Empfohlene Unterrichtsinhalte als Vorbereitung für dieses Thema: 01-02 (Was ist Python?), 02-01 (Variablen), 02-02 (Strings), 02-03 (Listen), 02-04 (Zahlen)

Einführung in die computerlinguistische Programmierung mit Python

Der Datentyp Bool - Wahrheitswerte in Python ✔ ✘

In den vergangenen Sitzungen habe Sie bereits einige Funktionen kennengelernt, die überprüfen, ob eine bestimmte Aussage wahr ist. Beispielsweise haben wir `.islower()` und `.isupper()` kennengelernt, die überprüfen ob ein String ausschließlich aus Groß- oder Kleinbuchstaben besteht. Ihr könnt die Funktionsaufrufe im folgenden Code-Block (rechts der Gleichheitszeichen) als Aussagen/Propositionen interpretieren. Der Python prüft diese Aussagen und gibt einen Wahrheitswert zurück. Diese Wahrheitswerte werden in Python **Bool**-Werte genannt und können genauso interpretiert werden wie **t** und **f**, die ihr vielleicht schon im Logik-Kurs kennengelernt habt. Welche Wahrheitswerte erwartet Ihr?

In [1]:
```
aussage_1 = "Nein".islower()
print(aussage_1)
print(type(aussage_1))

aussage_2 = ("DOCH".isupper())
print(aussage_2)
print(type(aussage_1))

aussage_3 = "0H!".isupper()
print(aussage_3)
```

Der Datentyp Bool besteht aussschließlich aus den Werten `True` und `False`; es gibt keinen Bool-Wert der Unbestimmtheit/Unbestimmbarkeit ausdrückt.

Auch mit Hilfe von **Vergleichsoperatoren**, die wir bereits im Zusammenhang mit Zahldatentypen kennengelernt haben, werden Aussagen dargestellt, deren Auswertung Bool-Werte zurück gibt.

In [1]:
```
aussage_4 = ("A" < "B")
print(aussage_4)

aussage_5 = (0.0 == 0)
print(aussage_5)

aussage_6 = ([] != [[], [], []])
print(aussage_6)
```

Logische Operatoren

Einfache logische Aussagen können mit den logischen Operatoren `and`, `or` und `not` zu komplexen Aussagen verknüpft werden.

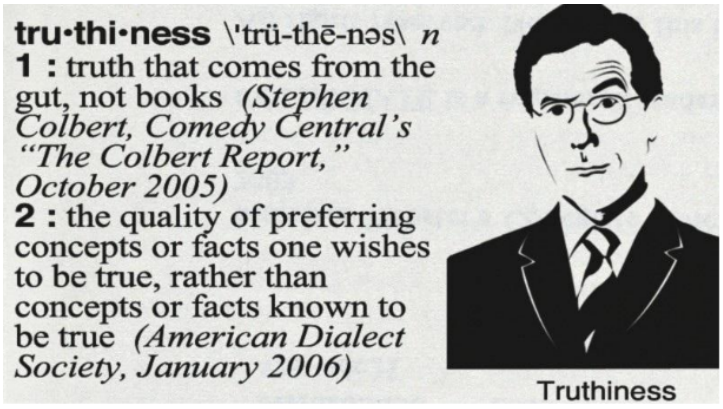
In [1]:
```
aussage_7 = True and False
print(aussage_7)

aussage_8 = True and not False
print(aussage_8)

aussage_9 = True or False
print(aussage_9)

aussage_10 = True or False and True and not False
```

Die Funktion Bool() und Truthiness



Mit Hilfe der Funktion `bool()` können Werte eines anderen Typs in Bool-Werte abgebildet werden:

In [1]:
```
print(bool("Hallo Welt"))
print(bool(9))
print(bool(9.0))
print(bool(["Hallo Welt", 9, 9.0]))

print(bool(""))
print(bool(" "))
print(bool(0))
print(bool(0.0))
print(bool([]))
```

Bei der Umwandlung anderer Typen kommt deren *Truthiness* zum Ausdruck. D.h. Werte die leer sind - etwa `""` und `" "` - oder Null entsprechen werden auf `False` abgebildet. Alle anderen Werte sind *truthy* und werden auf `True` abgebildet.

In [1]:
```
print(bool(1 - 1))
print(bool(9 * 0))
```

Zusammenfassung

- Mit Hilfe von Boolean-Werten können Wahrheitswerte dargestellt werden.
- Logikoperatoren ermöglichen die Konstruktion von komplexen Aussagen.
- Werte anderer Typen können mit Hilfe der Funktion `bool()` auf Boolean-Werte abgebildet werden.

Und nächste Woche lernen wir ...

- wie man mit Hilfe von Indexing und Slicing auf Teile bestimmter Daten zugreifen kann.
- wie Bool-Werte und Aussagen in Bedingungen genutzt werden um innerhalb eines Programms Entscheidungen zu treffen.
- wie Schleifen helfen repetitive Aufgaben zuverlässig zu bewältigen.
- wie man mit Python Zufallszahlen generieren kann