

Ausgabe

```
print("Hello World")
print("Hello", "John")
name = "John"
print("Hello", name)
print("Hello
{}".format(name))
print() # erzeugt leere
Zeile und Zeilenumbruch
```

Eingabe

```
eingabe = input("Eingabe: ")
# Datentyp ist immer(!)
String
# evtl. Casting zu anderem
Datentyp
try:
    eingabe = int(eingabe)
except ValueError as err:
    print("Fehler!")
    print("Meldung:
{}".format(err))
```

Variablen

Kleinschreibung

Trennung mehrerer Wörter durch Unterstrich

Keine Ziffer am Beginn, nur Buchstabe oder Unterstrich

Sprechende Namen verwenden

Kommentare

Einzeilige Kommentare werden durch das Hash-Zeichen (Raute) eingeleitet. Mehrzeilige Kommentare werden mit drei Anführungszeichen begonnen und müssen auch wieder mit drei Anführungszeichen beendet werden.

```
# Dies ist ein Kommentar
""" Dies ist ein
langer Kommentar
"""
```

Fehlerbehandlung

```
try:
    # Anweisungen
except ExceptionType as err:
    # Anweisungen
    # Zugriff auf Fehlermeldung: err
```

ExceptionType immer explizit angeben

Operatoren

x+y	Addition
x-y	Subtraktion
x*y	Multiplikation
x/y	Division
x%y	Modulo
x**y	Exponentiation

Bedingungen

<	kleiner als	wert < 10
>	größer als	wert > 5
==	gleich	wert == 23
<=	kleiner gleich	5 <= 7
>=	größer gleich	23 >= 22
!=	ungleich	78 != 93
in	(enthalten) in	b in "Libelle"
not in	nicht (enthalten) in	y not in "Vogel"

Datentypen

Integer	-25, 34
Float	-2.53, 43.1
String	"Hello", 'World'
Boolean	True, False
List	[value, ...]
Tupel	(value, ...)
Dictionary	{key:value, ...}

Wiederholungen / Schleifen

for-Schleife

```
container = [1,2,3]
for element in container:
    print(element)
```

while-Schleife

```
x = 5
while x > 0:
    print("x: {}".format(x))
    x-=1
```

for-Schleife hat enumerate-Erweiterung zum Mitzählen der Indexposition
Gefahr von Endlosschleifen beim Einsatz von while

Bedingte Abfragen

```
if bedingung:
    # Anweisungen
elif andere_bedingung:
    # Anweisungen
else:
    # Anweisungen
```

beliebig viele elif-Abschnitte möglich
else erhält keine Bedingung
Abarbeitung von oben nach unten

Funktionen

```
def fname(param1, param2):
    ergebnis = param1 + param2
    return ergebnis
```

Laden und Verwenden eines Moduls

Importanweisung	Befehlsaufruf
<code>import modulname modulname.methode()</code>	
<code>dir(modulname)</code> zeigt alle Methoden eines Moduls an	
<code>help(methode)</code> zeigt Hilfe für eine Methode an	

Slicing bei Sequenzen

						[6:10]					
0	1	2	3	4	5	6	7	8	9	10	11
M	o	n	t	y		P	y	t	h	o	n
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
						[-12:-7]					

Arbeiten mit Strings

<code>len(mystr)</code>	Anzahl der Zeichen von mystr
<code>mystr.lower()</code>	Umwandlung zu Kleinbuchstaben
<code>mystr.upper()</code>	Umwandlung zu Großbuchstaben
<code>mystr.replace(old,new)</code>	old durch new ersetzen
<code>mystr.split(char)</code>	teilt an char auf, erzeugt Liste
<code>mystr[1:5]</code>	Zeichen 1-5 ausschneiden
<code>mystr[1:5:2]</code>	nur jedes zweite Zeichen

Arbeiten mit Dateien

```
with open(fname, mod) as var:
    # Anweisungen, z.B.
    c = var.read()
    cl = var.readlines()
    var.write(string_content)
```

`read()` liest Inhalt komplett in einen String ein
`readlines()` liest Inhalt zeilenweise in eine Liste ein
`write(string_content)` schreibt Stringinhalt in Datei

Arbeiten mit Listen

<code>len(mylist)</code>	Anzahl der Elemente
<code>mylist[i]</code>	i-tes Element der Liste
<code>mylist[i:j]</code>	Ausschnitt von i bis j
<code>mylist[i:j:2]</code>	nur jedes zweite Element
<code>x in mylist</code>	True, wenn x in mylist ist
<code>mylist.append(x)</code>	x an mylist anhängen
<code>mylist[i] = x</code>	Element an Stelle i ersetzen
<code>"c".join(mylist)</code>	erzeugt String, c verbindet

Arbeiten mit Dictionaries

<code>len(mydict)</code>	Anzahl der Einträge
<code>del mydict[key]</code>	löscht Schlüssel key
<code>list(mydict.keys())</code>	Liste aller Schlüssel
<code>list(mydict.values())</code>	Liste aller Werte
<code>list(mydict.items())</code>	Liste von Tupeln
<code>key in mydict</code>	True, wenn key vorhanden