
==========

Doku --> http://docs.python.org/3/library/stdtypes.html#str
         http://docs.python.org/3/library/stdtypes.html#string-methods

Strings in Python sind seit Python 3.0 Sequenzen von Unicode-Einzelzeichen codiert im Format UTF-8, d.h. jedes Zeichen belegt 1-4 Bytes. In Python 2.X wurden Strings noch durch Einzelbyte-Zeichen dargestellt (z.B. ISO-8859).

Die String-Quotierungen "..." und '...' haben die exakt gleiche Bedeutung, Python selbst stellt per repr() erzeugte Strings in der Form '...' dar.

```
  "hallo welt"    # Unicode-String (in P3 automatisch)
  u"hallo welt"   # Unicode-String (in P2 notwendig, in P3 Standard)
  U"hallo welt"   # Unicode-String (in P2 notwendig, in P3 Standard)
  ""              # Leerer String
```

Folgende LESENDEN Sequenz-Operationen werden von Strings unterstützt (R, S, T, X sind Strings, I, N sind Ganzzahlen, C ist ein Zeichen, D ist ein Dictionary, L ist eine Liste von Strings, ENC ist ein Encoding):

ACHTUNG: String S bleibt UNVERÄNDERT, neuer String kommt als Ergebnis zurück.
         (ergibt sich aus der im-mutable Eigenschaft von Strings).

Operation	Bedeutung
for E in S: ...	Alle Zeichen E von S nach Index geordnet durchlaufen
T in S	Substring T in S enthalten
T not in S	Substring T nicht in S enthalten
S + T	Verkettung von S und T
S * N N * S	String S N-mal vervielfachen
S[I]	Zeichen mit Index I (Start bei 0)
S[I:J]	Substring von Index I bis J-1 (mit Schrittweite 1)
S[I:J:K]	Substring von Index I bis J-1 mit Schrittweite K
S == T	Strings S und T gleich?
S != T	Strings S und T ungleich?
S < T	S alphabetisch kleiner T (lexikografisch!) Unicode!
S <= T	S alphabetisch kleiner gleich T (lexikografisch!) Unicode!
S > T	S alphabetisch größer T (lexikografisch!) Unicode!
S >= T	S alphabetisch größer gleich T (lexikografisch!) Unicode!
len(S)	Länge von S (Anzahl Zeichen)
min(S)	Kleinstes Zeichen in S (gemäß Vergleich "<")
max(S)	Größtes Zeichen in S (gemäß Vergleich "<")
reversed(S)	Zeichen in umgekehrter Reihenfolge (Iterator)
sorted(S,key, reverse)	Sortierte Liste der Zeichen (gemäß Vergleich "<") (key=MAPFUNC, reverse=True --> absteigend)

Folgende LESENDEN String-Funktionen sind für Strings verfügbar:

ACHTUNG: String S bleibt UNVERÄNDERT, neuer String kommt als Ergebnis zurück
         (ergibt sich aus der im-mutable Eigenschaft von Strings).

Funktion		Bedeutung
S.lower()	U	Kleinschreibung
S.upper()	U	Großschreibung
S.swapcase()	U	Groß/Kleinschreibung vertauschen
S.capitalize()	U	1. Buchstaben groß schreiben, Rest klein
S.title()	U	Alle Worte beginnen mit Großbuchstaben, Rest klein
S.casefold()	U	Aggressiveres lower() (z.B. "ß" --> "ss") 3.3

S.startswith(T[,I[,N]]		Beginnt mit T (ab Position I, Länge N)	
S.endswith(T[,I[,N]]		Endet mit T (ab Position I, Länge N)	
S.count(T[,I[,N]])		Anzahl Vorkommen von T (ab Position I, Länge N)	
S.find(T[,I[,N]])		Von links Treffer-Index von T finden (-1 falls nein)	
S.index(T[,I[,N]])		Wie S.find(), ValueError statt -1 falls kein Treffer	
S.rfind(T[,I[,N]])		Von rechts Treffer-Index von T finden (-1 falls nein)	
S.rindex(T[,I[,N]])		Wie S.rfind(), ValueError statt -1 falls kein Treffer	
S.isspace()	U	Besteht nur aus Leerraum (mind. 1)	
S.isascii()		Besteht nur aus ASCII-Zeichen 0-127 (oder leer)	
S.isidentifier()		Gültiger Python-Bezeichner (--> keyword.iskeyword())	
S.isprintable()	U	Alle Zeichen in repr() ausdruckbar (oder leer)	
S.isalpha()	U	Besteht nur aus Buchstaben (mind. 1)	
S.isdecimal()	U	Besteht nur aus Ziffern (Dezimalziffern, mind. 1)	1
S.isdigit()	U	Besteht nur aus Ziffern (Fraction/Superscript, m. 1)	2
S.isnumeric()	U	Besteht nur aus Ziffern (römische, mind. 1)	3
S.isalnum()	U	Besteht nur aus Buchstaben oder Ziffern (mind. 1) (isalpha + isdecimal + isdigit + isnumeric)	
S.islower()	U	Besteht nur aus Kleinbuchstaben (mind. 1)	
S.isupper()	U	Besteht nur aus Großbuchstaben (mind. 1)	
S.istitle()	U	Alle Worte beginnen mit Großbuchstaben, Rest klein	
S.removeprefix(T)		Falls S mit T beginnt, T entfernen 3.9	
S.removesuffix(T)		Falls S mit T endet, T entfernen 3.9	
S.lstrip([T])		Beliebig viel Leerraum/Zeichen T links entfernen	
S.rstrip([T])		Beliebig viel Leerraum/Zeichen T rechts entfernen	
S.strip([T])		Beliebig viel Leerraum/Zeichen T links+rechts entf.	
S.center(N[,C])		Zentriert in Breite N mit Füllz. C (STD: Leerz.)	
S.ljust(N[,C])		Linksbündig in Breite N mit Füllz. C (STD: Leerz.)	
S.rjust(N[,C])		Rechtsbündig in Breite N mit Füllz. C (STD: Leerz.)	
S.zfill(N)		Links in Breite N mit "0" auffüllen	
S.join(L)		Verketten der Elemente in L getrennt durch S	
S.split(T[,MAX])		Zerlegen von links anhand T (alle/MAX)	
S.rsplit(T[,MAX])		Zerlegen von rechts anhand T (alle/MAX)	
S.splitlines(keepends=False)		Zerlegen anhand Zeilentrenner \n \r \r\n \f \v ... (Zeilenenden entfernen oder erhalten)	
S.partition(T)		Zerlegen in (Head, T, Tail) am ersten T von links	
S.rpartition(T)		Zerlegen in (Head, T, Tail) am ersten T von rechts	
S.replace(T,R[,MAX])		T durch R ersetzen (alle oder Anzahl MAX)	
S.translate(D)		Zeichen-Key in D durch Value ersetzen oder löschen	
S.make_trans(D)		(Übersetzungs-Tabelle für translate() erstellen)	
S.make_trans(S,T[,X])		(Übersetzungs-Tabelle für translate() erstellen)	
S.expandtabs(N)		Tabulatoren (Breite N) durch Leerzeichen ersetzen	
S.decode(ENC,ERR)		Decodieren gemäß Codierung ENC	
S.encode(ENC,ERR)		Codieren gemäß Codierung ENC	
F"S"		{var}-Platzhalter in S ersetzen durch Wert von var	
S.format(*args,**kwa)		{}-Platzhalter in S ersetzen durch Werte in (...)	
S.format_map(...)		Analog, aber **kwargs wird nicht in dict kopiert	
S % (...)		%-Platzhalter in S ersetzen durch Werte in (...)	

* HINWEIS: Kürzel "U" --> durch den Unicode-Standard definiert.

* func(T[,I[,N]]) steht für: von Index I (Std: 0) bis VOR Index N (Std: len(T))
  (,I,N oder ,N können auch fehlen).

* MAX steht für Anzahl Stücke MAXIMAL, sonst ALLE Stücke erzeugen/ersetzen.

* ERR kann "strict", "ignore", "replace" und "xmlcharrefreplace" sein. TODO

* Unterschied --> Mostly about Unicode classification
  isdecimal() <= isdigit() <= isnumeric()

```
    True           True           True     # 10-System Ziffern:      "038"  "o३t"  "0 3
8"
```

```
    False          True          True     # Fractions/Superscripts: "⁰³⁸", "0.3.8.",
"⓪③⑧""
    False          False         True     # Römische Zahlen:        "⅗⅛⅘", "ⅢⅧ",
"⑩⑬㊿", "壹貳參"
    False          False         False    # Alles andere            "abc", "38.0",
"-38"
```

* split() und rsplit() ohne Argument zerlegen den String an WHITESPACE
  (führende und abschließende Whitespaces werden dabei ignoriert).

* Leerraum (WHITESPACE) ist eine beliebig lange Folge folgender Zeichen:

\t	9	Horizontaler Tabulator	Zur nächsten Tabulator-Position
\n	10	Newline	Zu nächster Zeile
\v	11	Vertikaler Tabulator	
\f	12	Form Feed	Seitenvorschub (Drucker)
\r	13	Carriage Return	Zum Zeilenanfang
" "	32	Space	Leerzeichen