

Objekttypen	
Integer (int)	Ganzzahl
Float	Gleitkommazahl
String (str)	Zeichenkette
Bool	TRUE or FALSE
Liste	[a,b,c,d]
Tupel	'a','b','c'
Menge (set)	{'Java','Perl','Python'}
Dictionary	{'Birnen':2, 'Bier':1}

Arithmetische Operatoren	
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulus (Rest nach Division)
**	Exponent

Verketten von Zeichenkette	
+	Verkettung beider Zeichenketten
,	Verkettung gemischter Objekte

Vergleichsoperatoren	
A == b	Prüft, ob Werte A und b gleich sind
A != b	Prüft, ob Werte a und b ungleich sind
A > b	Prüft, ob Wert A größer als Wert b ist
A < b	Prüft, ob Wert A kleiner als Wert b ist
A >= b	Prüft, ob Wert a größer oder gleich Wert b ist
A <= b	Prüft, ob Wert a kleiner oder gleich Wert b ist
A is b	Prüft, ob a und b auf das selbe Objekt verweisen
Aussage wahr	TRUE

Logische Verknüpfungen	
and	und
or	oder
xor	entweder-oder
not	nicht

Slicing	
[a:b:c]	Zugriff auf bestimmten Teil einer Zeichenkette
a	untere Grenze
b	obere Grenze
c	Schrittweite
c nur angeben, wenn bestimmte Schritte gegangen werden sollen	

Listen	
Element = list[2]	Element mit Index 2 aus list
list.append(-Objekt)	fügt neues Objekt in Liste hinzu
del(list[Indexnr.])	löscht Objekt an [Indexnr.]
len(list)	Länge der Liste
list.reverse()	Liste wird umgedreht
list.sort()	Liste wird sortiert
"Zeichen".join(list)	Listenelemente als Zeichenkette verbunden über Zeichen-Teilstring

Tupel	
Listen die nicht veränderbar sind!	

Dictionaries	
dic =	(Key: Value, Key: Value)
element = dic[key]	Ausgabe Value vom Key
dic[key] = string	Aktualisiert dic am Position Key
dic[key] = string	Fügt neue Werte ein
dic.update(dic2)	Verbindet Dictionaries miteinander
len(dic)	Anzahl der vorhandenen Key im Dic.
dic.clear()	Löscht Inhalt eines Dictionary
list(dic.keys())	Liste mit allen Schlüsseln aus dem Dic.
list(dic.values())	Liste mit allen Values aus den Dic.
list(dic.items())	Liste aller Key-Value-Paare aus den Dic.
var = dic[key]	Zugriff auf Wert von key in dic



### Mengen

set(list or string)	Umwandlung von Listen oder Strings in Mengen
	set-Objekt --> einmalige & unveränderbare Elemente aus ursprünglichen Liste
menge.add(TEXT)	Fügt Element zur Menge hinzu
menge.clear()	Leert die Menge
menge.copy()	flache Kopie einer Menge
len(menge)	Anzahl der Elemente in einer Menge
menge.difference(string)	Ausgabe der Differenz einer Menge
menge.difference_update(string)	Aktualisiert ursprüngliche Menge direkt
menge.discard(element)	Entfernt Element aus einer Menge
menge.union(menge1, menge2)	Bestimmung der Vereinigungsmenge zweier Mengen
menge.intersection(menge1, menge2)	Bestimmung Schnittmenge zweier Mengen
menge.isdisjoint(menge1, menge2)	Ermittelt ob 2 Mengen disjunkt sind (Schnittmenge = leer)
menge.issubset(menge1, menge2)	Falls Menge eine Teilmenge einer anderen Menge ist
menge.issuperset(menge1, menge2)	Falls Menge, Obermenge einer anderen Menge ist
menge.pop()	Beliebiges Element einer nicht-leeren Menge zurückgeliefert und entfernt

### Zeichenketten auftrennen

split(was eingesetzt)	Zeichenkette an einem Substring aufgeteilt
-----------------------	--

### Zufallszahlen

random.random()	Float r mit $0 \leq r < 1$
random.randrange(start, stop, step)	Integer r mit $start \leq r < stop$
random.randint(start, stop)	Integer r mit $start \leq r \leq stop$
random.sample(Anzahl Zeichen)	Liste mit 4 zufällig gewählten Zeichen
random.uniform(start, stop)	Float r mit $start \leq r \leq stop$
import random	

### Ausgabe in Groß- und Kleinbuchstaben

string.upper()	Alles in Großbuchstaben
string.lower()	Alles in Kleinbuchstaben
string.capitalize()	1. Buchstabe groß geschrieben

### Länge einer Zeichenkette

len(string)	Längen einer Zeichenkette als int-Objekt
-------------	--

### Suche nach einem Substring

string.find(STRING, START, END)	Index des 1. Auftretens STRING in string (von links ausgehend)
string.rfind(STRING, START, END)	Index vom letzten Auftreten von STRING in string (von rechts ausgehend)
Index = -1 --> Falls STRING nicht in string	

### Teilstring-Anzahl

string.count(STRING, START, END)	Anzahl von STRING in string
Index END nicht enthalten	

### Ersetzung in einer Zeichenkette

string.replace(REP, NEW, MAX)	Teilstring REP durch Teilstring NEW ersetzt
-------------------------------	---



### Übersetzen von Zeichen in einer Tabelle

<code>string.maketrans(IN,OUT)</code>	Erstellt Übersetzungstabelle
<code>translate(string)</code>	Übersetzt erstellte Tabelle

### Abfrage String bzgl. bestimmter Kriterien

<code>isalnum(string)</code>	mind. 1 Zeichen, alle alphanummerisch
<code>isalpha(string)</code>	mind. 1 Zeichen, alle alphabetisch
<code>isdigit(string)</code>	Enthält nur Zahlen
<code>isnumeric(string)</code>	Unicode-String enthält nur numerische Zahlen
<code>isdecimal(string)</code>	Unicode-String enthält nur Dezimalzahl
Ausgabe: TRUE or FALSE	

### Aussagenlogik (Regeln log. Verknüpfungen)

Assoziativgesetze	$(a \text{ and } b) \text{ and } c = a \text{ and } (b \text{ and } c)$ $(a \text{ or } b) \text{ or } c = a \text{ or } (b \text{ or } c)$
Kommutativgesetze:	$A \text{ and } b = b \text{ and } a$ $A \text{ or } b = b \text{ or } a$
Distributivgesetze:	$A \text{ or } (b \text{ and } c) = (a \text{ or } b) \text{ and } (a \text{ or } c)$ $A \text{ and } (b \text{ or } c) = (a \text{ and } b) \text{ or } (a \text{ and } c)$
Absorptionsgesetze:	$A \text{ or } (a \text{ and } b) = a$ $A \text{ and } (a \text{ or } b) = a$

### Reguläre Ausdrücke

<code>re.search(MUSTER,S-STRING)</code>	Sucht MUSTER im STRING
<code>var.group()</code>	Zugriff auf einzelne Teile bei einem Match
<code>re.finditer(MUSTER,STRING)</code>	Sucht MUSTER im STRING, falls es mehrmals vorkommt
<code>re.sub(MUSTER,ERSETZUNG,STRING)</code>	Ersetzt MUSTER durch ERSETZUNG im STRING
<code>import re</code>	

### Metazeichen

<code>[abcdef]</code>	Ein Zeichen, aus angegebener Menge
<code>[^abcdef]</code>	Ein Zeichen, nicht aus angegebener Menge
<code>aa   bb</code>	Es muss aa oder bb vorkommen
<code>\$</code>	Ende der Zeichenkette
<code>^</code>	Anfang der Zeichenkette

### Zeichenklassen

<code>[0-9]</code> (Zeichenmenge)	<code>\d</code> (Kurzform) <code>\D</code> (Gegenteil) --> Ziffer
<code>[a-zA-Z0-9]</code>	<code>\w</code> <code>\W</code> --> Wortzeichen
<code>[\t\n\r\f]</code>	<code>\s</code> <code>\S</code> --> Whitespace

### Quantoren

<code>*</code>	Beliebig oft, darf auch fehlen
<code>+</code>	Mindestens einmal
<code>?</code>	Einmal oder keinmal
<code>{ZAHL}</code>	Muss genau ZAHL-mal vorkommen
<code>{ZAHL,}</code>	Mindestens ZAHL-mal
<code>{,ZAHL}</code>	Maximal ZAHL-mal
<code>{ZAHL1,ZAHL2}</code>	Min. Zahl1-mal, Max. ZAHL2-mal

### Modifizierer

<code>re.DEBUG</code>	DEBUG-Info ausgeben
<code>re.I</code>	Groß-& Kleinschreibung ignorieren
<code>re.M</code>	Multi-Zeilen-Treffer
<code>re.S</code>	<code>.</code> trifft auch Zeilenumbruch, ansonsten nicht

bei `re.search (MUSTER,STRING,MOD)` als MOD eingeben

### Dateien einlesen

```
with open ("datei.txt", "r") as fin:
--> for zeile in datei.txt:
--> ---> zeile = zeile.rstrip()
```

Dateien zeilenweise einlesen und Zeilenumbrüche entfernen

### Dateien schreiben

```
with open ("datei.txt", "w") as fout:
--> print ("Hallo Welt", file = fout)
-->Hallo Welt
```

