

Variable notation and datatypes

\$aFruitsList	Array (\$a)
\$bIsVisible	Bool (\$b)
\$dBinData	Binary (\$d)
\$fPrice	Float (\$f)
\$hGui	Handle (\$h)
\$iNumber	Integer (\$i)
\$idButtonOk	GUI control id (\$i)
\$mPairs	Map (\$m)
\$oExcel	Object (\$o)
\$pRect	Pointer (\$p)
\$sText	String (\$s)
\$tSTRUCT	Struct (\$t)
\$vData	Variant (\$v)

☰ [Complete list here](#)

Variable scope

Global	Scope over the entire program
Local	Scope related to the block of code
Dim	Hybrid-like scope, local if not exists already globally

❗ Tip: Avoid Dim, use explicit Local or Global

❗ Tip: Don't use Global in function scope

Assignment operators

=	Assignment	\$sText = 'Hi'
&	Concatenates/joins two strings	\$sText & ' Max'
&=	Concatenation assignment	\$sText &= ' Max' equivalent to \$sText = \$sText & ' Max'
+=	Addition assignment	\$iNum += 1 equivalent to \$iNum = \$iNum + 1
-=	Subtraction assignment	\$iNum -= 1 equivalent to \$iNum = \$iNum - 1
*=	Multiplicative assignment	\$iNum *= 2 equivalent to \$iNum = \$iNum * 2
/=	Divisive assignment	\$iNum /= 2 equivalent to \$iNum = \$iNum / 2

Arithmetic operators

+	Addition	10 + 20
-	Subtraction	20 - 10
*	Multiplication	20 * 10
/	Division	20 / 10
^	Power	2 ^ 4
Mod	Modulus	Mod(val1, val2)

Comparison operators

=	Equal (CI)	If \$sText = 'Hi' Then ..
==	Strict equal (CS)	If \$sText == 'Bye' Then ..
<>	Not equal (CI)	If \$sText <> 'hello' Then ..
>	Greater than (CL)	
>=	Greater than or equal (CL)	
<	Less than (CL)	
<=	Less than or equal (CL)	

❗ Legend: CI: case insensitive, CS: case sensitive, CL: compared lexicographically

Conditional operator

? :	Ternary	See section "Branching"
-----	---------	-------------------------

Logical operators

And	Logical And	If \$iNum1 = 5 And \$iNum2 > 6 Then ..
Or	Logical Or	If \$iNum1 = 5 Or \$iNum2 > 6 Then ..
Not	Logical Not	If Not \$iNum = 5 Then ..

Comments

;	Single line comment
#cs	
	This is a multi line comment
#ce	



By **SOLVE-SMART** (SOLVE-SMART)
cheatography.com/solve-smart/

github.com/Sven-Seyfert

Published 23rd February, 2023.
 Last updated 23rd February, 2023.
 Page 1 of 3.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>

Quotes and escaping

"First string"	Strings are enclosed in double-quotes
'Second string'	Strings can also be enclosed in single-quotes
"His name was ""Max"""	Double-quotes as escape character
'His name was "- Max"'	Or single-quotes instead
'I'm a string'	Single-quotes as escape character
"I'm a string"	Or double-quotes instead

☰ [Complete list here](#)

Functions

Without parameters	Func _PrintHello() ConsoleWrite('Hello' & @CRLF) EndFunc
With parameters	Func _PrintText(\$sText) ConsoleWrite(\$sText & @CRLF) EndFunc
With optional parameters	Func _SendWithDelay(\$sKeys, \$iDelay = 150) Send(\$sKeys) Sleep(\$iDelay) EndFunc
With Default keyword parameter	Func _SendWithDelay(\$sKeys, \$iDelay = Default) \$iDelay = (\$iDelay == Default) ? 150 : \$iDelay Send(\$sKeys) Sleep(\$iDelay) EndFunc

❗ Function calls:

```
_PrintHello()
_PrintText('This is a text')
_SendWithDelay('{ENTER}')
_SendWithDelay('{ENTER}', 300)
_SendWithDelay('{ENTER}', Default)
```

Loops

For..Next	For \$i = 1 To 10 Step 1 ConsoleWrite(\$i & @CRLF) Next
For..In..Next	For \$sFruit In \$aFruitsList ConsoleWrite(\$sFruit & @CRLF) Next

Loops (cont)

While..WEnd	While \$iNum < 25 Sleep(200) \$iNum += 1 WEnd
Do..Until	Do Sleep(200) \$iNum += 1 Until \$iNum = 25

Branching

If..Elseif..Else	If \$iNum = 1 Then ; do something Elseif \$iNum = 2 Then ; do something Else ; do something EndIf
Select	Select Case \$iNum = 42 ; do something Case \$sText = 'Test' ; do something Case Else ; do something EndSelect
Switch	Switch @HOUR Case 6 To 11 \$sMessage = 'Good Morning' Case 12 To 17 \$sMessage = 'Good Afternoon' Case Else \$sMessage = "Later than 17 o'clock" EndSwitch
Ternary	\$sVariable = (condition) ? 'thenExpression' : 'elseExpression' \$bIsAnswerToLife = (\$iNum = 42) ? True : False



Command line

\$CmdLine[0]	Total number of given command line call parameters
\$CmdLine[1]	First parameter (value) of given command line call

i Command line call (e.g. as GUI parameter):

```
C:\Development\AutoIt>MyProgram.exe "800" "600"
$GuiWidth = $CmdLine[1]
$GuiHeight = $CmdLine[2]
```

Macros (special read-only variables)

@Compiled	Indicates whether the script is compiled or not
@CRLF	Carriage Return Line Feed
@error	Status of the error flag
@extended	Extended function return
@HOUR	Hours value of clock
@MIN	Minutes value of clock
@SEC	Seconds value of clock
@ScriptDir	Directory containing the running script

☰ Complete list [here](#)

Directives

#include	Includes a file in the current script #include <Array.au3> or #include "path\filename.au3"
#include-once	Current file may only be included once
#NoTrayIcon	AutoIt tray icon will not be shown
#OnAutoItStartRegister	Registers a function to be called when AutoIt starts
#pragma	Changes how the script is compiled
#RequireAdmin	Current script requires full administrator rights to run

Au3Check* common parameters

-d	As Opt("MustDeclareVars", 1)
-w 1	Already included file (on)
-w 2	Missing #comments-end (on)
-w 3	Already declared var (off)
-w 4	Local var used in global scope (off)
-w 5	Local var declared but not used (off)

Au3Check* common parameters (cont)

-w 6	Warn when using Dim (off)
-w 7	Warn when passing Const or expression on ByRef param(s) (on)

i Au3Check* checks the script for syntax errors

☰ Complete list [here](#)

Au3Stripper* common parameters

/sf	Strip all unused Func's
/sv	Strip unused Global and Local variable declarations
/mo	Just merges the Include files into the source and strips the Comments This is similar to aut2exe and helps finding the errorline
/rm	Rename Variables and Functions to a shorter name
/rsln	Replace @ScriptLineNumber with the actual line number

i Au3Stripper* clean up the script through various options

☰ Complete list [here](#)

