines Strings/Zeile	`^Hello` passt zu `Hello World`, aber nicht zu `Well Hello`

`$` (Dollar)	Passt nur am **Ende** eines Strings/Zeile	`end$` passt zu `the end`, aber nicht zu `ending soon`
`*` (Sternchen)	0 oder mehr Wiederholungen des vorherigen Zeichens	`ba*` passt zu `b`, `ba`, `baa`, `baaa`
`+` (Plus)	1 oder mehr Wiederholungen des vorherigen Zeichens	`ba+` passt zu `ba`, `baa`, `baaa`, aber nicht zu `b`
`?` (Fragezeichen)	0 oder 1 Vorkommen des vorherigen Zeichens	`ba?` passt zu `b` und `ba`, aber nicht zu `baa`
`{n}`	Genau **n** Wiederholungen des vorherigen Zeichens	`ba{2}` passt zu `baa`, aber nicht zu `ba` oder `baaa`
`{n,m}`	Zwischen **n** und **m** Wiederholungen des vorherigen Zeichens	`ba{1,3}` passt zu `ba`, `baa`, `baaa`
`[]` (eckige Klammern)	Definiert eine **Zeichenklasse** (Menge von Zeichen)	`[aeiou]` passt zu einem beliebigen Vokal, `[A-Za-z]` zu einem beliebigen Buchstaben
`|` (Pipe)	**ODER**-Verknüpfung für Auswahl zwischen Optionen	cat|dog passt zu cat oder dog.
`()` (runde Klammern)	Gruppiert Ausdrücke und ermöglicht wiederholte Anwendung von Operatoren	`(ab)+` passt zu `ab`, `abab`, `ababab`
`\` (Backslash)	Maskiert Sonderzeichen oder definiert **spezielle Sequenzen**	`\.` passt zu einem Punkt, `\d` zu einer Zahl (0-9)