An introduction to Proc DS2

Daniel.Ringqvist@sas.com

What is DS2?

What is DS2?

DS2 is a new Base SAS programming language

- It looks like the DATA step
- SQL is integrated
- Easily extensible
- Very clean and precise

Data step

```
data WORK.one_DS;
   length Zaphod $ 25;
   retain Zaphod 'President';
   set WORK.Cars;
run;
```

DS2 program

```
proc ds2;
   data WORK.one_DS2;
   declare char(25) Zaphod;
   retain Zaphod 'President';
   method run();
      set WORK.Cars;
   end;
   enddata;
run;quit;
```

DS2 Superpowers

DS2 natively supports ANSI SQL data types for precise data manipulation.

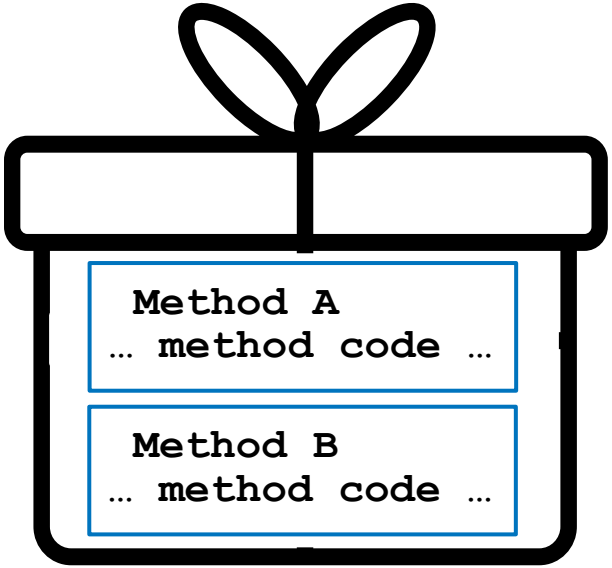
Examples:

Data Type	Examples
Floating Point Numeric	DOUBLE, FLOAT, REAL
Fixed Decimal Numeric	DECIMAL
Integer Numeric	BIGINT, INT, SMALLINT, TINYINT
Date and Time	DATE, TIME, TIMESTAMP
Character	CHAR, NCHAR, VARCHAR, NVARCHAR

SAS numeric variables equate to DOUBLE.
SAS character variables equate to CHAR.

DS2 Superpowers

DS2 improves the extensibility and reusability of code through the use of methods and packages, enabling object-oriented programming.

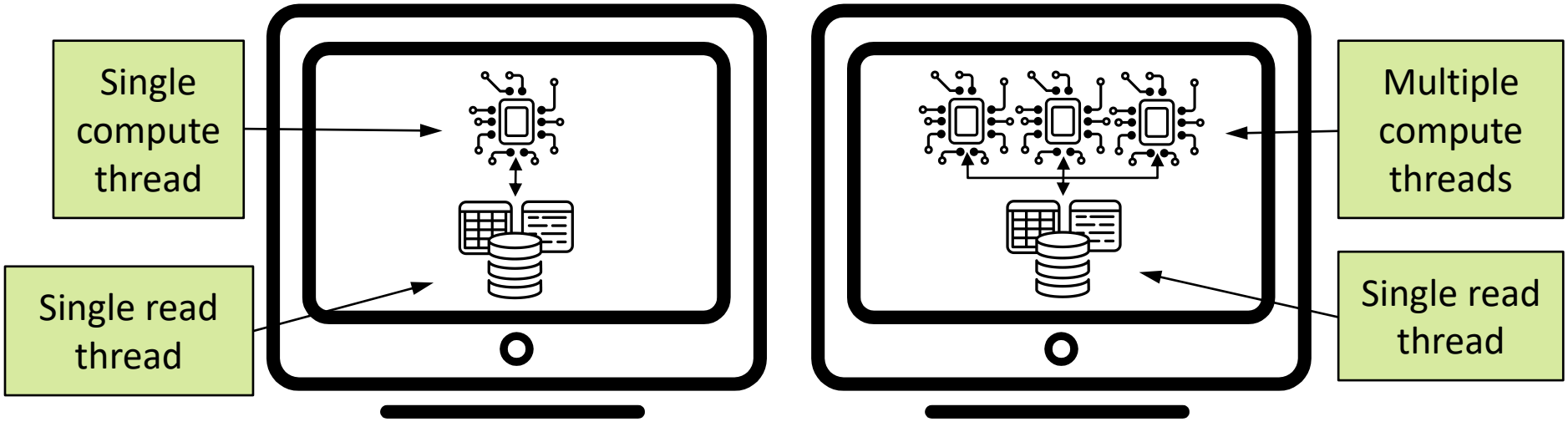


Methods and packages can be predefined or user-defined.

DS2 Superpowers

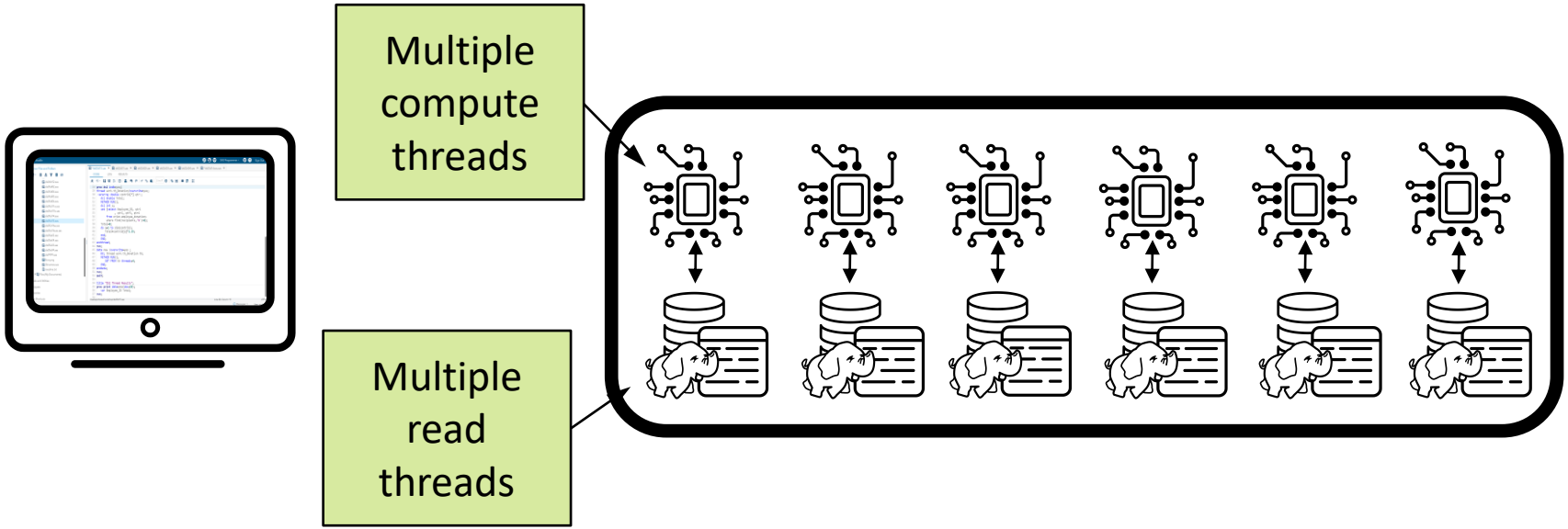
The Base SAS DATA step processes each observation sequentially.

DS2 can process observations in parallel on the SAS compute platform (*symmetric multiprocessing*, or *SMP*).



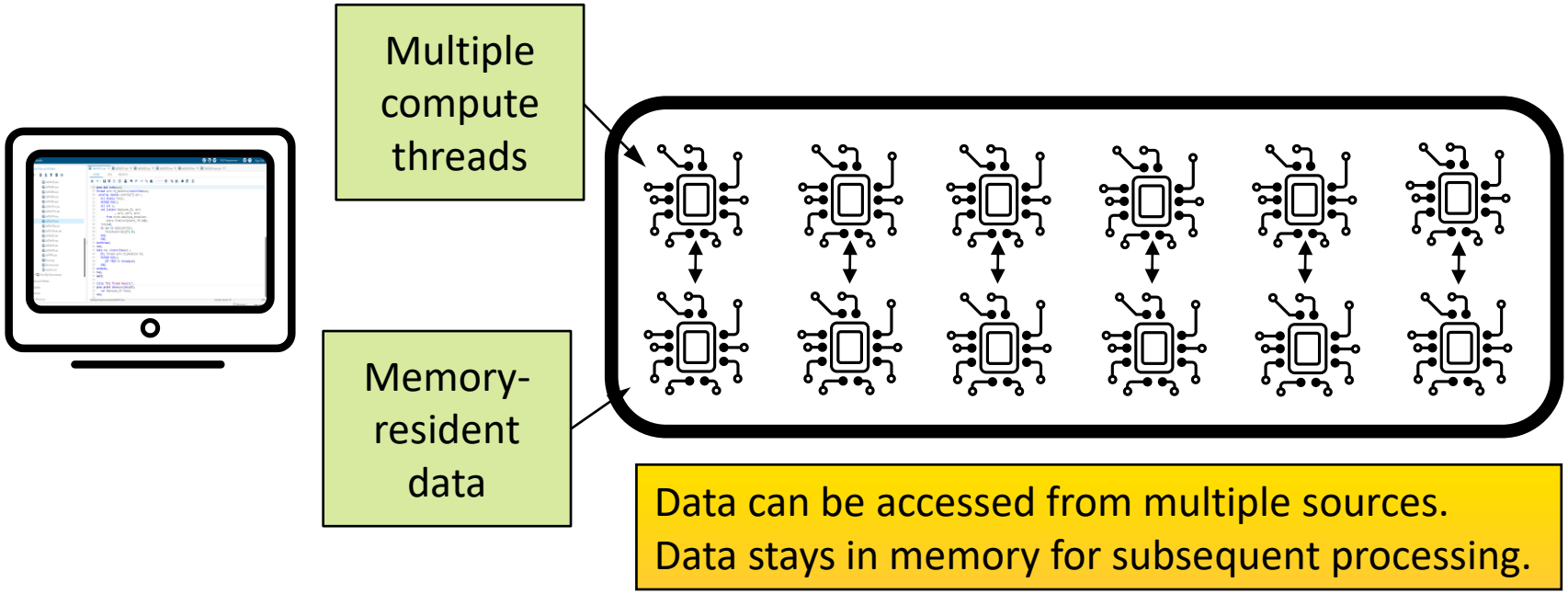
DS2 Superpowers

With the SAS In-Database Code Accelerator, DS2 threads can be executed natively, in parallel, in-database.



DS2 Superpowers

For SAS Viya, Cloud Analytic Services (CAS) provides native DS2 action sets, enabling DS2 programs to run natively in CAS.



DS2 Threaded Drivers

DS2 can access data from the following data sources:

- Amazon Redshift
- Aster
- CAS tables
- DB2 (UNIX/Windows)
- Greenplum
- Hadoop (Hive and HDMD)
- Impala
- JDBC-compliant databases
- Memory Data Store (MDS)
- Microsoft SQL Server
- MYSQL
- Netezza
- ODBC-compliant databases
- Oracle
- PostgreSQL
- SAP (Read only)
- SAP HANA
- SAP IQ
- SAS data
- SAS SPDE data
- SAS SPDS data (UNIX/Windows)
- Spark
- Teradata (UNIX/Windows)
- Vertica

When should I use DS2?

Write programs in DS2 when the application meets one or more of these criteria:

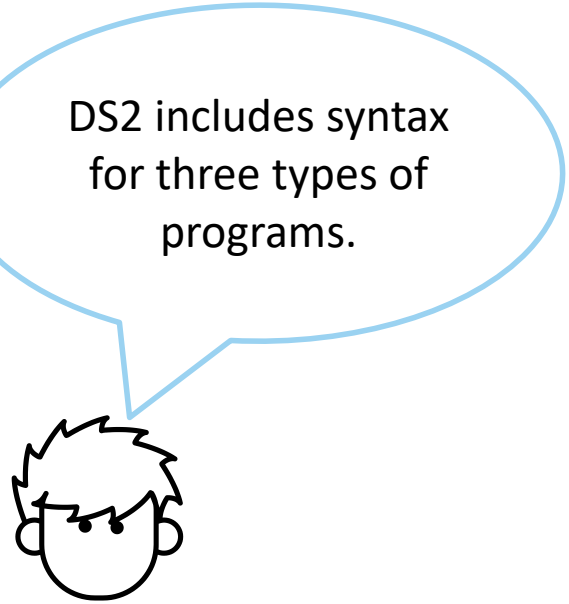
- The program can take advantage of threaded processing.
  - Base SAS DATA steps that are computationally complex (CPU bound)
  - read from and write to tables in a massively parallel processing (MPP) database where the SAS In-Database Code Accelerator installed
  - read from and write to caslibs
- You require the precision that the DS2 data types offer.
- You want to leverage the code reusability of DS2 packages.

DS2 syntax

Basic DS2 Syntax

```
proc ds2;
package work.pgk;
   <more program statements>
endpackage;
run;
thread work.thread;
   <more program statements>
endthread;
run;
data _null_;
   <more program statements>
enddata;
run;
quit;
```

PROC DS2 uses RUN-group processing.



Basic DS2 Syntax

The DS2 DATA program

- begins with a DATA statement
- ends with an ENDDATA statement
- requires a RUN statement to execute.

```
proc ds2;
data _null_;
   method init();
      Text='Hello, World!';
      put Text=;
   end;
enddata;
run;
quit;
```

Basic DS2 Syntax

New in DS2: Methods

- *Methods* are named, executable blocks of code.
  - The METHOD statement names the method.
  - The END statement terminates the method.
- All executable code must be encapsulated in a method.

```
proc ds2;
data _null_;
   method init();
      Text='Hello, World!';
      put Text=;
   end;
enddata;
run;
quit;
```

Basic DS2 Syntax

System methods execute automatically.

- INIT() – once at start
- RUN() – once for every row
- TERM() – once at termination

The RUN method includes an implicit OUTPUT at END.

```
data _null_;
   method init();
      dcl varchar(20) Text;
      Text='**> Starting';
      put Text;
   end;
   method run();
      set ds2_sas.banks;
      put _all_;
   end;
   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
```

DS2 DATA Program Flow

INIT

- Starts the data program execution
- Executes each statement once
- At END, calls the RUN method

RUN

- Starts only when called by INIT
- At END, implicit OUTPUT and RETURN
- Loops until input data is exhausted
- When out of data, calls TERM

TERM

- Starts only when called by RUN
- Executes each statement once
- At END, terminates data program processing

When no code is provided for a system method, the compiler inserts a null method on your behalf.

User-defined methods

User-defined methods have the following features:

- can accept arguments
- can return a value
- execute when referenced
- can be referenced multiple times

```
proc ds2;
data _null_;
   method c2f(double Tc) returns double;
   /* Celsius to Fahrenheit */
      return (((Tc*9)/5)+32);
   end;
   method init();
      dcl double Degc DegF;
      do DegC=0 to 30 by 15;
         DegF=c2f(DegC);
         PUT DegC= DegF=;
      end;
   end;
enddata;
run;
quit;
```

Method is defined.

Method is called.

Partial SAS Log

```
degc=0  degf=32
degc=15 degf=59
degc=30 degf=86
```

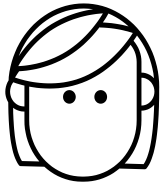
Basic DS2 Syntax

Declarative statements come before executable statements.

- ***Global*** declarative statements affect the entire data program and must be placed before any method definition code.
- ***Local*** declarative statements affect only the method in which they appear and must be placed before any executable statements.

```
proc ds2;
data test;
   dcl double MyVar;
   retain MyVar 0;
   method run();
      dcl integer i;
      set ds2_sas.banks;
      do i=1 to 3;
         MyVar=10**i;
      end;
   end;
enddata;
run;
quit;
```

DECLARE is the only declarative statement permitted inside of a method definition.



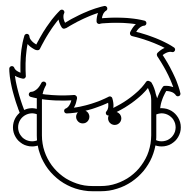
Basic DS2 Syntax

The DECLARE statement

- declares variables or temporary arrays
- assigns type, length, and other attributes.

DECLARE|DCL *data-type variable-list*
<HAVING LABEL '*string*' | FORMAT | INFORMAT>**;**

Where a variable is declared determines its scope.



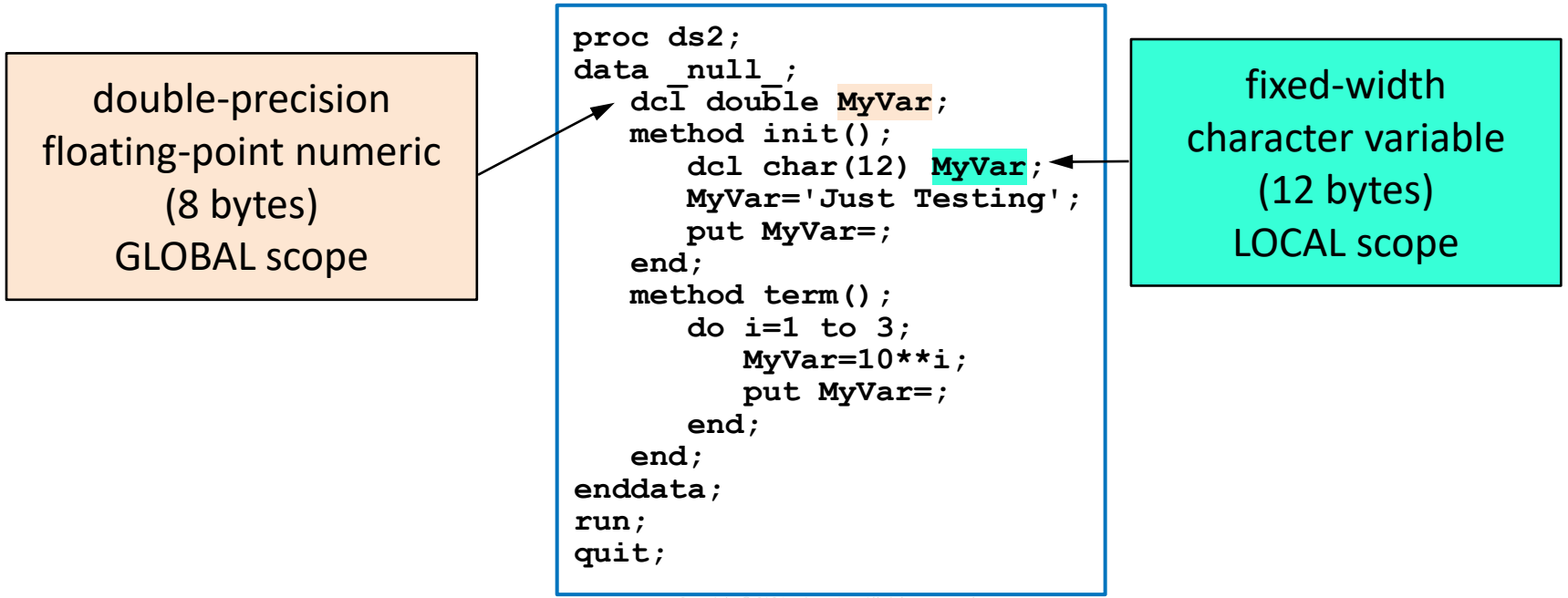
Examples:

```
/* double precision floating point variables, formatted dollar12.2*/
dcl double Var1 Var2 Var3 having format dollar12.2;

/* high-precision fixed point numeric variable */
dcl decimal(35,5) Var1;

/* fixed-width character variable labeled 'My Text'*/
dcl char(25) Var1 having label 'My Text';
```

Basic DS2 Syntax

The DECLARE statement

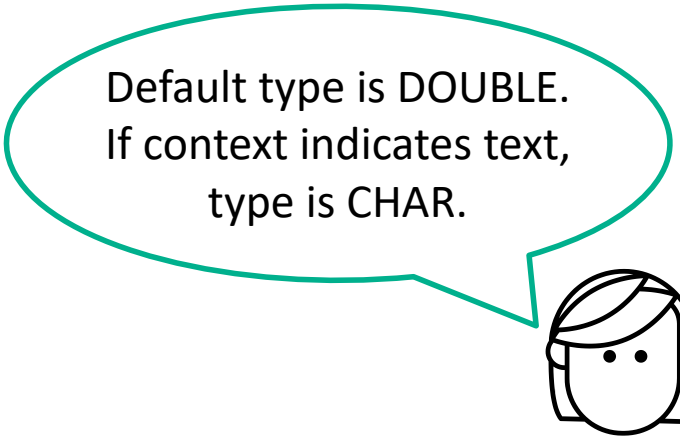
- declares variables or temporary arrays
- assigns type, length, and other attributes.



Basic DS2 Syntax

Undeclared Variables

- Neither read in via SET statement nor defined in a DECLARE statement.
- Global scope
- Produce warnings in the SAS log.



Undeclared, GLOBAL scope

```
proc ds2;
data _null_;
   dcl double MyVar;
   method term();
      do i=1 to 3;
         MyVar=10**i;
         put MyVar=;
      end;
   end;
enddata;
run;
quit;
```

Basic DS2 Syntax

Variable Scope

- Globally declared and undeclared variables have a global scope and are available anywhere in the DATA program.

Global: Available anywhere in the DATA program

```
data _null_;
   dcl double MyVar;
   method init();
      dcl char(12) MyVar;
      MyVar='Just Testing';
      put MyVar=;
   end;
   method term();
      do i=1 to 3;
         MyVar=10**i;
         put MyVar=;
      end;
   end;
enddata;
run;
```

Basic DS2 Syntax

Variable Scope

- Locally declared variables have a local scope and are private to the method in which they were declared.

```
data _null_;
   dcl double MyVar;
   method init();
      dcl char(12) MyVar;
      MyVar='Just Testing';
      put MyVar=;
   end;
   method term();
      do i=1 to 3;
         MyVar=10**i;
         put MyVar=;
      end;
   end;
enddata;
run;
```

Local: Available only to the INIT method

Basic DS2 Syntax

By default:

- Global variables are in the PDV.
- Local variables are not in the PDV.

PDV

_n_	MyVar	i
1	30	4

```
proc ds2;
data _null_;
   dcl double MyVar;
   method init();
      dcl char(12) MyVar;
      MyVar='Just Testing';
      put MyVar=;
   end;
   method term();
      do i=1 to 3;
         MyVar=10**i;
         put MyVar=;
      end;
   end;
enddata;
run;
quit;
```

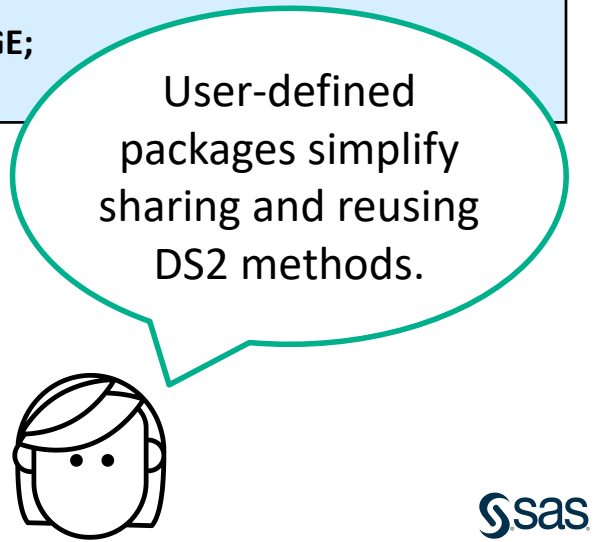
DS2 packages

DS2 User-Defined Packages

DS2 user-defined packages

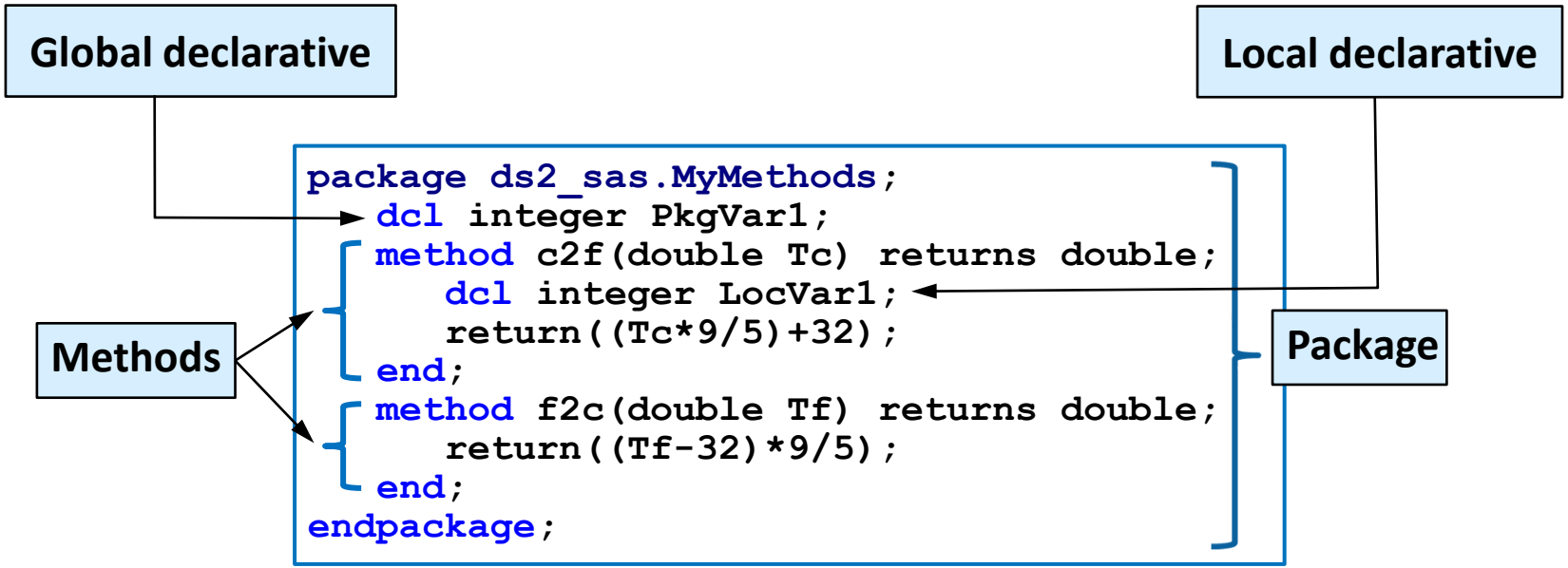
- enable users to extend DS2
- are collections of user-defined methods
- are stored in SAS libraries
  - Work library for temporary (session) packages
  - permanent library for reusable packages.

```
PACKAGE libref.package_name [/table-options];
    METHOD A();
        …method code…
    END;
    METHOD B();
        …method code…
    END;
ENDPACKAGE;
RUN;
```



DS2 PACKAGE Program Structure

PACKAGE program block structure:



Global package variables are private to the package instance. They can be referenced by any package method, but never appear in the PDV.

User-Defined Packages

To use a package in a DS2 data program, declare an instance of the package and call the methods using dot notation.

DECLARE|DCL PACKAGE *instance_variable (<constructor arguments>)*;

```
data _null_;

   dcl package ds2_sas.my_methods m();

   method init();
      dcl integer i;
      set ds2_sas.banks;
      do i=1 to 3;
         MyVar=10**i;
      end;
   end;
enddata;
run;
```

Predefined packages

- **FCMP**
  Supports calls to FCMP functions and subroutines from within the DS2 language. Not supported in CAS.
- **Hash and hash iterator**
  Enables you to quickly and efficiently store, search, and retrieve data based on unique lookup keys.
- **HTTP**
  Constructs an HTTP client to access HTTP web services.
- **JSON**
  Enables you to create and parse JSON text.
- **Logger**
  Provides a basic interface (open, write, and level query) to the SAS logging facility.

Predefined packages

- **Matrix**
  Provides a powerful and flexible matrix programming capability.
  Not supported in CAS.
- **PCRXFIND and PCRXREPLACE**
  Provides a way to find a substring within a given string and replace a substring.
- **SQLSTMT**
  Provides a way to pass FedSQL statements to a DBMS for execution and to access the result set returned by the DBMS.
  Not supported in CAS.
- **TZ**
  Provides a way to process local and international time and date values.

HTTP package

```
method run();
  /* instantiate the package */
  declare package http h();
  declare varchar(1024) character set utf8 body;
  declare int rc;

  /* create a GET */
  h.createGetMethod('http://api.worldbank.org/countries/fr/');
  /* execute the GET */
  h.executeMethod();
  /* retrieve the response body as a string */
  h.getResponseBodyAsString(body, rc);
  put body;
end;
```

Demo

More info

Doc and other resources

High-Performance Data Manipulation with SAS DS2
by Mark Jordan from SGF 2020

https://youtu.be/HP7tzXHxkt0

DS2 reference
https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/ds2ref/titlepage.htm

DS2 Programmer's guide
https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/ds2pg/titlepage.htm

Want to learn more?

Check out the web for the latest training and book resources.

sas.com/training | sas.com/books

SAS Training

Public training courses

- Programming for SAS® Viya®
- High-Performance Data Manipulation with SAS® DS2
- SAS Academy for Data Science

SAS books

- Mastering the SAS® DS2 Procedure: Advanced Data-Wrangling Techniques (Second Edition)
- SAS® Viya®: The Python Perspective

Thank you

sas.com