SAS Macro

SAS Training Courses

By

Amadeus Software Ltd

AMADEUS SOFTWARE LIMITED SAS TRAINING

Amadeus have been delivering SAS Training since 1989 and our aim is to provide you with best quality SAS training where, when and how you choose.

Delivery Options Available:

- Public Training
- On-Site Training
- On-Demand Training

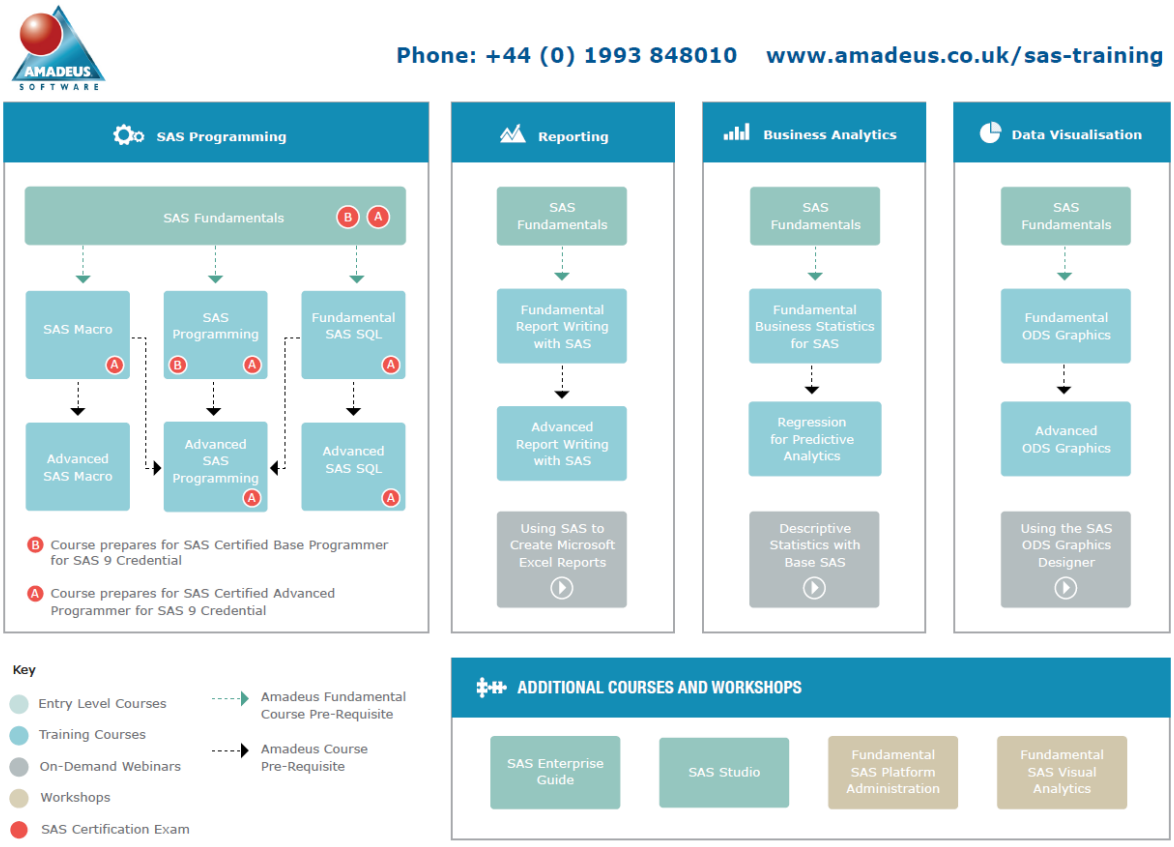
Types of Training Available:

Training Courses: Classroom based with a mixture of demonstration sessions and workshop sessions. Each delegate receives a copy of course material which can be used as reference material after the class.

Workshops: Classroom based with practical sessions. There will be very little in the way of lecturing; delegates will be expected to actively contribute during the workshop. Written material will be provided to delegates, this material will not be a textbook, but a record of the practical steps followed during the workshop.

On-Demand: Amadeus On-Demand training is delivered through a web browser to any computer or device with internet access. Training is recorded and conducted by our most experienced instructors using in-depth examples, video demonstrations and animations.

RECOMMENDED TRAINING CURRICULUM PATH



CONTACT

Amadeus Software Limited
The Old School Hall, 11 Wesley Walk, Witney, Oxfordshire, OX28 6ZJ, England.
Email: info@amadeus.co.uk

Facebook: Amadeussoftwareltd | Youtube: AmadeusSoftware | LinkedIn: amadeus-software

Credits

This SAS Software training course was written by Staff of Amadeus Software Ltd.

Version 6.8

Copyright Notice

Copyright © 2013 - 2021 by Amadeus Software Ltd. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying or otherwise, without prior permission of the publisher, Amadeus Software Ltd, Witney, Oxfordshire, England.

Trademark Notice

The Amadeus logo is a trademark of Amadeus Software Ltd, Witney, Oxfordshire, England.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration

All other trademarks recognised.

Disclaimer

These course notes are intended to supplement documentation supplied by SAS Institute, and not to replace it.

Where SAS syntax is given here, it may not include all possible options. Similarly, lists of statements, functions, formats, macro variables and other SAS features given here are not intended to be comprehensive. For complete and up-to-date information, please refer to the SAS Institute documentation.

.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Table of Contents

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

M1 Introduction

Common Coding Tasks

General Strategy

Workshop Session

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Common Coding Tasks

Those with some experience in SAS DATA and procedure step programming may have come across several classic problems during the course of their work.

Example 1

How would you include the current date in the title of a report? In order to do so, you would have to be able to write a TITLE statement that could somehow extract the current date from the system clock.

Here is a first attempt at a variable title statement:

```
data _null_;
  nowdate=today();
  format nowdate date9.;
  title 'Job run on nowdate';
run;

proc print data=mdata.policies;
run;
title;
```

And not surprisingly, the output starts as:

Job run on nowdate

Obs	name	address1	address2	address3	address4	premium	polnum	key	date	type
1	Mrs. Judy Spencer	102 Fairfield Drive	Norton Green	Norton	Cambs	78.50	107-5374-B-5Y	107-5374-B-5Y15AUG88	15AUG88	Buildings
2	Mr. Henry Higgins	45 Orchard Way	Panshanger	Welwyn Garden City	Herts	145.90	107-4983-C-9Y	107-4983-C-9Y04JAN86	04JAN86	Contents
3	Ms. Steffi Graf	15-40	The Nettings	Wimbledon Common	London SW15	654.85	207-6342-L-33	207-6342-L-3316AUG67	16AUG67	Life
4	Mr. M. Gorbachov	The Red House	Red Square	Moscow	USSR	35432.90	999-8414-L-63	999-8414-L-6323SEP36	23SEP36	Life
5	Mr. Steve Davis	3 The Green	Little Chalk	Near Pocket	Avon	256.60	107-5423-M-3Z	107-5423-M-3Z23OCT83	23OCT83	Motor
6	Mr. N. Mansell	500 Brands Pits Road	Near Hatching	Kent		15993.20	427-9263-M-39	427-9263-M-3903FEB50	03FEB50	Motor

Not very successful!

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

The macro solution:

```
title "Job run on &sysdate9";
proc print data=mdata.policies;
run;
title;
```

Job run on 22MAR2021

Obs	name	address1	address2	address3	address4	premium	polnum	key	date	type
1	Mrs. Judy Spencer	102 Fairfield Drive	Norton Green	Norton	Cambs	78.50	107-5374-B-5Y	107-5374-B-5Y15AUG88	15AUG88	Buildings
2	Mr. Henry Higgins	45 Orchard Way	Panshanger	Welwyn Garden City	Herts	145.90	107-4983-C-9Y	107-4983-C-9Y04JAN86	04JAN86	Contents
3	Ms. Steffi Graf	15-40	The Nettings	Wimbledon Common	London SW15	654.85	207-6342-L-33	207-6342-L-3316AUG67	16AUG67	Life
4	Mr. M. Gorbachov	The Red House	Red Square	Moscow	USSR	35432.90	999-8414-L-63	999-8414-L-6323SEP36	23SEP36	Life
5	Mr. Steve Davis	3 The Green	Little Chalk	Near Pocket	Avon	256.60	107-5423-M-3Z	107-5423-M-3Z23OCT83	23OCT83	Motor
6	Mr. N. Mansell	500 Brands Pits Road	Near Hatching	Kent		15993.20	427-9263-M-39	427-9263-M-3903FEB50	03FEB50	Motor

You can use several ‘automatic’ macro variables. ‘Automatic’ implies that the macro variable is available to you when you are running SAS programs.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Example 2

As a second example, suppose you want to execute some code conditionally, based upon the day of the week. You may want to execute different steps dependent upon the results of tests. The code below will NOT work because step types are being mixed:

```
data _null_;
    if day(today()) = 6 then do;  /*  6 = Friday  */
       title 'Print of Life Policies';
       proc print data=mdata.policies;
       run;
       title;
    end;
    else do;
       title 'Means of Motor Policies';
       proc means data=mdata.policies;
       run;
       title;
    end;
run;
```

```
33
34          data _null_;
35           if day(today()) = 6 then do;  /*  6 = Friday  */
36              title 'Print of Life Policies';
37

NOTE: The SAS System stopped processing this step because of errors.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.03 seconds



37       !     proc print data=mdata.policies;
                _
                117
ERROR 117-185: There was 1 unclosed DO block.
38              run;

NOTE: There were 6 observations read from the data set MDATA.POLICIES.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.02 seconds
      cpu time            0.03 seconds


39              title;
40           end;
             ___
             180

ERROR 180-322: Statement is not valid or it is used out of proper order.
```


No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

```
41          else do;
            ____
            180

ERROR 180-322: Statement is not valid or it is used out of proper order.

42             title 'Means of Motor Policies';
43
43       !     proc means data=mdata.policies;
44             run;

NOTE: There were 6 observations read from the data set MDATA.POLICIES.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.05 seconds
      cpu time            0.01 seconds


45             title;
46          end;
            ___
            180

ERROR 180-322: Statement is not valid or it is used out of proper order.
```

You cannot mix DATA steps and PROC steps in a SAS program. They have to be separate and only the macro language can pass values from one step to another.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

For example, this solution uses a 'SAS Macro' to provide the logic to determine which PROC step to run. Notice there is no DATA step being used here:

```
%macro policies(day);
    %if &sysday = &day %then %do;
       title "Print of Life Policies (&sysday Report)";
       proc print data=mdata.policies;
       run;
       title;
    %end;
    %else %do;
       title "Means of Motor Policies (&sysday Report)";
       proc means data=mdata.policies;
       run;
       title;
    %end;
%mend;

%policies(Friday)
```

The results of the program, run on a Monday:

Means of Motor Policies (Monday Report)

The MEANS Procedure

Analysis Variable : premium				
N	Mean	Std Dev	Minimum	Maximum
6	8760.32	14500.59	78.5000000	35432.90

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Example 3

In this third example, suppose you have a monthly series of twelve SAS data sets all to be run with the same report. In this case our 'wrong' attempt may look something like:

```
data _null_;
  do i=1 to 12;
    proc tabulate data=mdata.ds_i;
     class status gender;
     var salary;
     table status,gender*salary;
    run;
  end;
run;
```

Here we are assuming our twelve SAS data sets are called MDATA.DS_1 to MDATA.DS_12. This again, will not work because of the attempt to mix steps. We could write 12 individual Proc TABULATE steps however.

This alternative solution creates a ‘bundle of code’ called a ‘SAS Macro’ which will pass a different value to the Proc TABULATE code each time round the loop.

```
%macro table;
  %do i=1 %to 12;
    proc tabulate data=mdata.ds_&i;
     class status gender;
     var salary;
     table status,gender*salary;
    run;
  %end;
%mend;

%table
```


No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Example 4

The fourth example again shows data dependency.

Suppose we wish to print a SAS data set and in the title of the report insert the average value of one of the variables.

The data set used in the next example is the EPIDEMIC data set. This contains the number of influenza cases reported per week for five age ranges of the population (categories A to E). We wish to print the data set and include the average number of CASES for category 'A' within the title of the report.

We first need to calculate the average CASES, but then how to we include this calculated value into the report title?

The scenario:

```
proc means data=mdata.epidemic(where=(category='A')) noprint;
 var cases;
 output out=work.avcases mean=avnum;
run;

/*  How to we get the AVNUM variable value into a title ? */

title "Average number of cases for Category A is AVNUM";
proc print data=mdata.epidemic (where=(category='A'));
run;
title;
```

This program will not include the AVNUM variable in the report title as this is just part of a literal string.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

The macro language again can offer us a solution by using a CALL SYMPUTX statement in a DATA step to create a macro variable from a data set variable:

```
proc means data=mdata.epidemic(where=(category='A')) noprint;
 var cases;
 output out=work.avcases mean=avnum;
run;

data _null_;
   set work.avcases;
   call symputx('average',avnum);
run;

title "Average number of cases for Category A is &average";
proc print data=mdata.epidemic (where=(category='A'));
run;
title;
```

The output report:

Average number of cases for Category A is 85.380952381

Obs	week_no	date	category	cases
1	1	10838	A	10
2	2	10845	A	21
3	3	10852	A	25
4	4	10859	A	32
5	5	10866	A	36
6	6	10873	A	45
7	7	10880	A	56
8	8	10887	A	86
9	9	10894	A	120
10	10	10901	A	140
11	11	10908	A	168
12	12	10915	A	190
13	13	10922	A	224
14	14	10929	A	180
15	15	10936	A	128
16	16	10943	A	114
17	17	10957	A	85
18	18	10964	A	63
19	19	10971	A	35
20	20	10978	A	23
21	21	10985	A	12

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

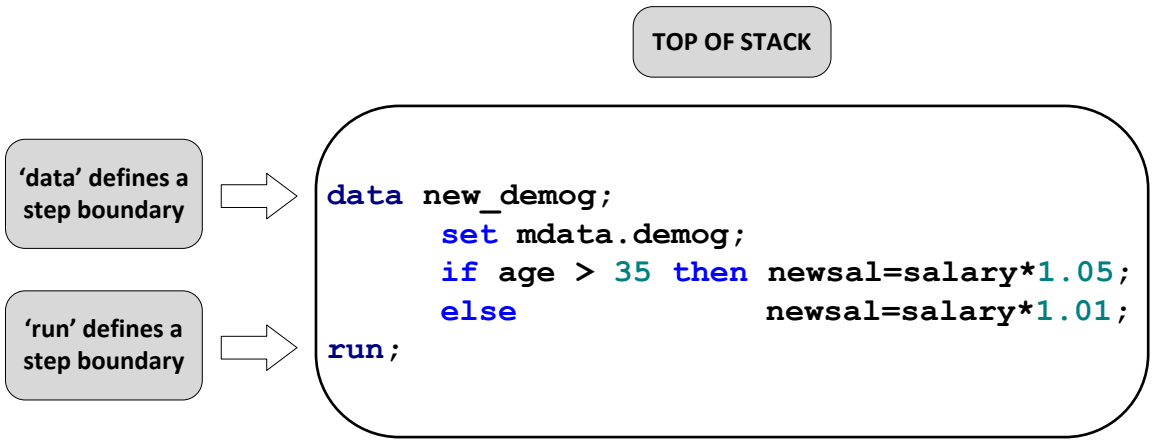
General Strategy

By now you will have noticed the key symbols within the macro language are the ampersand (&) and the percent sign (%).

Before we look at the effect of these two symbols, let us consider the progress of a normal DATA step program.

```
data new_demog;
     set mdata.demog;
     if age > 35 then newsal=salary*1.05;
     else             newsal=salary*1.01;
run;
```

This program has one step; a DATA step. The SAS Supervisor places all of these statements in a program input stack, takes one word at a time from the stack and passes them to the SAS compiler or procedure parser (if the step is a proc). At the end of the statement when a semi-colon is reached the statement is checked for syntax. At a step boundary, this process is halted and the step is compiled or parsed, then executed.



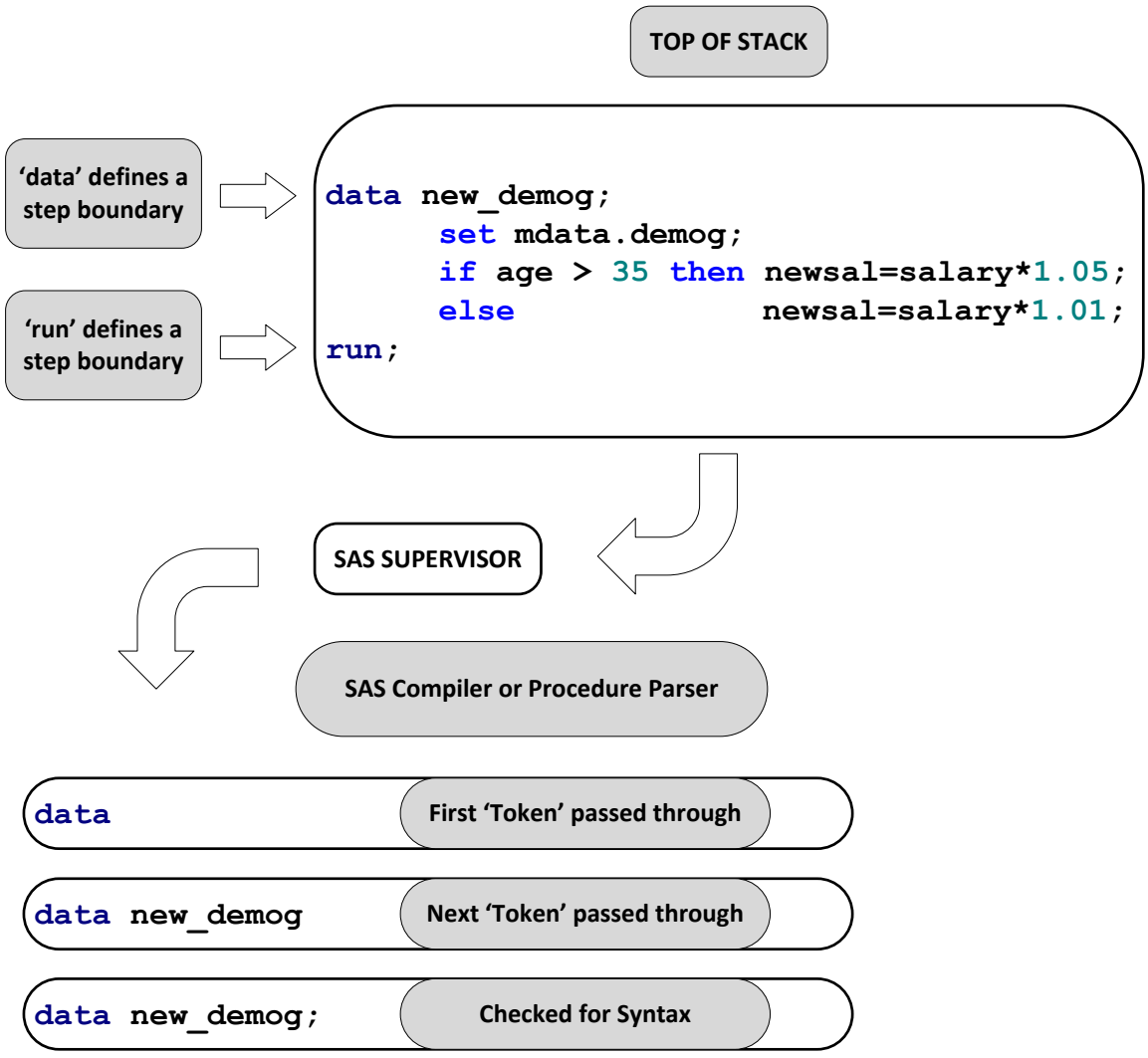
No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

A SAS program is simply a sequence of DATA and PROC steps

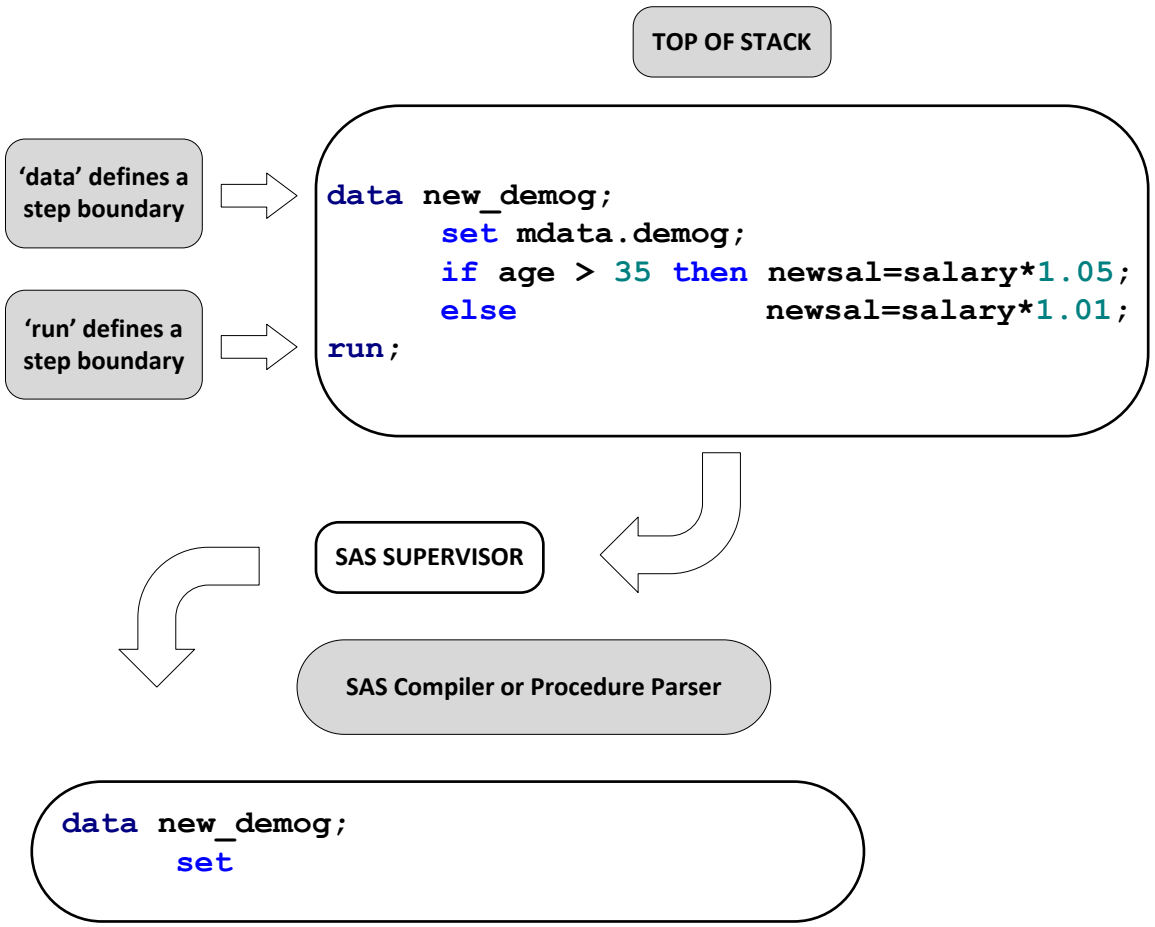
Each step is delimited by the keywords such as:

DATA
PROC
RUN
QUIT
ENDSAS

A 'word' (or 'token') at a time is transferred to the DATA step compiler or procedure parser.



No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.



…and so on, until the RUN; step boundary is found.

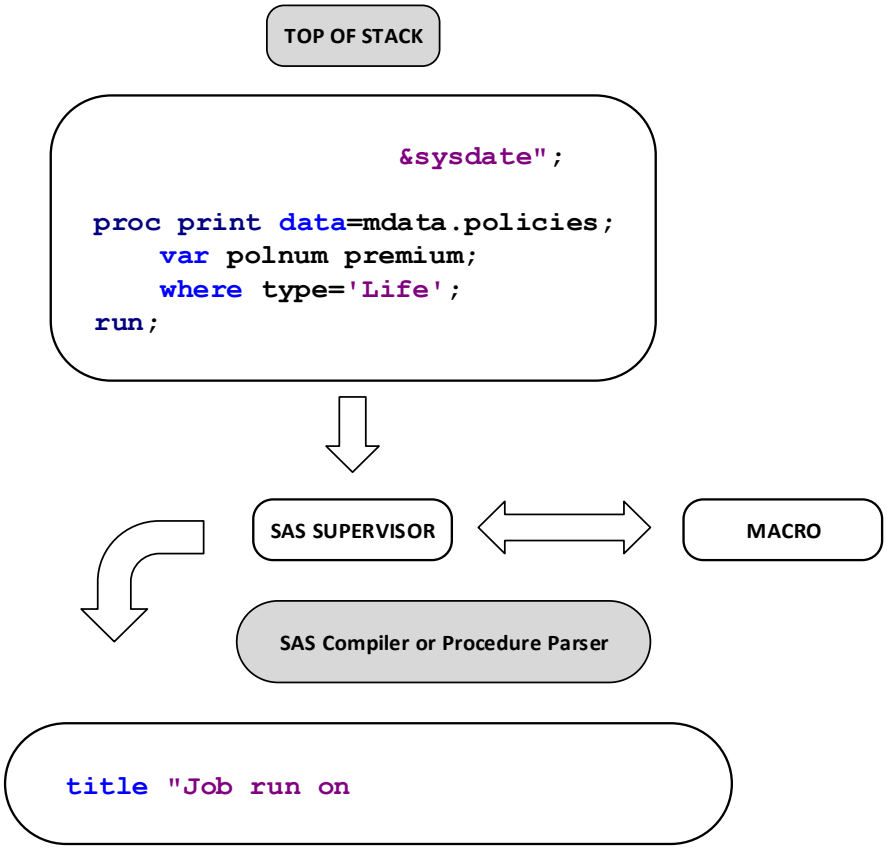
Upon a step boundary, the SAS Supervisor suspends processing the SAS statements and the completed step is compiled (or parsed in the case of a procedure) and then executed.

Each step independently compiles (parses) and then executes.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

A SAS program is, and can only be, a series of DATA and PROC steps plus any global statements. The role of Macro is simply to generate code that is to be put on top of the program input stack.

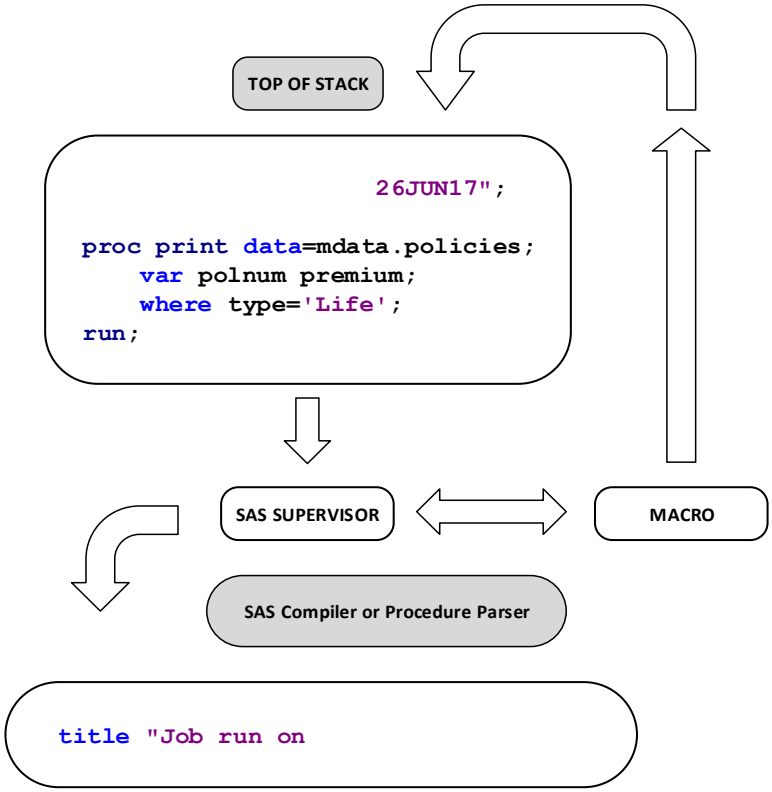
```
title "Job run on &sysdate";

proc print data=mdata.policies;
     var polnum premium;
     where type='Life';
run;

title;
```



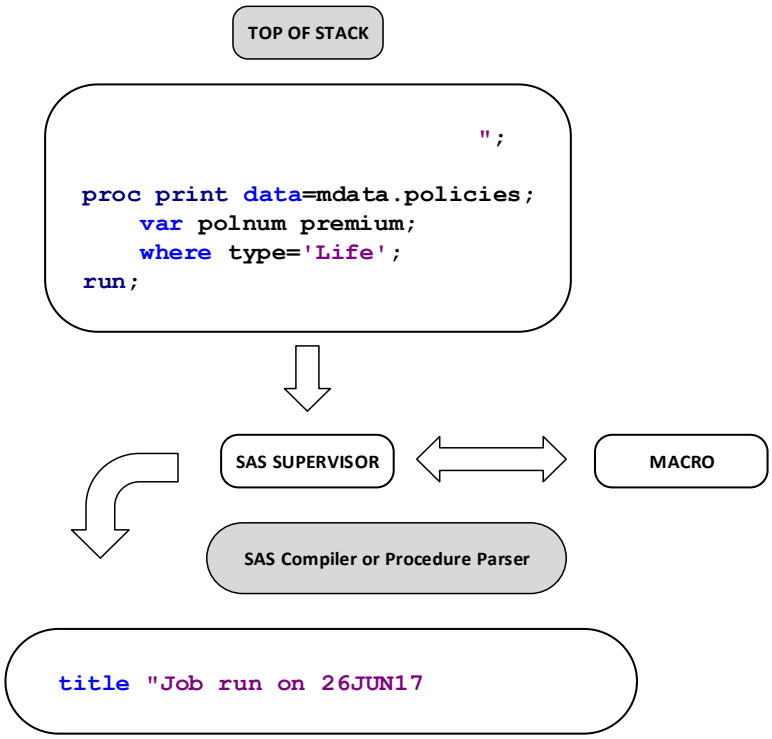
The presence of the '&' followed by a non-blank character triggers the macro facility. The token is passed to the macro facility, which substitutes a value on the input stack.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

The value of the macro variable is retrieved by the macro facility and placed on the input stack:



Whereupon the replaced code is treated as normal:



No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

So, the SAS macro facility is concerned with extending and customising the SAS language. It gives users the ability to:

- Generate code dynamically;
- Handle text string substitution within code;
- Enable repeated execution of code bundles;
- Use conditional logic to execute code.

It should be stressed that the SAS macro facility will not enable SAS code to run quicker, but rather produce SAS programs that are potentially re-usable, dynamic and can be maintained more easily.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Workshop Session

Exercise 1

Create a SAS Data Library called MDATA which points to the course data. The instructor will let you know this location.

Exercise 2

The aim of exercises 2 – 4 is to recognise the benefits of using the SAS Macro Language.

Type in and submit the following code:

```
%let sas_course=Macro;

%put I am on the &sas_course course;

%put The date today is &sysdate;
```

What do you see in the log window?

Exercise 3

Type in and submit the following code. What do you see in the log?

```
data _null_;
    set mdata.demog nobs=num_obs end=e;
    total_salary + salary;
    if e then do;
        average_salary = total_salary / num_obs;
        call symputx('avsal',average_salary);
    end;
run;

%put The average salary is &avsal;
```


No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Exercise 4

Type in and submit the following code:

```
%macro summary (dataset=);
    proc means data=&dataset;
    run;
%mend;

%summary(dataset=mdata.demog);
%summary(dataset=mdata.bp1);
```

What output is produced in the output window?

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.

Exercise 5

The aim of this exercise is to identify a pattern of repetition when writing code which could be solved by a macro:

a) Using the DEMOG table in the course library, create a new data set called EMP001 that only contains the records relating to employee 001.

b) Modify your DATA step so that two data sets called EMP001 and EMP002 are created. Again each data set should only contain records relating to one employee. Your code should now look similar to the following:

```
data emp001 emp002;
  set mdata.demog;
       if staffno='001' then output emp001;
  else if staffno='002' then output emp002;
run;
```

c) Modify your DATA step again so that three data sets called EMP001, EMP002 and EMP003 are created. Similarly, each data set should only contain records relating to one employee.

What would your DATA step look like if you needed to do this for all 104 employees?

** Do NOT attempt to solve this with the macro language. The solution will be discussed later in the course. **

Notice what pattern is emerging within your code.

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software Ltd. All rights reserved.