

Python 3.0 Spickzettel	
Giuseppe Accaputo g@accaputo	
Kurs: Grundlagen der Programmierung für Nicht-Informatiker, HS18	
Variablen	
variablen_name = <wert>	
typ_der_variable = type(variable)	
Datentypen	
Integer (Int)	-25, 2, 14, 100, -20
Float	2.4123, -1.1312, 4.14123
String	'Hallo 1', 'Hallo 2'
Boolean	True, False
List	[1, 1.2423, 'zwei']
Tupel	(1, 2, 3, 'vier')
Dictionary	{'key1': wert1, 'key2': wert2}
Eingabe	
eingabe = input('Bitte Name eingeben:')	
eingabe = int (input('Bitte Zahl eingeben:'))	
eingabe = float (input('Bitte Zahl eingeben:'))	
Ausgabe	
print('Wert der Variable:', variable	
print('Typ der Variable', type(variable))	
print('Keine neue Zeile am Ende dieses Texts',)	
Strings – Teil 1	
ein_string = 'Hallo, Welt!'	Variable vom Typ String definieren
ein_string[1:5]	Segment von Index 1 bis und mit Index 4 selektieren
ein_string[-1]	Auf das letzte Zeichen zugreifen
ein_string.lower()	In Kleinbuchstaben umwandeln
ein_string.upper()	In Grossbuchstaben umwandeln
ein_string.replace(alt, neu)	alt durch neu in ein_string ersetzen
string1 == string2	Ist string1 gleich string2 ?

zahl_als_string = str(1.234)	Typumwandlung zu String
hallo = 'Hallo, ' welt = 'Welt!' hallo_welt = hallo + welt	+ Operator: Zwei Strings verknüpfen
area = 'Area ' area_zahl = 51 area_51 = area + str(area_zahl)	+ Operator und str() : String und Zahl verknüpfen
Zahlen	
int(variable)	Typumwandlung zu Int
float(variable)	Typumwandlung zu Float
Mathematische Operatoren	
x ** y	Exponent, x^y
x % y	Modulus; berechnet den Rest der Division x geteilt durch y
x / y	Division
x * y	Multiplikation
x - y	Subtraktion
x + y	Addition
x op y, und x, y sind beide Ints	Resultat der Operation ist ein Int (op kann +, -, *, / sein)
x op y, und x oder y ist Float	Resultat der Operation ist ein Float (op kann +, -, *, / sein)
Funktionen	
def hallo(): print('Hallo!') hallo() # Aufruf	Eine Funktion ohne Rückgabewert
def hallo(name): print('Hallo, ', name) hallo('Klasse') # Aufruf	Eine Funktion mit einem Argument
def summe(x, y, z): return x + y + z print(summe(1,2,3)) # Aufruf	Eine Funktion mit einem Rückgabewert
import math math.sqrt(zahl) # Wurzel math.log(zahl) # Logarithmus	Mathematische Funktionen verwenden

Bedingte Anweisungen	
<pre>if alter >= 18: print('Du darfst abstimmen!')</pre>	Eine einfache Abfrage
<pre>if zahl < 2: print('Zahl ist kleiner als 2') elif zahl >= 2 and zahl <= 5: print('Zahl liegt in [2,5]') else: print('Zahl ist grösser als 5')</pre>	Eine Abfrage mit mehreren Verzweigungen (if-elif-else)
Vergleichsoperatoren	
<code>x == y</code>	Ist x gleich y?
<code>x != y</code>	Ist x ungleich y?
<code>x > y</code>	Ist x grösser als y?
<code>x < y</code>	Ist x kleiner als y?
<code>x >= y</code>	Ist x grösser oder gleich y?
<code>x <= y</code>	Ist x kleiner oder gleich y?
Logische Operatoren	
<code>a and b</code>	Nur True, wenn a und b True sind
<code>a or b</code>	True, wenn a oder b, oder beide True sind
<code>not a</code>	Negiert a
Die while Schleife	
<pre>countdown = 10 while(countdown > 0): print(countdown) countdown = countdown - 1</pre>	Solange <code>countdown > 0</code> , gib <code>countdown</code> aus und dekrementiere den Wert von <code>countdown</code> um 1
Datenstrukturen [S: String, L: Liste, T: Tupel, D: Dictionary]	
<code>s = 'ein String'</code>	String (unveränderbar)
<code>l = [1,2,3, 'vier']</code>	Liste (veränderbar)

<code>t = (1, 'zwei', 'drei', 4)</code>	Tupel (unveränderbar)
<code>d = {'key1': 'wert1', 'key2': 'wert2'}</code>	Dictionary (veränderbar)
<code>erstes_element = ds[0]</code> <code>zweites_element = ds[1]</code>	Auf das i-te Element zugreifen (erstes Element befindet sich bei Index 0) [S, L, T, D]
<code>ds[i] = neuer_wert</code>	i-te Element ändern [L, D (i ist Schlüssel)]
<code>len(ds)</code>	Gibt die Anzahl Elemente zurück [S, L, T, D]
<code>letztes_element = ds[-1]</code> <code>zweitletztes_element = ds[-2]</code>	Rückwärts auf Elemente zugreifen [S, L, T]
<code>for k in ds:</code> <code> print(k)</code>	Alle Elemente traversieren [S, L, T, D (k ist Schlüssel)]
<code>for (key, wert) in ds:</code> <code> print(key, ':', wert)</code>	Alle Wert-Schlüssel Paare in einem Dictionary traversieren [D]
<code>ds[0:6]</code>	Segment von Index 0 bis und mit Index 5 selektieren [S, L, T]
<code>ds[3:]</code>	Segment von Index 3 bis und mit Ende der Datenstruktur selektieren [S, L, T]
<code>k in ds</code>	Existiert Element k in Datenstruktur [S, L, T, D (k ist Schlüssel)]
<code>ds1 == ds2</code>	Enthalten die beiden Datenstrukturen dieselben Elemente? [S, L, T, D]
<code>ds1 + ds2</code>	Zwei Datenstrukturen zusammenknüpfen [S, L, T]
<code>del dic[key]</code>	Entferne das Element mit dem Schlüssel key [D]
<code>del liste[i]</code>	Entferne das Element an der Position i [L]
<code>liste.sort()</code>	Liste sortieren [L]
<code>liste.append(element)</code>	Element ans Ende der Liste anfügen [L]