

Arbeiten mit grundlegenden Sprachelementen

1. Was verstehen Sie unter einer Variablen, und welche Gültigkeitsbereiche gibt es?

Eine Variable ist ein vom Programm verwalteter Speicherbereich, dessen Größe vom Datentyp der Variablen abhängig ist. Um einer Variablen einen Wert zuzuweisen, kann der Programmierer sie über deren Namen ansprechen.

Globale Variablen sind im gesamten Programm gültig.

Lokale Variablen werden innerhalb eines Anweisungsblocks (bzw. innerhalb einer Funktion oder Methode) definiert und sind auch nur in diesem Bereich gültig.

Statische Variablen bleiben während der gesamten Programmlaufzeit erhalten, besitzen jedoch eine Gültigkeit wie lokale Variablen.

Externe Variablen bleiben ebenfalls während der gesamten Programmlaufzeit erhalten, werden aber von einer anderen Programmdatei bzw. einer Bibliothek bereitgestellt.

2. Welche Zahlenart kann eine mit `int` (integer) deklarierte Variable in Python darstellen?

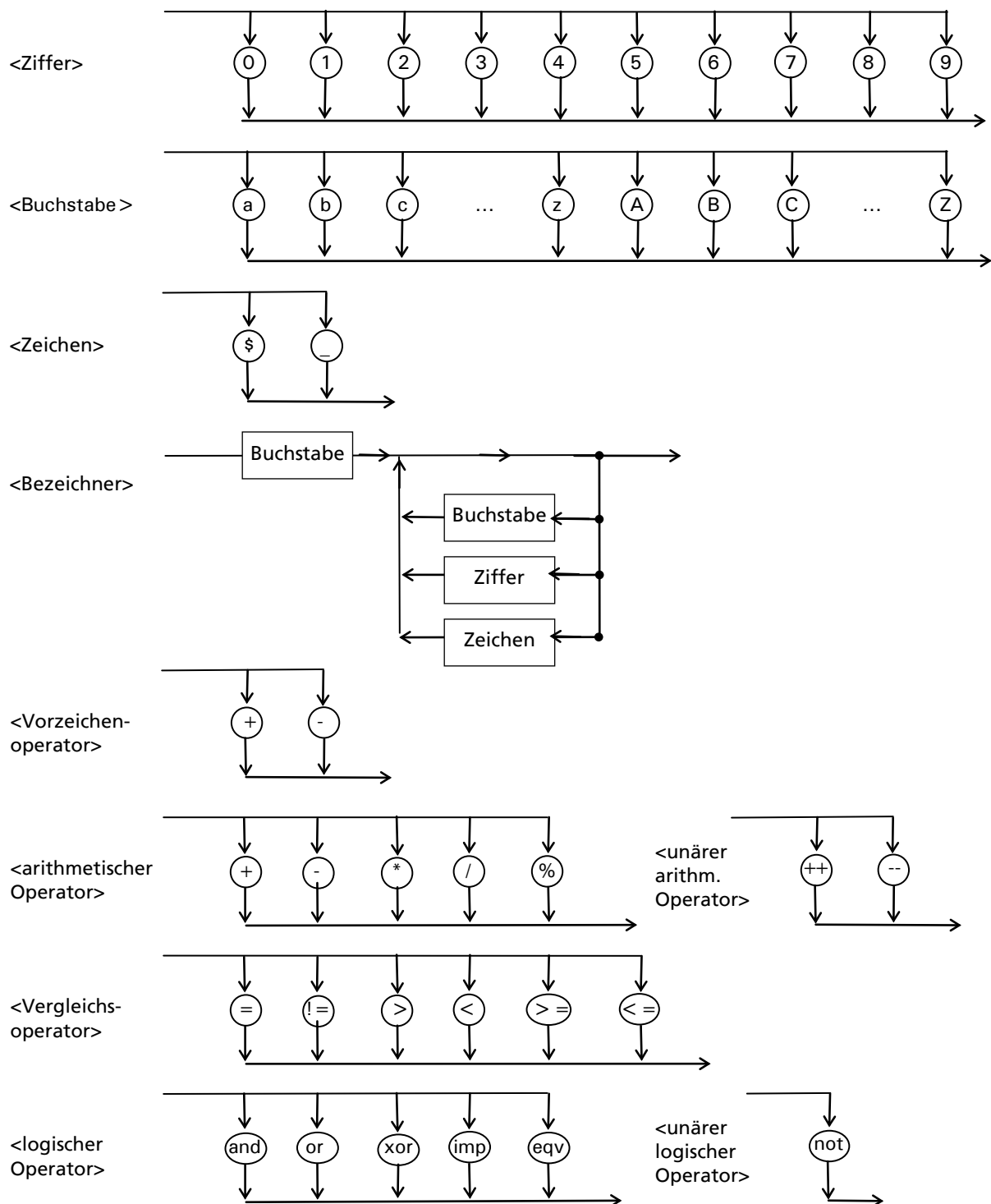
Eine Integervariable kann positive und negative Ganzzahlen speichern.

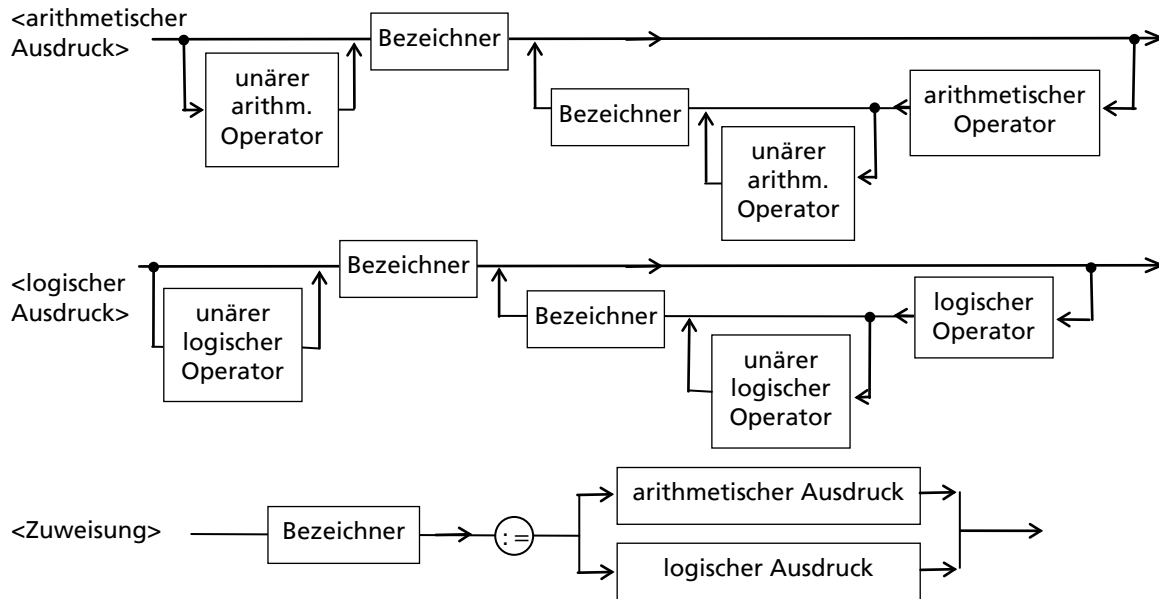
3. Zwei Variablen, `var1` und `var2`, vom Datentyp `char` (character) sind gegeben. Die Werte der beiden Variablen sollen vertauscht werden. Beschreiben Sie den Algorithmus im Pseudocode.

```
Initialisierung der Variablen var1, var2, hilfsvARIABLE
Zuweisung der Werte der Variablen var1, var2
Ausgabe der Variablen vor dem Tausch
hilfsvARIABLE = var1
var1 = var2
var2 = hilfsvARIABLE
Ausgabe der Variablen nach dem Tausch
```

4. Stellen Sie die arithmetischen, logischen und Vergleichsoperatoren in einem Syntaxdiagramm und in der erweiterten BNF dar. Fassen Sie dabei unäre und binäre Operatoren in je einer Darstellung zusammen. Verwenden Sie jeweils eines der möglichen Zeichen pro Operator. Erzeugen Sie das Syntaxdiagramm und die erweiterte BNF für einen Ausdruck. Verwenden Sie dabei unäre und binäre Operatoren.

Syntaxdiagramm



erweiterte BNF

<ziffer>	::= "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
<buchstabe>	::= "a" "b" "c" ... "z" "A" "B" "C" ... "Z"
<zeichen>	::= "\$" "_"
<bezeichner>	::= <buchstabe>{<buchstabe> <ziffer> <zeichen>}
<vorzeichen-operator>	::= "+" "-"
<arithmetischer operator>	::= "+" "- "*" "/" "%"
<unärer arithm. operator>	::= "++" "--"
<vergleichsoperator>	::= "=" "!=" ">" <" >=" <="
<logischer operator>	::= "and" "or" "xor" "imp" "eqv"
<unärer logischer operator>	::= "not"
<arithmetischer ausdruck>	::= {[<unärer arithm. operator>] <bezeichner> <arithmetischer operator>} [<unärer arithm. operator>] <bezeichner>
<logischer ausdruck>	::= {[<unärer logischer operator>] <bezeichner> <logischer operator>} [<unärer logischer operator>] <bezeichner>
<zuweisung>	::= <bezeichner> ":=" <arithmetischer ausdruck> <logischer ausdruck>

5. Werten Sie folgende Bedingungen aus:

- ✓ `(value1 > 500) or (value2 < 800) and not (value3 == 400)`
- ✓ `(value1 > 500) or ((value2 < 800) and not (value3 == 400))`

Die Variablen sollen folgende Werte enthalten:

`value1 = 501; value2 = 799; value3 = 400`

In den Bedingungen wird angenommen, dass die Abarbeitung von links nach rechts erfolgt.

Auswertung: `(value1 > 500) or (value2 < 800) and not (value3 == 400)`

	<code>(value1 > 500)</code>	<code>or</code>	<code>(value2 < 800)</code>	<code>and</code>	<code>not (value3 == 400)</code>
1.	true	or	true	and	true
2.	true			and	false
3.	false				

Auswertung: `(value1 > 500) or ((value2 < 800) and not (value3 == 400))`

	(value1 > 500)	or	((value2 < 800)	and	(value3 == 400))
1.	true	or	(true	and	false)
2.	true	or	false				
3.	true						

Durch die Kammersetzung ändert sich der Wahrheitswert des gesamten Ausdrucks.