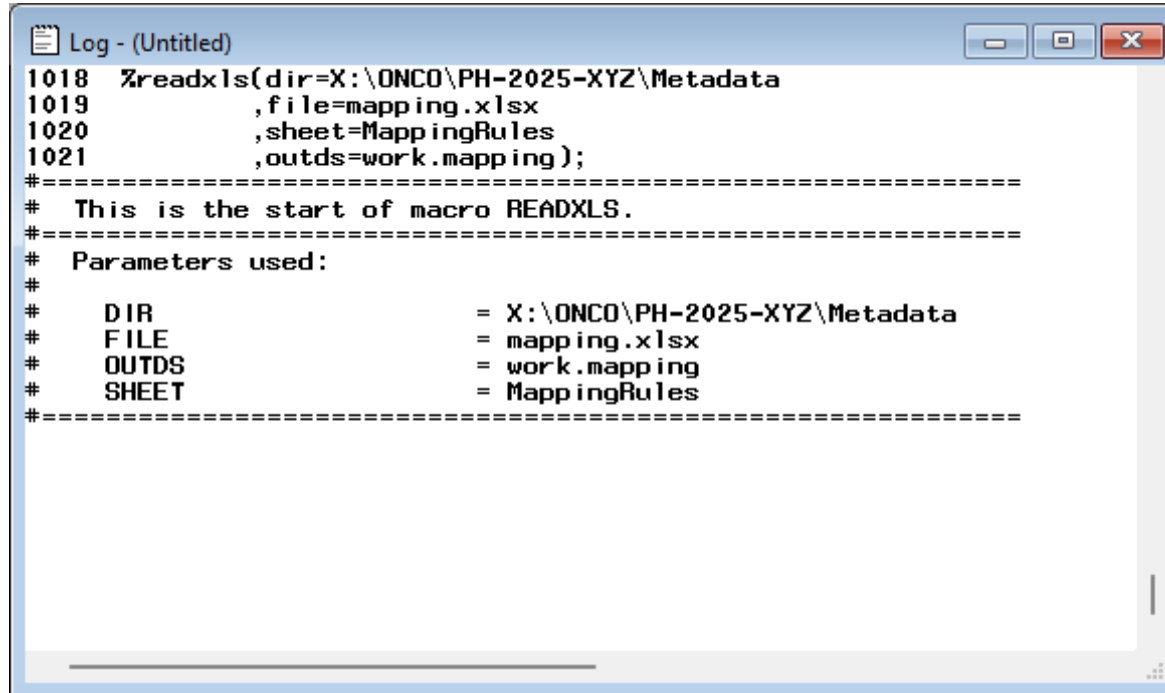


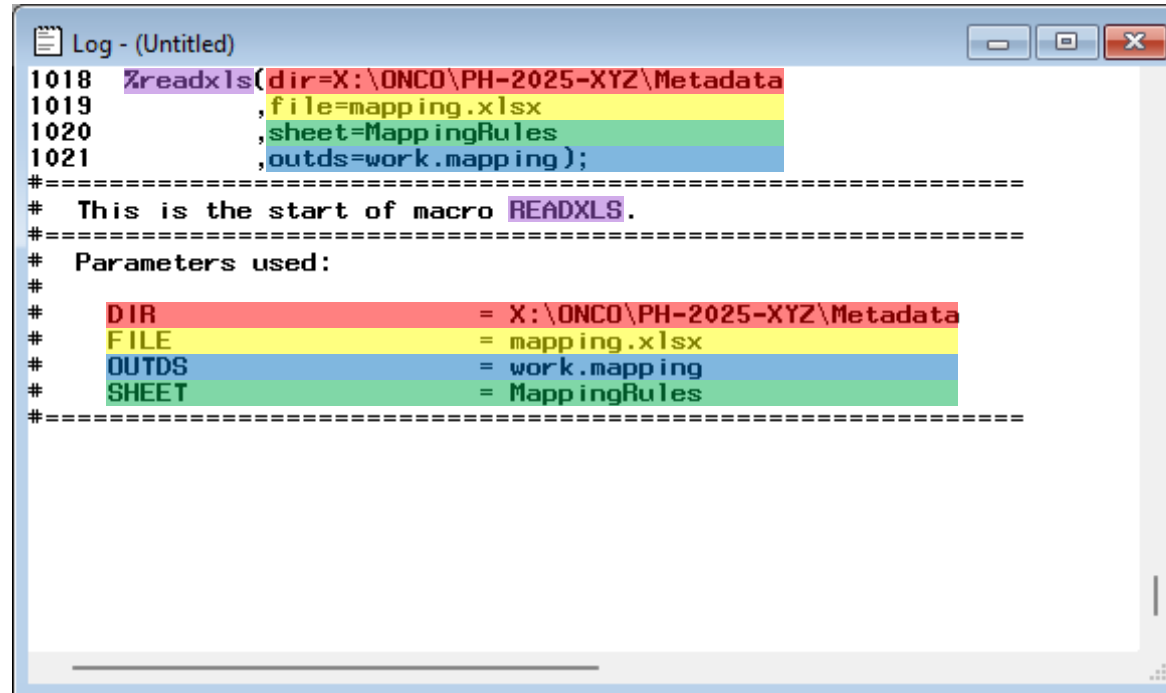
Smarter SAS logging—with a macro that AI hasn't mastered (yet)

Jules van der Zalm

PHUSE EU Connect 2025 | Scripts, Macros & Automation | SM15

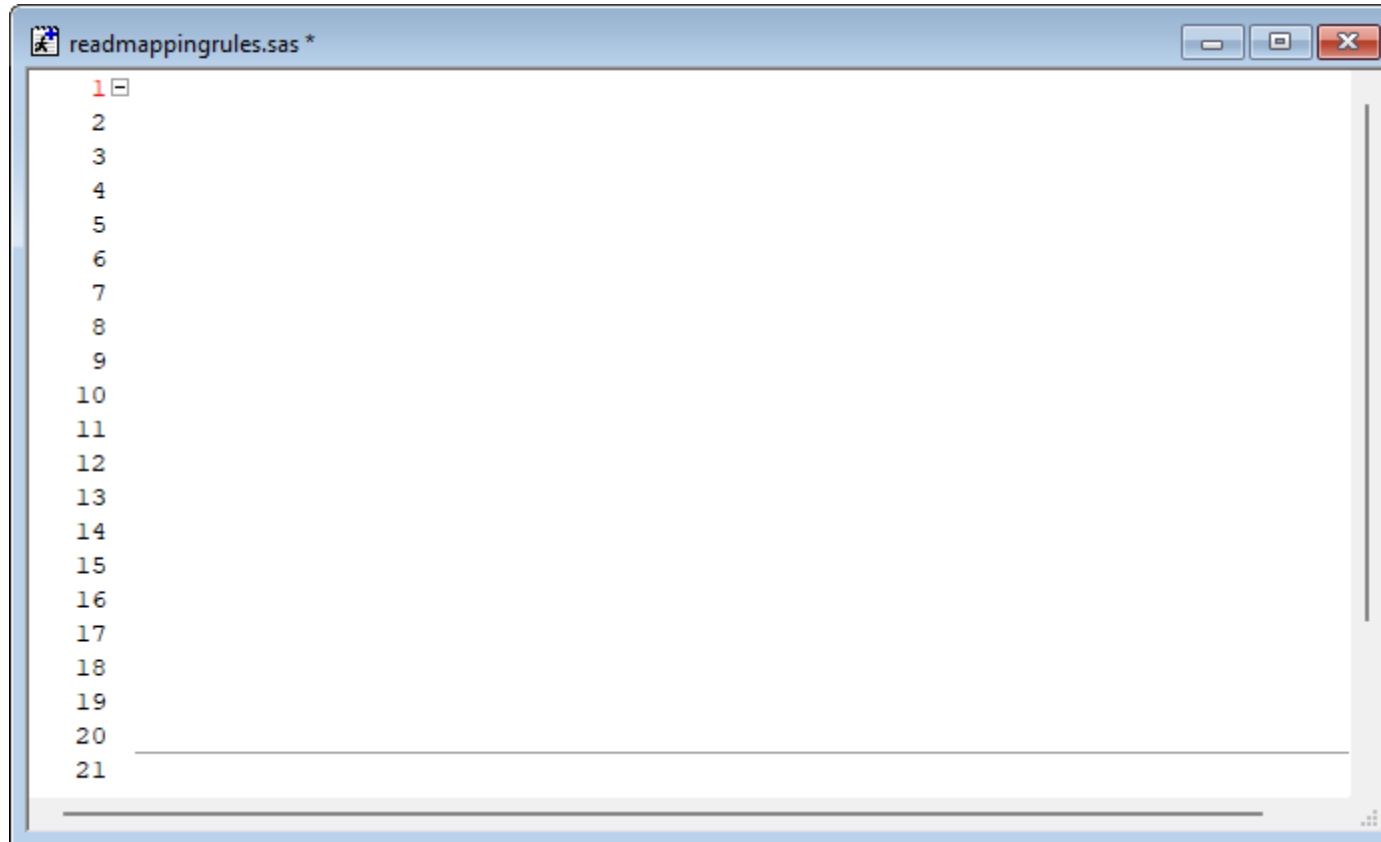
A screenshot of a log window titled "Log - (Untitled)". The window contains the following text:

```
1018 %readxls(dir=X:\ONCO\PH-2025-XYZ\Metadata
1019           ,file=mapping.xlsx
1020           ,sheet=MappingRules
1021           ,outds=work.mapping);
=====
# This is the start of macro READXLS.
=====
# Parameters used:
#
#   DIR           = X:\ONCO\PH-2025-XYZ\Metadata
#   FILE          = mapping.xlsx
#   OUTDS         = work.mapping
#   SHEET         = MappingRules
=====
```

A screenshot of a Notepad window titled 'Log - (Untitled)'. The window contains VBA code for a macro named 'READXLS'. The code is as follows:

```
1018 %readxls(dir=X:\ONCO\PH-2025-XYZ\Metadata
1019           ,file=mapping.xlsx
1020           ,sheet=MappingRules
1021           ,outds=work.mapping);
=====
# This is the start of macro READXLS.
=====
# Parameters used:
#
#   DIR           = X:\ONCO\PH-2025-XYZ\Metadata
#   FILE          = mapping.xlsx
#   OUTDS         = work.mapping
#   SHEET         = MappingRules
=====
```

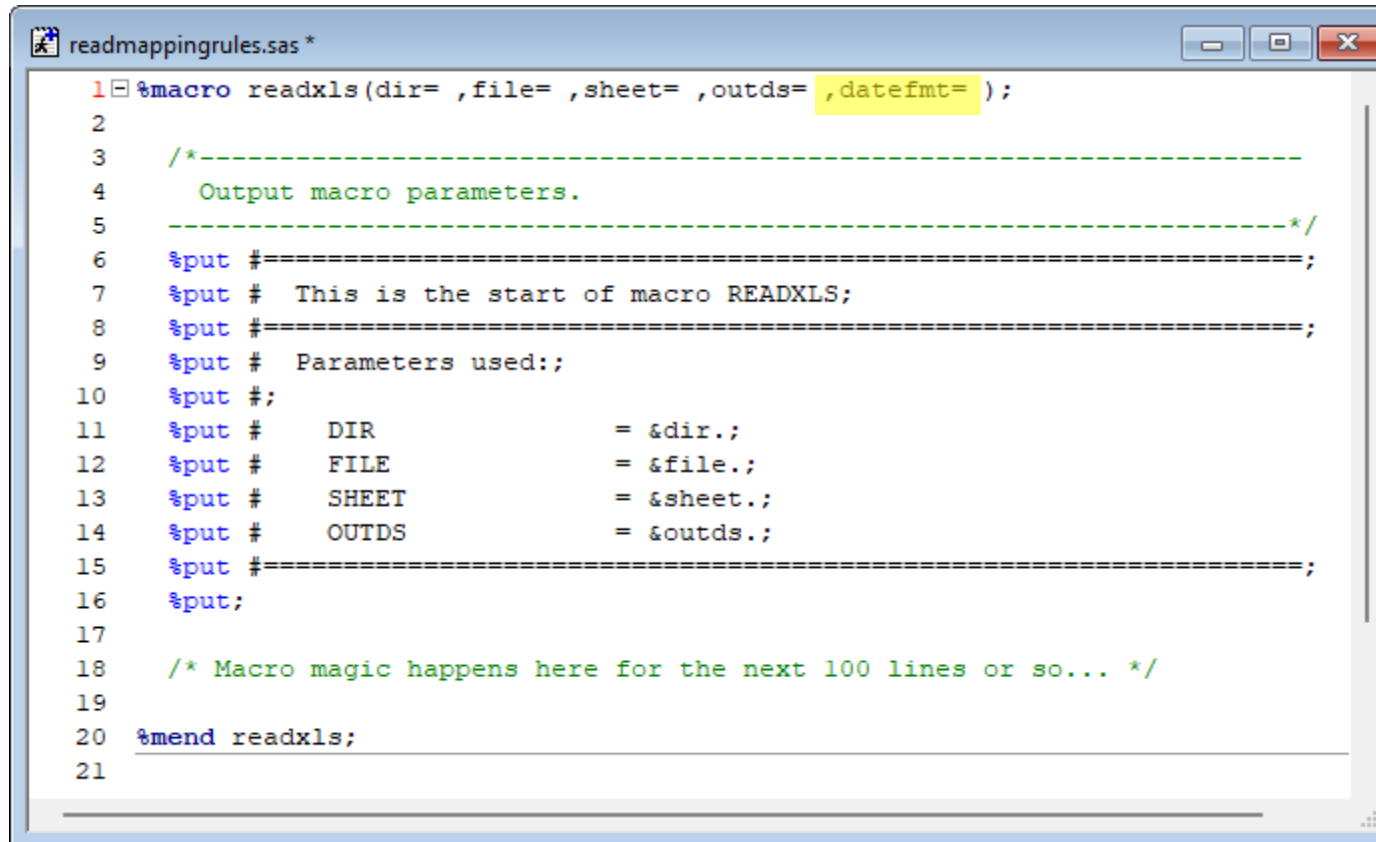
The code is color-coded: the macro name is purple, the directory path is red, the file name is yellow, the sheet name is green, and the output dataset name is blue. The parameter list at the bottom also uses these colors to match the code above.



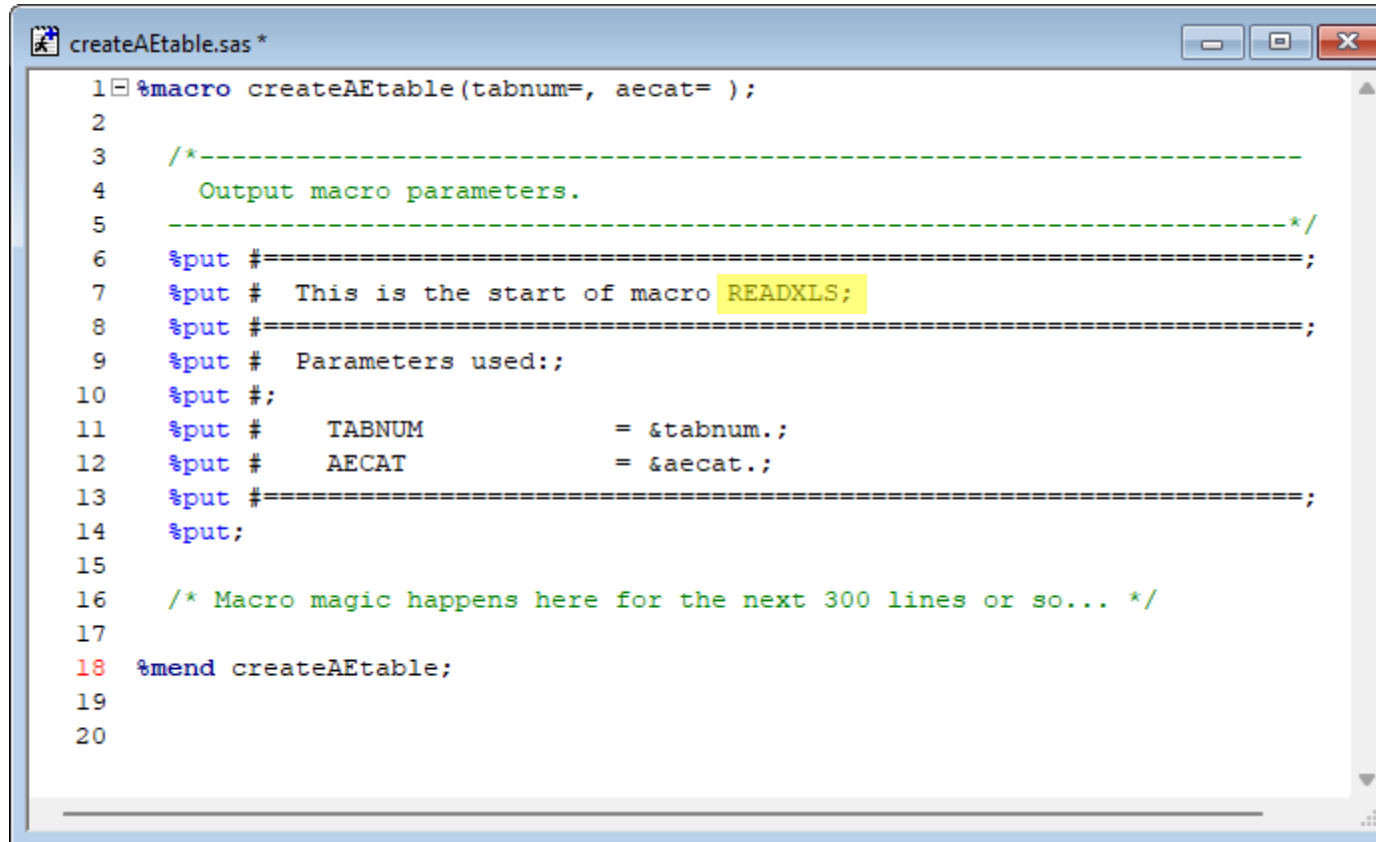
A screenshot of a SAS editor window titled "readmappingrules.sas *". The window contains a list of numbers from 1 to 21, with line 1 highlighted in red. The numbers are listed vertically on the left side of the editor area.

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21
```

But this approach has risks!



```
1 %macro readxls(dir= ,file= ,sheet= ,outds= ,datefmt= );
2
3  /*-----
4   Output macro parameters.
5   -----*/
6  %put #=====;
7  %put # This is the start of macro READXLS;
8  %put #=====;
9  %put # Parameters used;;
10 %put #;
11 %put # DIR = &dir.;
12 %put # FILE = &file.;
13 %put # SHEET = &sheet.;
14 %put # OUTDS = &outds.;
15 %put #=====;
16 %put;
17
18 /* Macro magic happens here for the next 100 lines or so... */
19
20 %mend readxls;
21
```

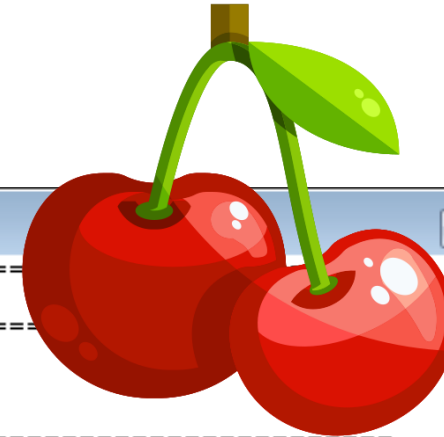
A screenshot of a SAS editor window titled 'createAETable.sas *'. The window contains a SAS macro definition. The macro is named 'createAETable' and takes two parameters: 'tabnum=' and 'aecat='. The macro body includes several lines of code: a comment 'Output macro parameters.', a series of '%put' statements to output the macro parameters and their values, and a comment 'Macro magic happens here for the next 300 lines or so...'. The macro ends with '%mend createAETable;'. The line number 18 is highlighted in red in the original image.

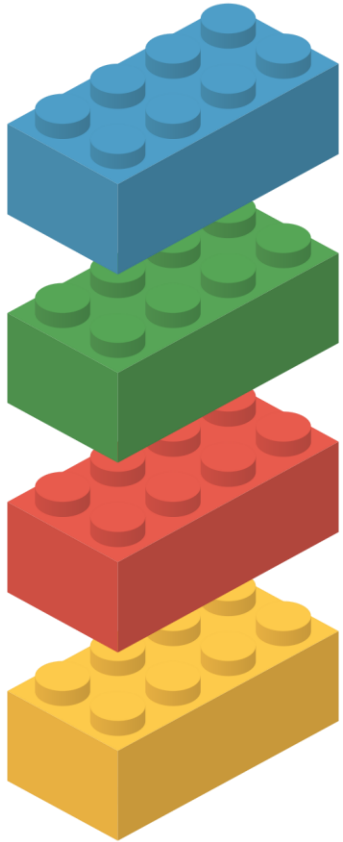
```
1 %macro createAETable(tabnum=, aecat= );
2
3  /*-----
4   Output macro parameters.
5  -----*/
6  %put #=====;
7  %put # This is the start of macro READXLS;
8  %put #=====;
9  %put # Parameters used;;
10 %put #;
11 %put #   TABNUM           = &tabnum.;
12 %put #   AECAT           = &aecat.;
13 %put #=====;
14 %put;
15
16 /* Macro magic happens here for the next 300 lines or so... */
17
18 %mend createAETable;
19
20
```

So we're going to work towards this...

```
readmappingrules.sas *
1 %macro readxls(dir=, file=, sheet=, outds= );
2
3 %logPrintMacroStart;
4
5 /* Macro magic happens here for the next 100 lines or so... */
6
7 %mend readxls;
8
9
10 %readxls(dir=X:\ONCO\PH-2025-XYZ\Metadata
11         ,file=mapping.xlsx
12         ,sheet=MappingRules
13         ,outds=work.mapping);
14
```

```
Log - (Untitled)
#=====
# This is the start of macro READXLS.
#=====
# Stack trace:
# -OPEN CODE
# -READXLS
#=====
# Parameters used:
#
# DIR = X:\ONCO\PH-2025-XYZ\Metadata
# FILE = mapping.xlsx
# OUTDS = work.mapping
# SHEET = MappingRules
#=====
```





SASHELP.VMACRO for macro parameters

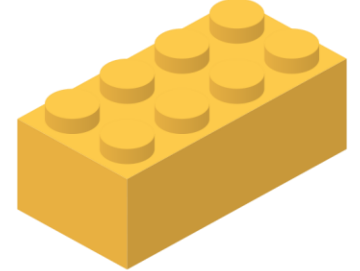
Combining yellow + red = generate full stack trace

...yes, I'm aware yellow + red does not give green

%SYSMEXECDEPTH obtains the nesting depth of macros

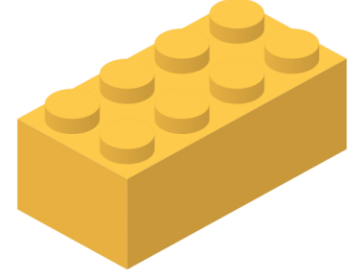
%SYSMEXECNAME obtains the name of our macros

%SYSMEEXECNAME — returns the macro name

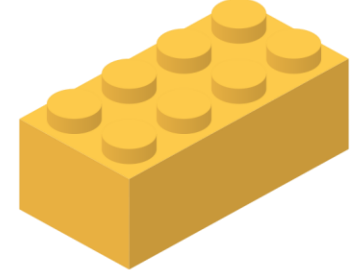


```
%macro readxls;  
    %put %sysmexecname(0);  
%mend;  
%readxls;
```

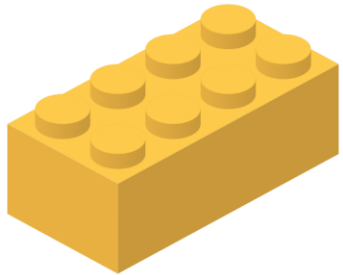
%SYSMEEXECNAME — returns the macro name of any given nesting level



So what does that give us..?

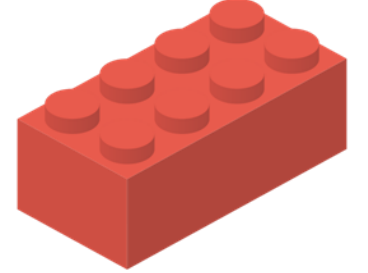


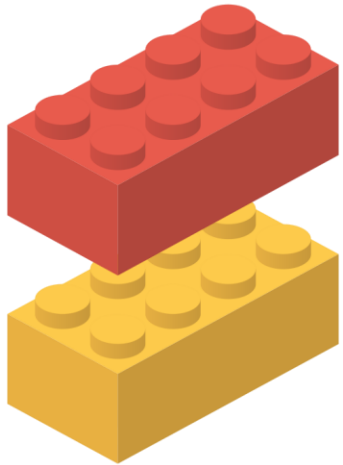
```
readmappingrules.sas *
1 %macro readxls(dir= ,file= ,sheet= ,outds= );
2
3 /*-----
4    Output macro parameters.
5    -----*/
6 %put #=====;
7 %put # This is the start of macro READXLS;
8 %put #=====;
9 %put # Parameters used;;
10 %put #;
11 %put #   DIR           = &dir.;
12 %put #   FILE          = &file.;
13 %put #   SHEET         = &sheet.;
14 %put #   OUTDS         = &outds.;
15 %put #=====;
16 %put;
17
18 /* Macro magic happens here for the next 100 lines or so... */
19
20 %mend readxls;
21
```



`%SYSMEEXECNAME`: Provides the name of a macro at nesting level X

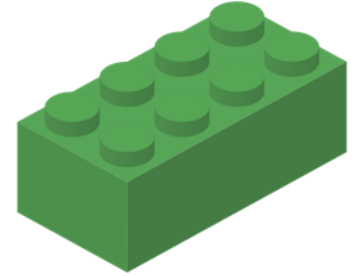
%SYSMEEXECDEPTH — provides the nesting depth of a macro



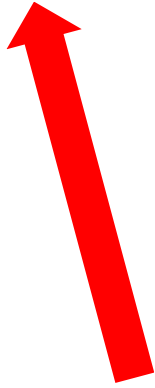
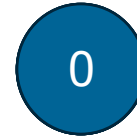


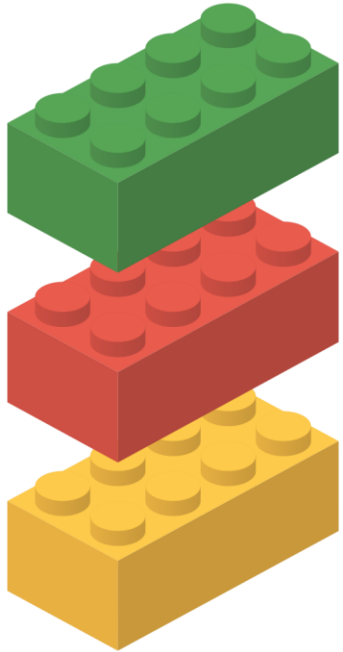
`%SYSMECDEPTH`: Provides the nesting depth of the current macro, providing X

`%SYSMECNAME`: Provides the name of a macro at nesting level X



%sysmexecdepth = 4

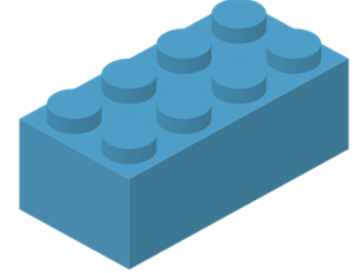




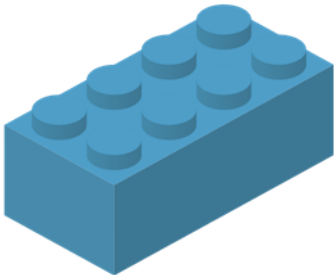
Yellow + red = full stack trace

`SYSMEXECDEPTH`: Provides the nesting depth of the current macro, providing X

`SYSMEXECNAME`: Provides the name of a macro at nesting level X



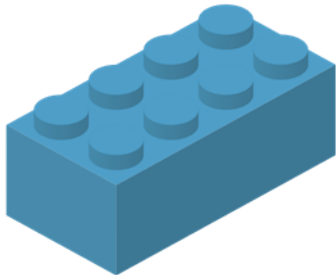
```
#=====
#  This is the start of macro READXLS.
#=====
#  Parameters used:
#
#    DIR           = X:\ONCO\PH-2025-XYZ\Metadata
#    FILE          = mapping.xlsx
#    OUTDS         = work.mapping
#    SHEET         = MappingRules
#=====
```



VIEWTABLE: Work.Vmacro				
	scope	name	offset	value
1	READXLS	DIR	0	X:\ONCO\PH-2025-XYZ\Metadata
2	READXLS	FILE	0	mapping.xlsx
3	READXLS	OUTDS	0	work.mapping
4	READXLS	SHEET	0	MappingRules
5	AUTOMATIC	AFDSID	0	0
6	AUTOMATIC	AFDSNAME	0	
7	AUTOMATIC	AFLIB	0	
8	AUTOMATIC	AFSTR1	0	
9	AUTOMATIC	AFSTR2	0	
10	AUTOMATIC	FSPBDV	0	
11	AUTOMATIC	CYSADDRBITE	0	64

	scope	name	offset	value
1	READXLS	DIR	0	X:\ONCO\PH-2025-XYZ\Metadata
2	READXLS	FILE	0	mapping.xlsx
3	READXLS	OUTDS	0	work.mapping
4	READXLS	SHEET	0	MappingRules

21	AUTOMATIC	SYSDMG	0	0
22	AUTOMATIC	SYSDSN	0	_NULL_
23	AUTOMATIC	SYSENCODING	0	wlatin1
24	AUTOMATIC	SYSENDIAN	0	LITTLE
25	AUTOMATIC	SYSENV	0	FORE
26	AUTOMATIC	SYSERR	0	0
27	AUTOMATIC	SYSERRORTEXT	0	
28	AUTOMATIC	SYSFILRC	0	0
29	AUTOMATIC	SYSHOSTINFOLONG	0	X64_10PRO WIN 10.0.26100 Workstation
30	AUTOMATIC	SYSHOSTNAME	0	OCSNL-Laptop122



1	READXLS
2	READXLS
3	READXLS
4	READXLS

data
se
wh
pu
run;

```
%let dsid=%sysfunc(open(sashelp.vmacro));
%let rc=%sysfunc(fetch(&dsid.));

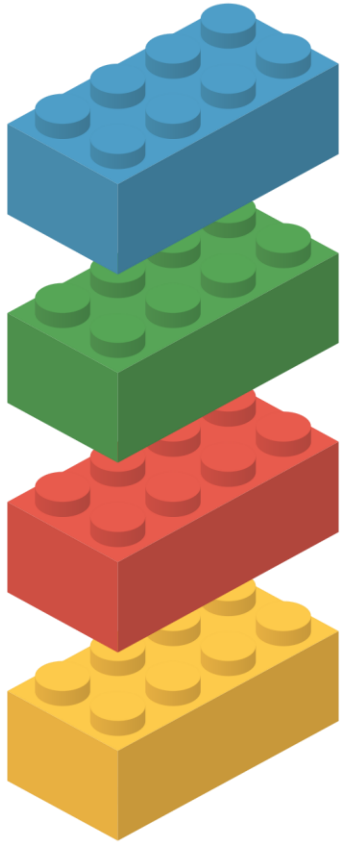
/* Loop through all records in SASHELP.VMACRO and print the parameter
   name and value if it belongs to the calling macro. */
%do %while(&rc=0);
  %let name_ds=%sysfunc(getvarc(&dsid., 1));
  %if %lowercase(&name_ds.) = %lowercase(&name.) %then %do;
    %let mvar=%sysfunc(getvarc(&dsid., 2));
    %let mvalue=%sysfunc(getvarc(&dsid., 4));
    %put %str(&char2. ) &mvar. %sysfunc(repeat(%str( ),%eval(20-%length(&mvar.)))) = &mvalue.;
  %end;
  %let rc=%sysfunc(fetch(&dsid.));
%end;

%let qid=%sysfunc(close(&dsid.));
```

value
X:\ONCO\PH-2025-XYZ\Metadata



#	DIR	= X:\ONCO\PH-2025-XYZ\Metadata
#	FILE	= mapping.xlsx
#	OUTDS	= work.mapping
#	SHEET	= MappingRules

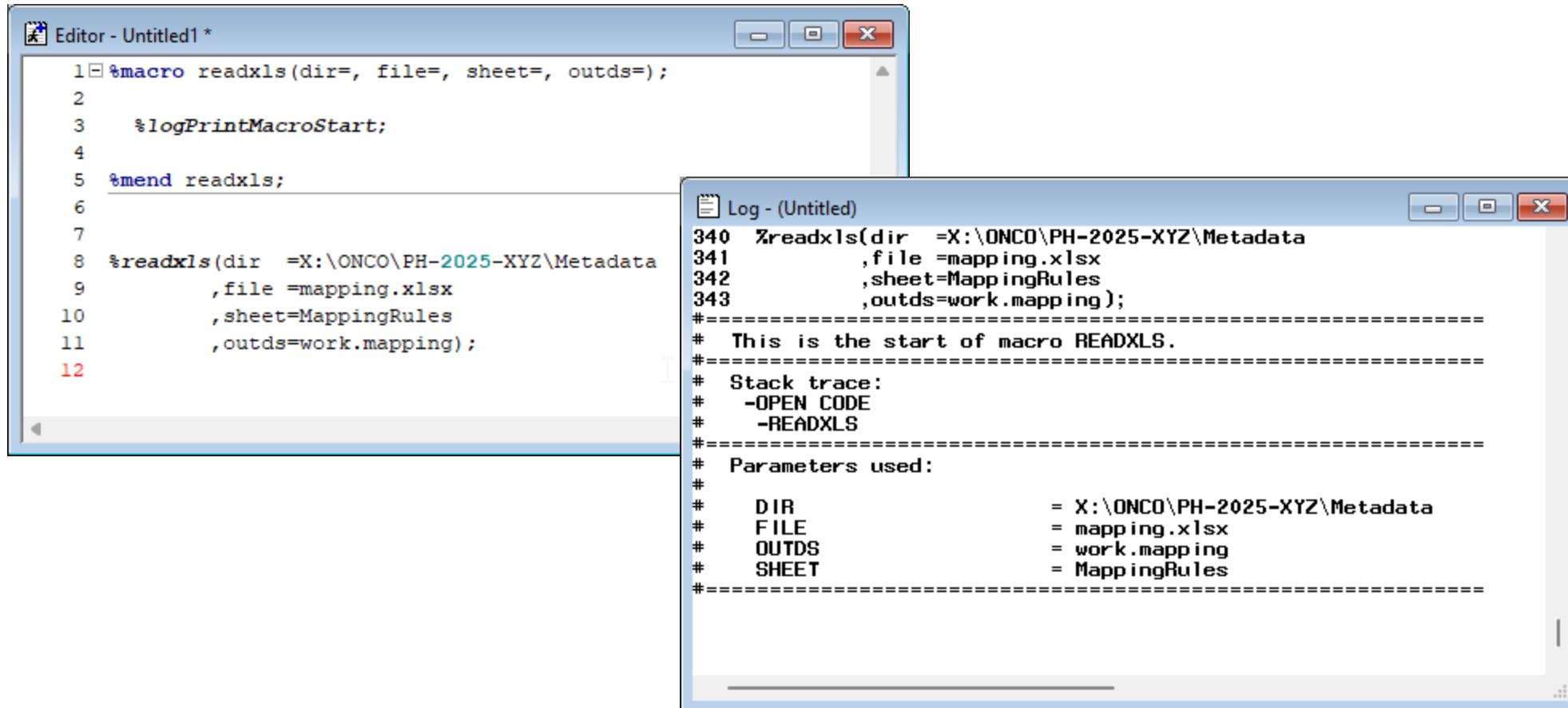


Parameters from SASHELP.VMACRO

Yellow + red = full stack trace

%SYSMEXECDEPTH: Provides the nesting depth of the current macro, providing X

%SYSMEXECNAME: Provides the name of a macro at nesting level X

The image shows two overlapping windows from a software application. The background window is titled 'Editor - Untitled1 *' and contains a macro definition for 'readxls'. The foreground window is titled 'Log - (Untitled)' and displays the execution log for the macro, including the call parameters and a stack trace.

```
1 %macro readxls(dir=, file=, sheet=, outds=);  
2  
3     %logPrintMacroStart;  
4  
5 %mend readxls;  
6  
7  
8 %readxls(dir =X:\ONCO\PH-2025-XYZ\Metadata  
9         ,file =mapping.xlsx  
10        ,sheet=MappingRules  
11        ,outds=work.mapping);  
12
```

```
340 %readxls(dir =X:\ONCO\PH-2025-XYZ\Metadata  
341         ,file =mapping.xlsx  
342         ,sheet=MappingRules  
343         ,outds=work.mapping);  
#-----  
# This is the start of macro READXLS.  
#-----  
# Stack trace:  
# -OPEN CODE  
# -READXLS  
#-----  
# Parameters used:  
#  
# DIR = X:\ONCO\PH-2025-XYZ\Metadata  
# FILE = mapping.xlsx  
# OUTDS = work.mapping  
# SHEET = MappingRules  
#-----
```

- Hallucinates
 - Dreams up non-existing SAS syntax
 - &sysmexecname instead of %sysmexecname
 - Invalid parameters to existing functions (thinks it can get the parameters from %sysmexec)

- Overdoes it

- Claude Sonnet 3.7 generated >20,000 lines of code instead of <200

	scope	name	offset	value
1	READXLS	DIR	0	X:\ONCO\PH-2025-XYZ\Metadata
2	READXLS	FILE	0	mapping.xlsx
3	READXLS	OUTDS	0	work.mapping
4	READXLS	SHEET	0	MappingRules

- Doesn't assume position of variables in SASHELP.VMACRO

- My takeaway

- AI is an excellent sparring partner and perhaps even a junior programmer
 - For SAS not (yet) at the level where it is for Python, perhaps also R

- Never again write manual macro announcements to the log
- AI is not a real threat (yet) for SAS programmers
- Fully working macro available, free of charge

Jules van der Zalm
OCS Life Sciences
The Netherlands

jules.vanderzalm@ocs-consulting.com
<https://www.ocs-lifesciences.com>

Direct download link (Github)

