INTRODUCTION TO THE SAS MACRO FACILITY

THE BASICS

PURPOSE OF THE MACRO FACILITY

The *macro facility* is a text processing facility for automating and customizing SAS code. The macro facility helps minimize the amount of SAS code you must type to perform common tasks.

PURPOSE OF THE MACRO FACILITY

The macro facility enables you to do the following:

- create and resolve **macro variables** anywhere within a SAS program
- write and call **macro programs** (**macro definitions** or **macros**) that generate custom SAS code

AUTOMATIC MACRO VARIABLES

Example: Include system values within a SAS program.

```
proc print data=health.claims_sample;
   title 'Claims Report';
   footnote "Created &systime, &sysday &sysdate9";
   footnote2 "on the &sysscp system using &sysver";
run;
```

USER-DEFINED MACRO VARIABLES

Example: Reference the same value repeatedly throughout a program.

```
proc print data=health.claims_sample;
   title "Report for 2006";
   var providerID memberID chgamt lndiscamt eeresp paidamt;
   where svc_year="2006";
   footnote;
run;
```

REPETITIVE PROCESSING

Example: Generate a similar report each year.

```
proc print data=health.year2006;
run;
```

```
proc print data=health.year2007;
run;
```

```
proc print data=health.year2008;
run;
```

POINTS TO REMEMBER

- The macro facility can reduce development time and maintenance time.
- Efficiency of underlying code should be considered.

MACRO VARIABLES

MACRO VARIABLES

- Dynamically modify text in a SAS program.
- Assign large or small amounts of text to macro variables, and use that text by referencing the variable that contains it.

DEMONSTRATION

- Using automatic macro variables.
- Creating and referencing user-defined macro variables.

MACRO FUNCTIONS

MACRO FUNCTIONS

Macro functions manipulate arguments within the context of the macro language.

The following are true for macro functions:

- manipulate macro variables and expressions
- may mimic the functionality of DATA step functions
- are executed by the macro processor

MACRO FUNCTIONS

Examples of functions with corresponding DATA step functionality include:

- %SUBSTR
- %UPCASE
- %INDEX
- %LENGTH

MACRO FUNCTIONS

Other functions are specific to the macro language and perform manipulations often needed at the macro level. Some examples include:

%SYSFUNC – executes SAS functions
%STR, %BQUOTE – quoting functions

COMPILATION AND EXECUTION

TIMING OF COMPILATION AND EXECUTION

Compiler

Macro Processor

Word Scanner

Input Stack

```
%let year=2006;
proc print data=health.claims_sample;
…
title "Report for &year";
   where svc_year="&year";
run;
```

Symbol Table	

TIMING OF COMPILATION AND EXECUTION

Compiler

Word Scanner

```
%let
```

Input Stack

```
year=2006;
proc print data=health.claims_sample;
…
title "Report for &year";
   where svc_year="&year";
run;
```

Macro Processor

Symbol Table

TIMING OF COMPILATION AND EXECUTION

Compiler

Word Scanner

Input Stack

```
proc print data=health.claims_sample;
…
title "Report for &year";
   where svc_year="&year";
run;
```

Macro Processor

```
%let year=2006;
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Macro Processor

Word Scanner

```
&year
```

Input Stack

```
";
   where svc_year="&year";
run;
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Word Scanner

Macro Processor

```
&year
```

Input Stack

```
";
   where svc_year="&year";
run;
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Macro Processor

Word Scanner

Input Stack

```
2006";
   where svc_year="&year";
run;
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for 2006";
where svc_year=“2006";
run;
```

Word Scanner

Input Stack

Macro Processor

Symbol Table	
year	2006

MACRO PROGRAMS

MACRO PROGRAMS

Macro programs require a 2 step process:

1. Define (or compile) the macro.
2. Call the macro.

DEFINING A MACRO

General form of a macro definition:

%MACRO *macro-name*;
macro-text
%MEND <*macro-name*>;

CALLING A MACRO

A *macro call*

- causes the macro to execute
- is specified by placing a percent sign before the name of the macro
- can be made anywhere in a program (similar to a macro variable reference)
- is **not** a statement (no semicolon required).

General form:

```
%macro-name
```

DEMONSTRATION

- Compile and call a macro program.
- Compile and call a macro program with a parameter.

SUPPORT.SAS.COM RESOURCES

- SAS® Macro Language Reference

http://support.sas.com/documentation/onlinedoc/base/index.html

- Papers & SAS Notes

http://www2.sas.com/proceedings/sugi30/130-30.pdf
http://www2.sas.com/proceedings/sugi30/237-30.pdf

- SAS Training

https://support.sas.com/edu/schedules.html?id=246&

SUPPORT.SAS.COM RESOURCES

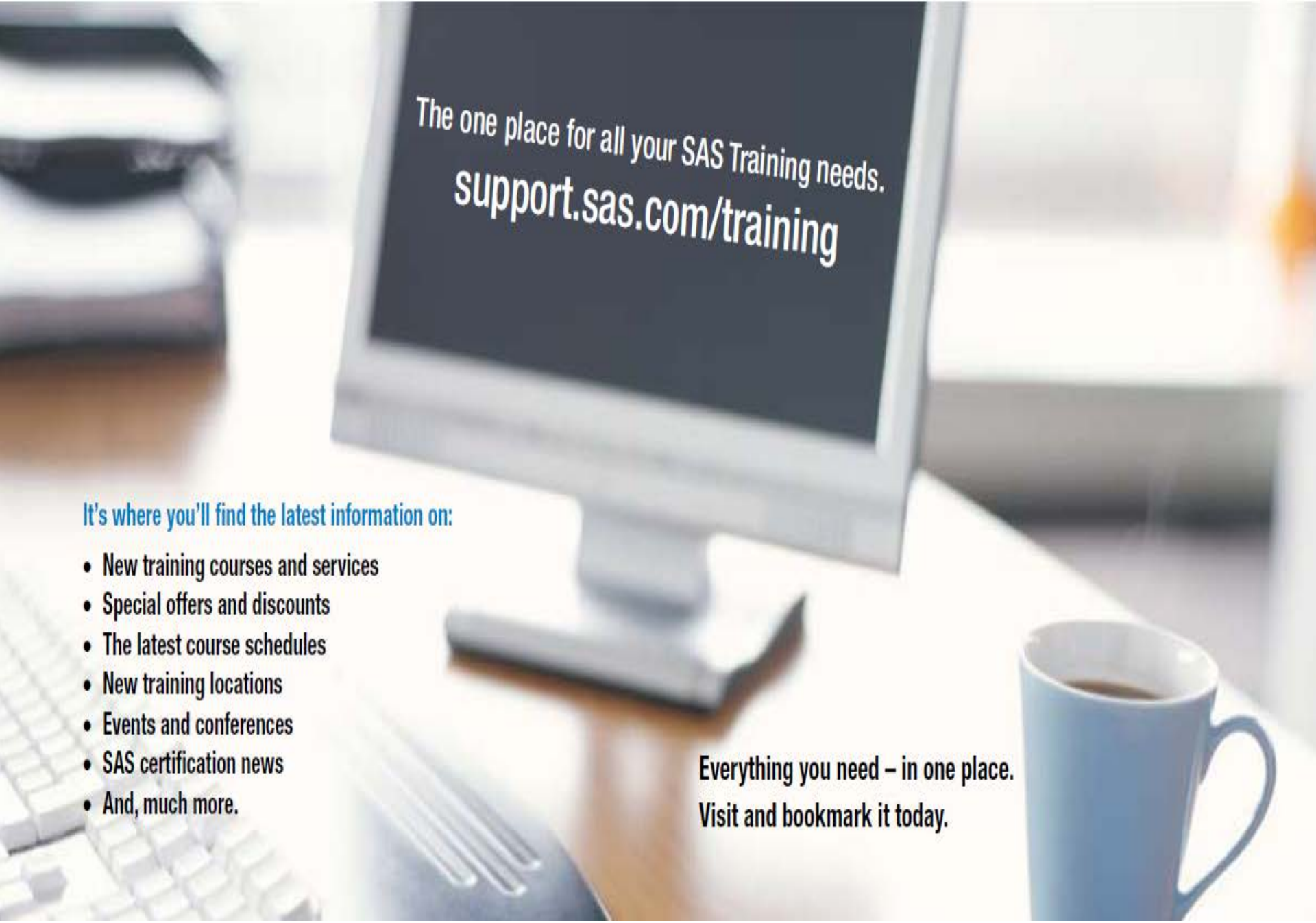
- RSS & Blogs

http://support.sas.com/community/rss/index.html

http://blogs.sas.com

- Discussion Forums

http://communities.sas.com/index.jspa



It's where you'll find the latest information on:

- New training courses and services
- Special offers and discounts
- The latest course schedules
- New training locations
- Events and conferences
- SAS certification news
- And, much more.

Everything you need – in one place.
Visit and bookmark it today.

THANK YOU FOR ATTENDING!

sas.com