Programmierhandzettel 2:

Vergleichsoperatoren, Funktionen zur Typwandlung, Conditional Statements

Vergleichsoperatoren liefern als Wert einen Boolean, also `True` oder `False`.

Operator	Beschreibung
`X < Y`	echt kleiner als
`X <= Y`	kleiner oder gleich
`X > Y`	echt größer als
`X >= Y`	größer oder gleich
`X == Y`	gleicher Wert
`X != Y`	ungleicher Wert
`X is Y`	Identisches Objekt (dieselbe id)
`X is not Y`	Negierte Objektidentität

Funktionen zur Typwandlung

Ergebnistyp (Kürzel)	Konvertierungsfunktionen	Anmerkungen
Integer (int)	`int(O[,basis])` `ord(C)`	Ist O ein String, muss O ein gültiges Literal für int sein, optional kann eine Basis (beliebiger Integer) angegeben werden. C ist ein String der Länge 1.
Float (float)	`float(O)`	Ist O ein String, muss O ein gültiges Literal für float sein.
Complex (complex)	`complex(O)`	Ist O ein String, so muss O ein gültiges Literal für complex, float oder int sein.
Boolean (bool)	`bool(O)`	Jeder Wert ≠ 0 bei numerischen Datentypen oder ein nichtleerer String liefert `True`.

String (str)	`str(O)`
	`repr(O)`
	
	`chr(I)`
	`hex(I)`
	`oct(I)`
		liefert ein pretty-print (String) von O.
		sehr ähnlich wie str(O), erzeugt ein von eval() auswertbaren String.
		Parameter `I` muss ein Integer im zulässigen Wertebereich sein, liefern alle Strings.

Ein- und Ausgabe: input() und print()

Syntax: `input('prompt'), prompt` ist ein String; input() liefert einen String

Syntax: `print([value [,value]*], sep = ' ', end = '\n', file = \sys.stdout, flush = False)`

Hier angegeben sind die Default-Werte für den Separator sep = , , für den ausgedruckten Abschluss des print()-Statements end =, file= und flush=. [file und flush sollten Sie zunächst nicht verändern.]

`value` kann alles sein, was einen Wert liefert, insbesondere ein Variablenname, ein Ausdruck, Literale, …

Conditional Expressions

nennt man auch conditional operator, inline if (iif) oder ternary if.

```
x = true_value if condition else false_value
```

… und ist ein Einzeiler. Aufpassen: **keine** Doppelpunkte!

- Der else-Zweig muss **immer** vorhanden sein.
- Operator Präzedenz der Conditional Expression ist geringer als or
  (dass heißt: man kann quasi alles in condition notieren ohne zu klammern.)

Zuerst wird condition ausgewertet (Boolean Context), dann abhängig vom Ergebnis true_value oder false_value errechnet und x zugewiesen.

Die Operator Präzedenz der Conditional Expression ist geringer als or
(das heißt: man kann quasi alles in condition notieren ohne zu klammern.)