Empfohlene Unterrichtsinhalte als Vorbereitung für dieses Thema: 01-02 (Was ist Python?)

Einführung in die computerlinguistische Programmierung mit Python

02-01: Variablen in Python 📩

Als nächstes beschäftigen wir uns damit, wie Informationen in Python-Programmen gespeichert und während der Laufzeit verändert werden können. Dazu erzeugt man im Programmcode **Variablen**, die bestimmte Werte annehmen können, und die man mit bestimmten Befehlen verändern kann.

Wird eine Variable im Programmcode aufgerufen, beispielsweise mit `print()`, bezieht der Aufruf sich immer auf den Wert, der in der Variable gerade gespeichert ist. Der Interpreter bewegt sich von oben nach unten durch das Programm und führt nacheinander alle Befehle aus.

Um eine neue Variable zu erzeugen, überlegen wir uns zunächst einen **Variablennamen**, z.B. `aktuelle_zahl`. Als nächstes folgt das Gleichheitszeichen als **Zuweisungsoperator**. Darauf folgt der **Wert**, den wir der Variable zuordnen wollen.

```
aktuelle_zahl = 0
```

Um den Code lesbarer zu gestalten, ist es üblich vor und nach Operatoren ein Leerzeichen `" "` einzufügen.

Wenn wir jetzt `print(aktuelle_zahl)` aufrufen, wird der aktuelle Wert der Variable ausgegeben.

Versucht den Wert im Kästchen unten zu verändern und den Code dann mit $Ctrl + Enter$ auszuführen, um zu sehen, dass immer der aktuelle Wert der Variable angezeigt wird.

In [1]:
```
aktuelle_zahl = 0
print(aktuelle_zahl)
```

Wir können den Wert der Variable beliebig oft überschreiben, indem wir mehrere Wertzuweisungen im Code einfügen. Nach jeder Änderung wird beim Aufruf von `print()` der zuletzt zugewiesene Wert angezeigt:

In [1]:
```
aktuelle_zahl = 0
print(aktuelle_zahl)

aktuelle_zahl = 1
print(aktuelle_zahl)

aktuelle_zahl = "das ist gar keine Zahl!"
print(aktuelle_zahl)

aktuelle_zahl = 5/2
print(aktuelle_zahl)
```

Variablen & Typen

Wie man sieht, können wir der Variable $\text{aktuelle}_{\text{zahl}}$ ganz unterschiedliche Werte zuweisen. Welchen Typ die Variable gerade hat, können wir mit dem Befehl `type(aktuelle_zahl)` ermitteln. Das Ergebnis dieses Aufrufs wird dann noch in einen Print-Befehl verschachtelt. Probiert es aus und ergänzt im Kästchen oben nach jeder Änderung der Variable einen Aufruf von `print(type(aktuelle_zahl))`!

Wir sehen, dass die Variable zunächst zum Typ `<class 'int'>` gehört, später dann zu den Typen `<class 'str'>` und `<class 'float'>`. Es gibt noch mehr grundlegende Datentypen in Python – hier eine Auswahl:

Kurzbezeichnung	Bedeutung	Beispiel
`int`	Integer (ganze Zahl)	`zahl = 5`
`str`	String (Zeichenkette)	`wort = "Python"`
`float`	Float (Kommazahl)	`kommazahl = 3.1415`
`list`	Liste (geordnete Sequenz von Werten)	`liste = [1,2,3]`
`dict`	Dictionary (Sequenz von Wertpaaren mit eindeutiger Zuordnung)	`lexikon = {"dog": "Hund"}`

Kurzbezeichnung	Bedeutung	Beispiel
`bool`	Wahrheitswert/Boolean	`True`, `False` (Großschreibung beachten!)

Aufgabe

Welche Ausgaben können wir mit Hilfe der obigen Tabelle vorhersagen? Denkt über jedes Beispiel zuerst nach und führt es dann aus, um Eure Vermutung zu überprüfen.

In [1]:
```
print(type("Nie ohne Seife waschen"))
```

In [1]:
```
print(type(["Wer", "Summen", "kürzt", "der", "ist", "ein", "Schaf"]))
```

In [1]:
```
print(type(4-9))
```

In [1]:
```
print(type("Hallo" + " " + "Welt"))
```

In [1]:
```
print(type({"S": "NP VP"}))
```

Variablennamen: Regeln & Konventionen

In den Variablenzuweisungen oben haben wir uns bisher einfach einen Namen ausgedacht und diesem einen Wert zugewiesen. Wir haben ebenfalls schon gesehen, dass Python von sich aus einige Befehle und Funktionswörter kennt. Diese sollten wir deshalb nicht als Variablennamen nutzen, weil dies zu Verwirrung beim Programmieren führen könnte und auch der Python-Interpreter sich unerwartet verhalten könnte. Darüber hinaus gibt es einige Vorgaben, die wir bei der Benennung von Variablen beachten müssen:

1. Variablennamen müssen mit einem Buchstaben `a–z` oder einem Unterstrich `_` beginnen. Unterstriche dürfen auch innerhalb des Namens auftauchen.

In [1]:
```
variable_eins = 1
print(variable_eins)

_variable_zwei = 2
print(_variable_zwei)
```

1. Python ist “case-sensitive”, d.h. die Groß-/Kleinschreibung wird vom Interpreter unterschieden. Die folgenden Variablen sind verschieden voneinander:

In [1]:
```
variable_eins = 1
Variable_eins = 2
VARIABLE_EINS = 3
VariableEins = 4
VAriable_eins = 5
# Und so weiter...
print(variable_eins)
print(Variable_eins)
print(VARIABLE_EINS)
print(Variable_Eins)
print(VAriable_eins)
```

Um einen längeren Variablennamen, der aus mehr als einem Wort besteht, gut lesbar zu halten, kann man die Wortgrenzen entweder per Unterstrich oder mit Hilfe des **CamelCase** verdeutlichen. ImCamelCaseWirdImmerDerErsteBuchstabeEinesWortsGroßGeschrieben, wie es in Zeile 4 des Codeblocks oben der Fall ist. Die groß geschriebenen Buchstaben erinnern dabei an die Höcker von Kamelen. (Weitere Informationen: Binnenmajuskel)

1. Variablennamen dürfen Ziffern `0–9` enthalten. Sie dürfen aber nicht mit Ziffern beginnen, da der Interpreter diese als Zahlen erkennt und versucht diese entsprechend zu interpretieren.

In [1]:
```
variable1 = "eins"
print(variable1)
var1ab1e2 = "zwei"
print(var1ab1e2)

# 1variable = "Hmmmm"
```

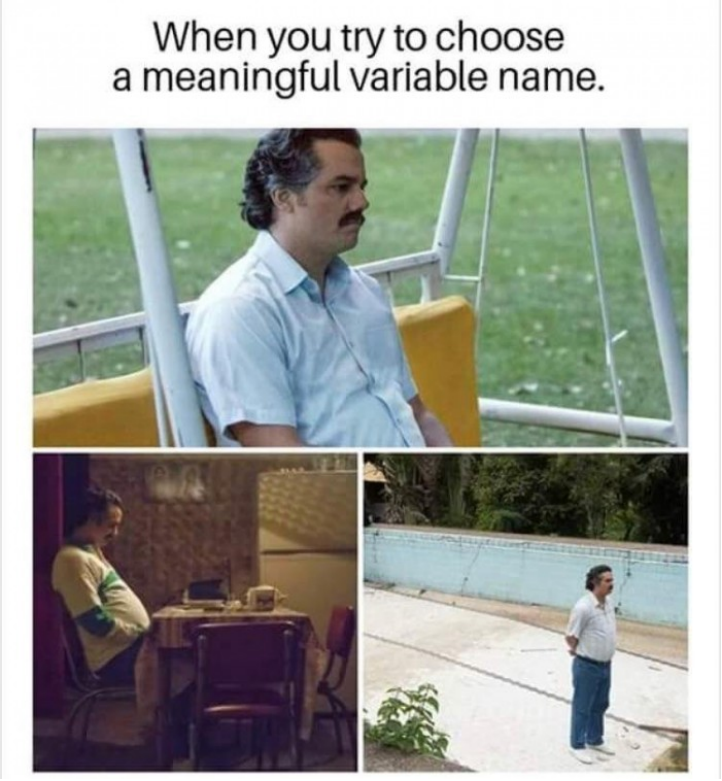
1. Interpunktionszeichen, wie `.` und `,`, sowie Sonderzeichen und alle Arten von Klammern, wie `$`, `%`, und `()`, dürfen nicht in Variablennamen benutzt werden. Auch das Leerzeichen `" "` darf nicht innerhalb von Variablennamen genutzt werden.

Problemfälle

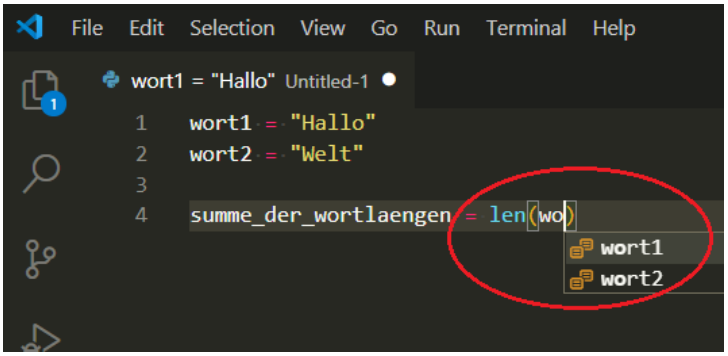
Ausgehend von den obigen Regeln: Welche der folgenden Variablennamen sind zulässig?

In [1]:
```
#Var = "This is the 12th try"

#H4.s = "This is the 13th try"

#u89B = "This is the 14th try"

#8nas = "This is the 15th try"

#iN8nF = "This is the 16th try"

#name(1) = "This is the 17th try"

#"Produkt" = "This is the 18th try"

#'element' = "This is the 19th try"

#xy34d = "This is the 20th try"

#_r_ = "This is the 21st try"

#@km_anzahl = "This is the 22nd try"
```

Sprechende Namen & Konventionen



Idealerweise solltet Ihr Namen so wählen, dass man auf den ersten Blick erkennt, was der Inhalt der Variable ist. Wir reden dann von **sprechenden Variablennamen**. Grundsätzlich sollte man sowohl die Länge der Veriablennamen, als auch deren Verständlichkeit für andere im Auge behalten. VS Code hilft Euch einmal initialisierte Variablennamen zu erinnern, in dem es bei der Eingabe Vorschläge zur Vervollständigung der Namen macht.



In [1]:
```
wort1 = "Hallo"
wort2 = "Welt"

summe_der_wortlaengen = len(wort1) + len(wort2)

pie = 3  # Au weia!
```

Darüber hinaus gibt es einige Konventionen, die wir bei der Bennenung von Variablen im Hinterkopf behalten sollten. So werden Variablen, die als Wert nur einzelne Zeichen annehmen, werden oft mit `c` für "character" bezeichnet; solche, die Zahlen enthalten mit `n` für "number" und andere, die Wörter enthalten mit `w` für "word".

| Art der Variablenwerte | Konvention für die Bezeichnung |

| Einzelne (alpha-numerische) Zeichen | `c` für character |

| Zahlen | `n` für number |

| Wörter | `w` für word |

Zusammenfassung

- In Python werden Variablen erzeugt indem man einem Variablennamen einen Wert zuweist.
- Variablen können beliebig oft überschrieben werden.
- Mit der Funktion `type()` kann man der Type eines Variablenwerts bestimmen.
- Es gibt verschiedene Regeln und Konventionen zur Benennung von Variablen.
- Sprechende Variablennamen sind hilfreich und wünschenswert.

Weitere Themen dieser Woche

- 02-02: Strings
- 02-03: Listen
- 02-04: Integers und Floats - Zahlen in Python
- 02-05: Bool - Wahrheitswerte