

Variablen und Strings	Datentypen	Schleifen	Listen
Text ausgeben <pre>print(" Hello world! ") >> Hello world!</pre>	Numerische Typen int, long für ganze Zahlen (long ist für besonders große Zahlen) float für Kommazahlen	Eine einfache While-Schleife <pre>current_value = 1 while current_value <= 5: print(current_value) current_value += 1</pre>	Liste erstellen Strings gehören in einfache Anführungszeichen <pre>liste = [elem0, elem1, elem2]</pre>
Text mit einer Variablen ausgeben <pre>msg = " Hello world! " print(msg) >> Hello world!</pre>	Boolsche Werte bool Wahrheitswerte (true, false)	Den Benutzer entscheiden lassen, wann die Schleife beendet werden soll <pre>msg = '' while msg != 'quit': msg = input("Wie ist deine Nachricht? ") print(msg) << Hallo Welt! >> Hallo Welt!</pre>	erstes und letztes Element erhalten <pre>erstesElem = liste[0] letztesElem = liste[-1]</pre>
Verkettung (Strings kombinieren) <pre>first_name = 'albert' last_name = 'einstein' full_name = first_name + ' ' + last_name print(full_name) >> albert einstein</pre>	Arithmetische Operationen x+y Addition x-y Subtraktion x*y Multiplikation x/y Division x%y Rest bei ganzzahliger Division x**y Potenz	Eine while-Schleife wiederholt einen Codeblock, solange wie eine bestimmte Bedingung erfüllt ist. 	Elemente zu einer Liste hinzufügen <pre>liste = [] liste.append(elem1) liste.append(elem2) liste.append(elem3)</pre>
Variablen werden zum Speichern von Zahlenwerten, Zeichen oder Strings verwendet. Ein String ist eine Reihe von Zeichen, die von einfachen oder doppelten Anführungszeichen umgeben sind.		Tupel Tupel erstellen <pre>dimensionen = (1920, 1080)</pre>	Listen Durchgehen einer Liste <pre>for elem in liste: print(elem)</pre> je nachdem, was elem für einen Typ hat, wird eine Zahl, ein Zeichen oder ein String ausgegeben <pre>>>4 >>a >>Hallo >>3,46</pre>
		Tupel ähneln Listen, aber die Elemente in einem Tupel können nicht geändert werden.	Numerische Listen erstellen <pre>quadrate = [] for x in range(1, 11): quadrate.append(x*x)</pre>



Listen	Benutzereingabe	Funktionen (cont)	Verzweigung
Abkürzung <code>quadrates = [x**2 for x in range(1, 10)]</code>	Aufforderung zur Zahleneingabe <code><< 16</code> <code>alter = int(input("Wie alt bist du?"))</code> Hier findet eine Typumformung statt, da jede Eingabe zunächst als String gespeichert wird.	Erstellt eine einfach belegte Pizza <code>def make_pizza(size, toppings):</code> <code> print("Du bekommst eine Pizza mit " + str(size) + " Zoll und " + str(len(toppings)) + " Toppings.")</code> <code> make_pizza(16, ['pepperoni', 'mushrooms', 'olives', 'anchovies'])</code> Du bekommst eine Bacon Pizza. Du bekommst eine Pepperoni Pizza.	Vergleichsbedingungen gleich <code>x == 42</code> ungleich <code>x != 42</code> größer als <code>x > 42</code> größer gleich <code>x >= 42</code> kleiner als <code>x < 42</code> kleiner gleich <code>x <= 42</code>
Listen teilen <code>finalisten = ['sam', 'bob', 'ada', 'bea']</code> <code>ersten Beiden = finalisten[:2]</code>	Listen kopieren <code>kopieVonListe = liste[:]</code> Eine Liste speichert eine Reihe von Elementen (Zahlen, Zeichen, Strings, Objekte, ...) in einer bestimmten Reihenfolge. Du greifst auf Elemente über einen Index oder innerhalb einer Schleife zu.	Rückgabe eines Wertes <code>def summiere(x, y):</code> <code> return x + y</code> <code>sum = summiere(3, 5)</code> <code>print(sum)</code> <code>>> 8</code>	Bedingungen mit Listen <code>elem1 in liste</code> <code>elem3 not in liste</code> Boolesche Werte zuweisen <code>aktiv = True</code> <code>pausiert = False</code>
Benutzereingabe Aufforderung zur Eingabe <code>name = input("Wie ist dein Name? ")</code> <code>print("Hallo, " + name + " !")</code> <code><< Steve</code> Wenn auf dem Bildschirm die Frage "Wie ist dein Name?" steht, ist man aufgefordert eine Eingabe zu tätigen. Diese wird in der Variablen name gespeichert. <code>>> Hallo, Steve!</code>	Funktionen Eine einfache Funktion <code>def begruesse_benutzer():</code> <code> print("Hallo!")</code> <code>begruesse_benutzer()</code> <code>>> Hallo!</code> Zeigt eine Begrüssung. Argumente übergeben <code>def greet_user(username):</code> <code> print("Hallo, " + username + " !")</code> <code>greet_user('SummerBYTE')</code> <code>>> Hallo, SummerBYTE !</code> Standardwerte für Parameter	Funktionen sind gekennzeichnete Codeblöcke, die für eine bestimmte Aufgabe konzipiert sind. Informationen, die an eine Funktion übergeben werden, werden als Argument bezeichnet, und Informationen, die von einer Funktion empfangen werden, werden als Parameter bezeichnet.	Einfache if-Verzweigung <code>if alter >= 18:</code> <code> print("Du darfst wählen")</code> <code>>> Du darfst wählen!</code> if-elif-else-Verzweigungen nur ein Codeblock wird ausgeführt (Reihenfolge beachten!)



Verzweigung (cont)	Klassen	Wörterbücher (cont)	Wörterbücher (cont)
<pre>if alter < 4: ticket Preis = 0 elif alter < 18: ticket Preis = 10 else: ticket Preis = 15</pre>	Vererbung <pre>class SARDog(Dog): Stelle einen Rettungshund dar. def __init__(self, name): Initialisiert die Instanz des Rettungshundes. super().__init__(name) print(name + ' liebt die 17') def search(self): Simuliert die Suche. print(self.name + " sucht.") my_dog = SARDog('Fang') print(my_dog.name + " ist ein Rettungshund.") my_dog.search() >> Fang sucht.</pre>	neues Schlüssel-Wert-Paar einfügen <pre>alien['x_position'] = 0</pre> Durchlaufen aller Schlüssel-Wert-Paare <pre>for name, number in Lieblingszahl.items(): print(str(number) + ' ist ein Liebling') >> 17 ist ein Liebling >> 4 ist ein Liebling</pre> Durchlaufen aller Schlüssel <pre>for name in Lieblingszahl.keys(): print(name + ' liebt eine Zahl') >> erik liebt eine Zahl >> julia liebt eine Zahl</pre>	Wörterbücher speichern Verbindungen zwischen Informationen. Jedes Element in einem Wörterbuch ist ein Schlüsselwertpaar. Durchlaufen aller Werte
Verzweigungen werden verwendet, um bestimmte Bedingungen zu testen und entsprechend zu reagieren.	Klassen Erstellen einer Klasse "Hund" <pre>class Dog(): Stellt einen Hund dar. def __init__(self, name): Initialisiert eine Instanz der Klasse Hund. self.name = name def sit(self): Simuliert sitzen. print(self.name + " sitzt.") my_dog = Dog('Fluffy') print(my_dog.name + " ist ein braver Junge.") >> Fluffy ist ein braver Junge. my_dog.sit() >> Fluffy sitzt.</pre>	Wörterbücher Einfaches Wörterbuch <pre>alien = {'Farbe': 'grün', 'Punkte': 5}</pre> Zugriff auf einen Wert <pre>print("Die Farbe des Aliens ist " + alien['Farbe']) >> Die Farbe des Aliens ist grün</pre>	



By ProjektSB19

Published 3rd June, 2019.

Last updated 17th June, 2019.

Page 3 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>