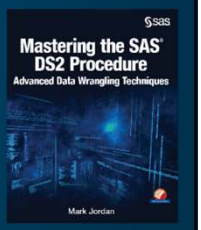
A Brief Introduction to DS2

By Mark Jordan

Email: Mark.Jordan@sas.com
Twitter: @SASJedi
Blog: http://sasjedi.tips
Author Info: http://support.sas.com/jordan



1

Objectives

- Describe the DS2 programming language.
- Identify when it is most appropriate to use the DS2 language.
- Compare the DS2 DATA program to Base SAS DATA step execution.
- Understand basic DS2 Syntax
  - Explain DS2 variable declaration and its effect on variable scope.
  - Name three DS2 system methods and the conditions under which they execute.
- Parallel Processing in DS2
  - Describe the similarities and differences between DS2 DATA and THREAD programs
  - Convert a DS2 DATA program to a DS2 thread.
  - Execute DS2 threads from a DS2 DATA program

2

Introduction to DS2

- Advanced data manipulation language
- DATA step-like syntax
- Closely integrated with SQL
- Convenient reusability

3

Base SAS DATA Step

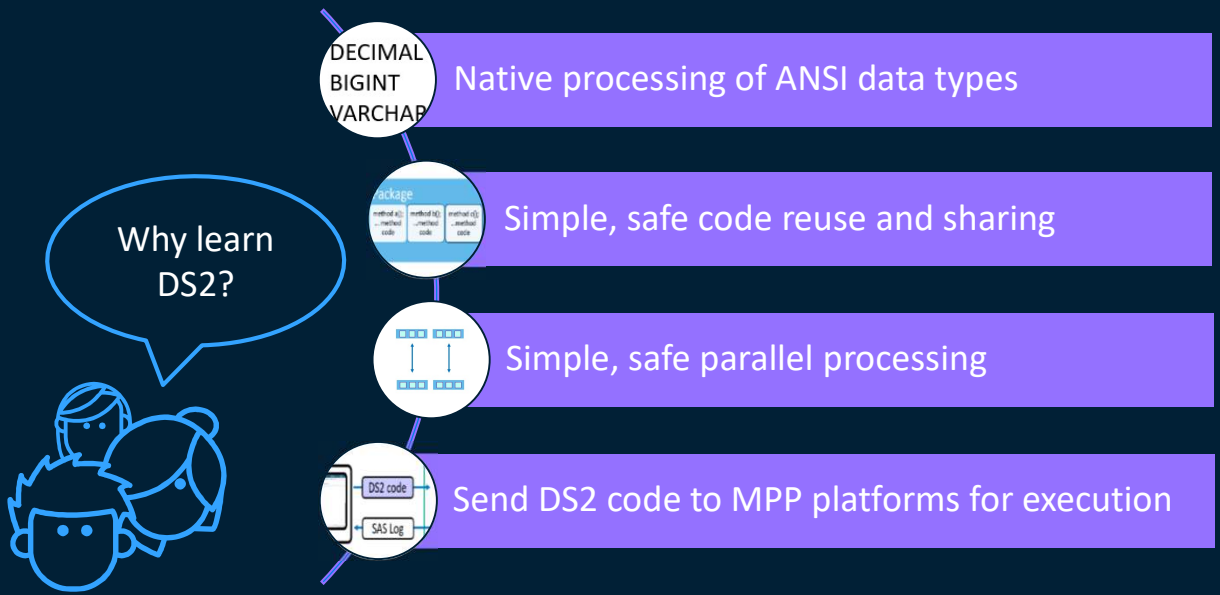
```
data _null_;
    Text='Hello, World!';
    put Text=;
run;
```

DS2 Data Program

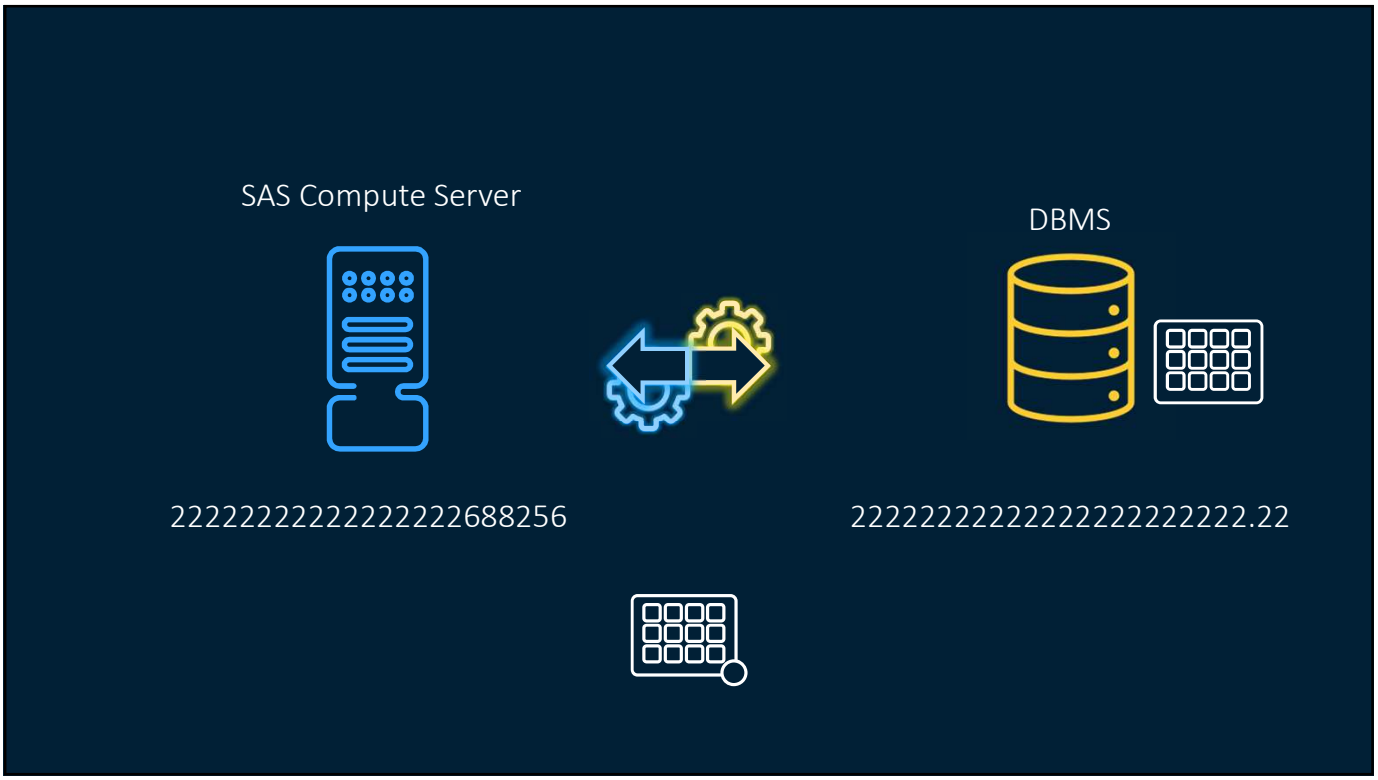
```
data _null_;
    method init();
        Text='Hello, World!';
        put Text=;
    end;
enddata;
run;
```

4

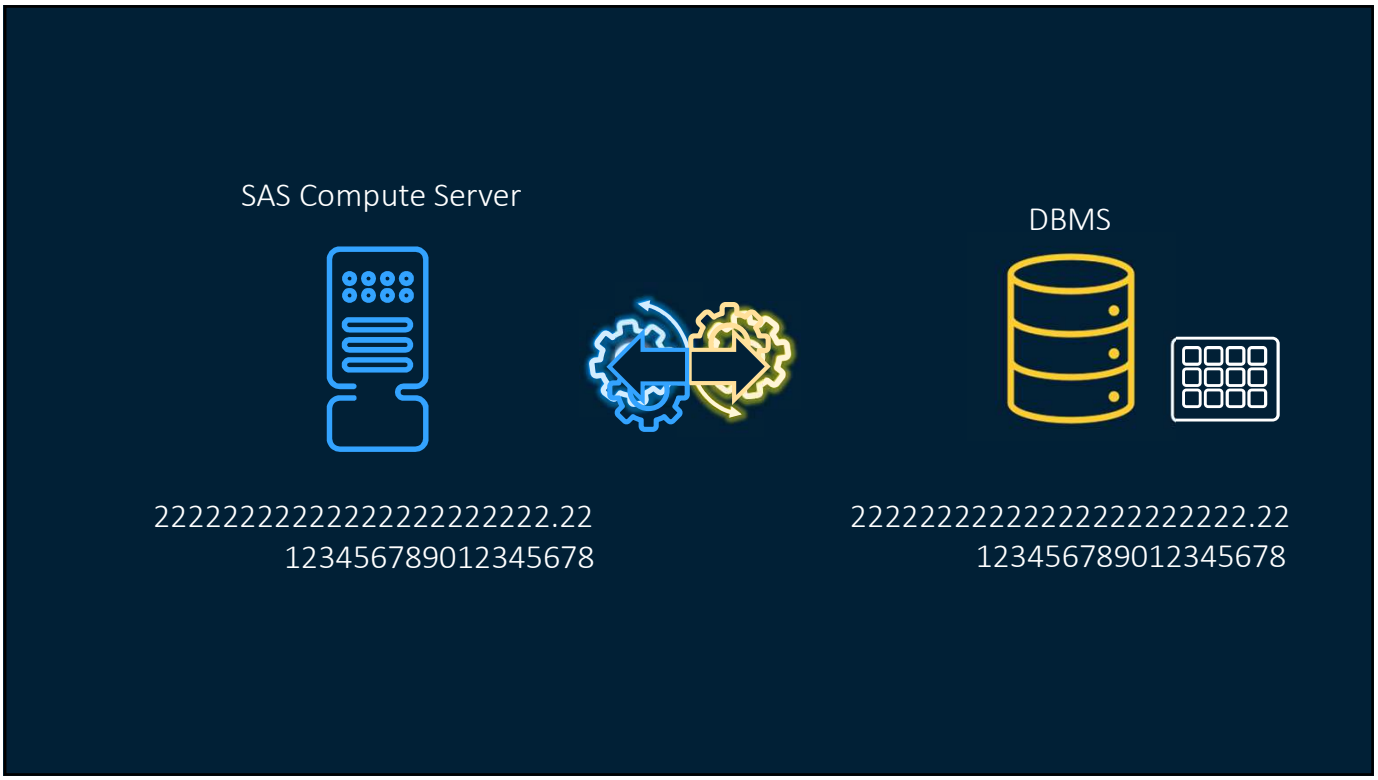
DS2 Superpowers



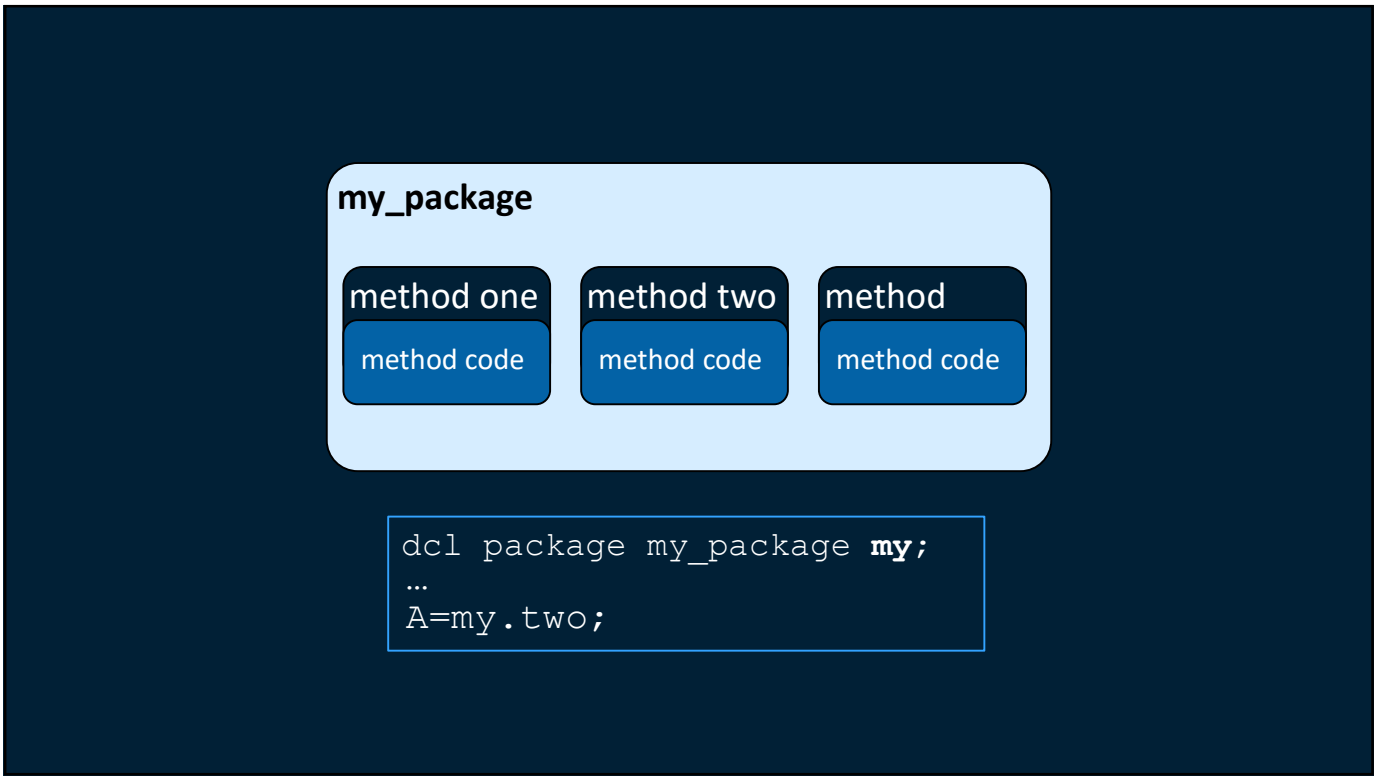
5



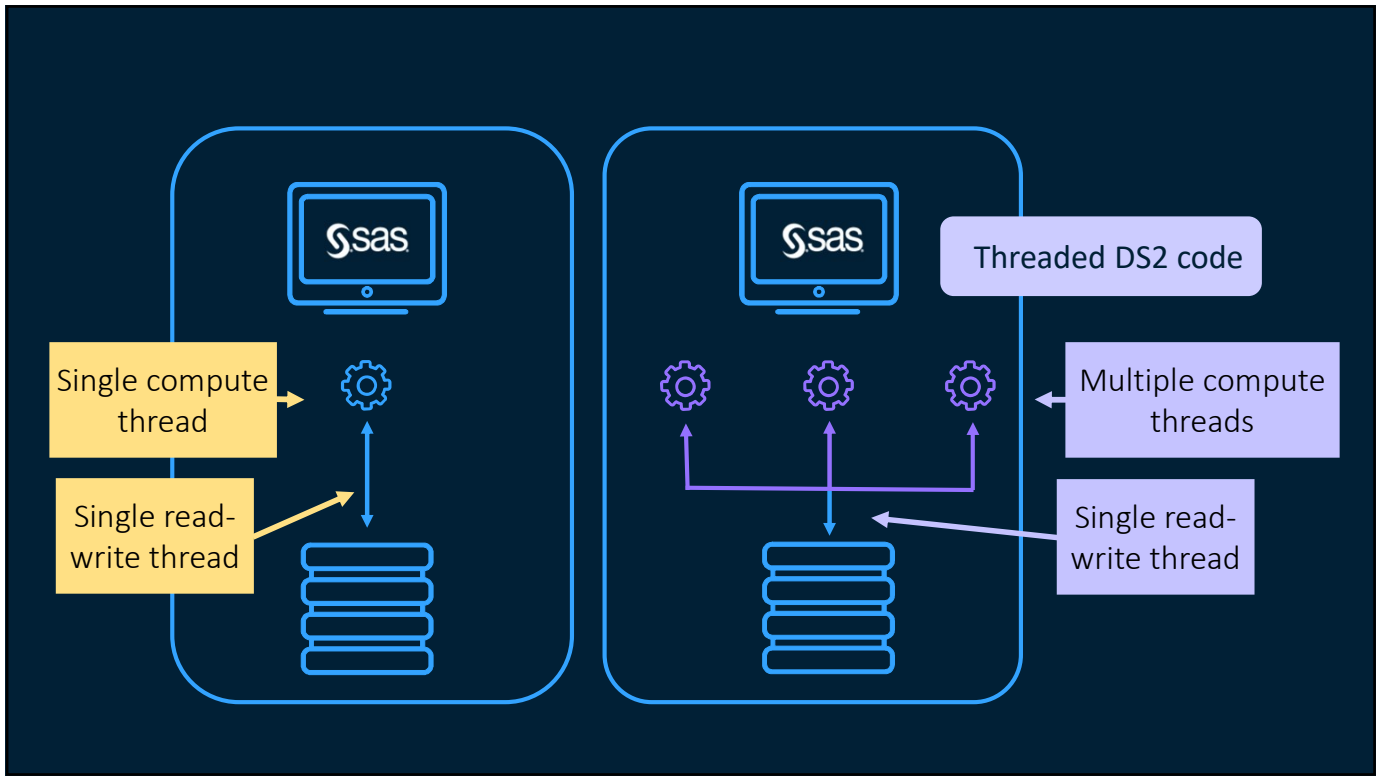
6


SAS Compute Server
DBMS
22222222222222222222.22
123456789012345678
22222222222222222222.22
123456789012345678

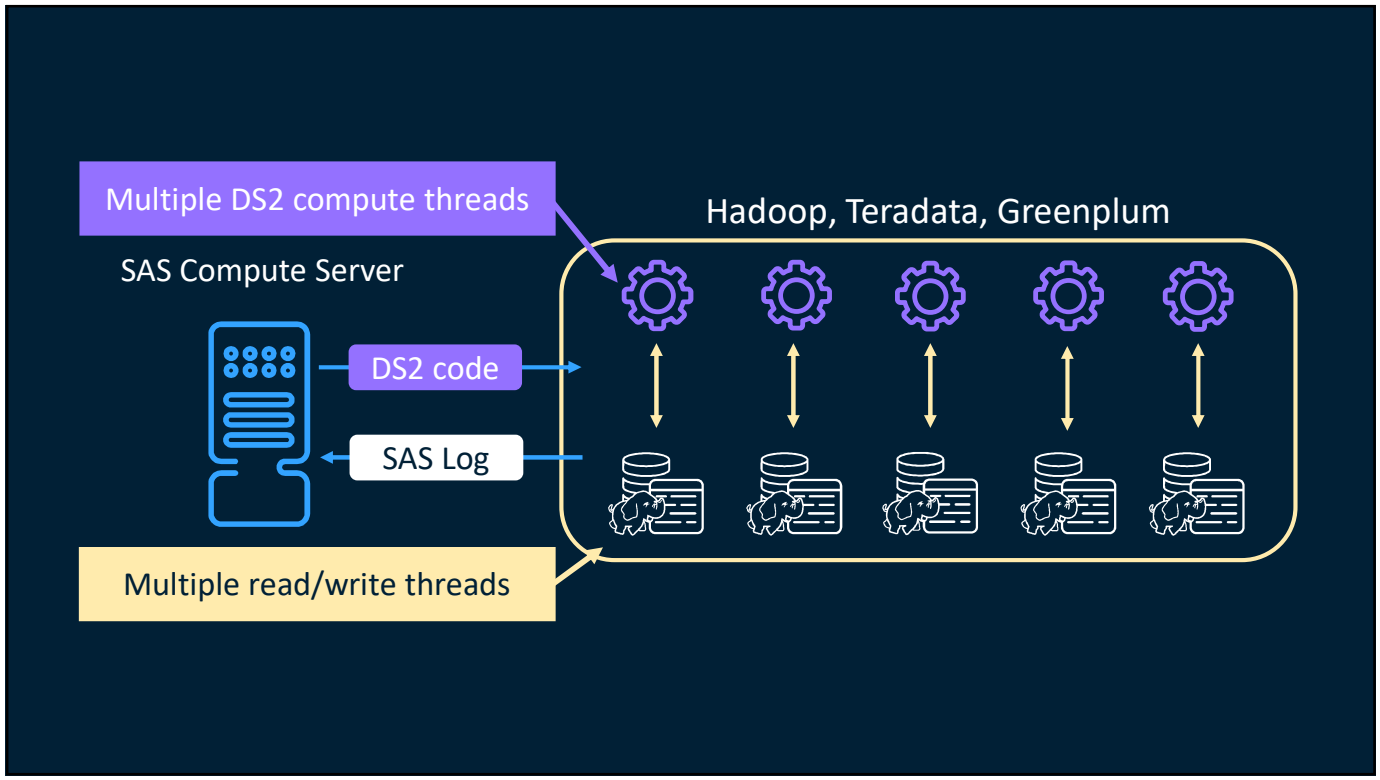
7


my_package
method one
method code
method two
method code
method
method code
dcl package my_package my;
…
A=my.two;

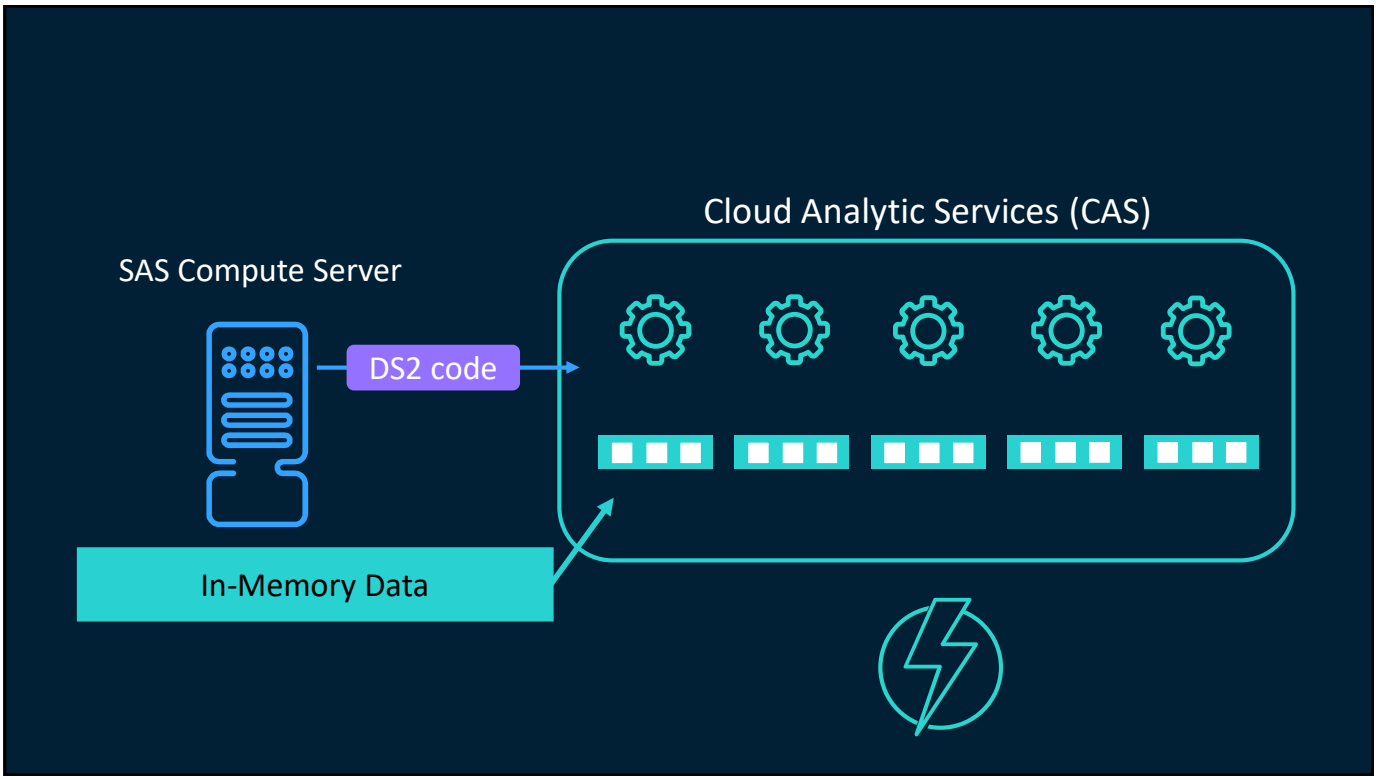
8


Single compute thread
Single read-write thread
Threaded DS2 code
Multiple compute threads
Single read-write thread

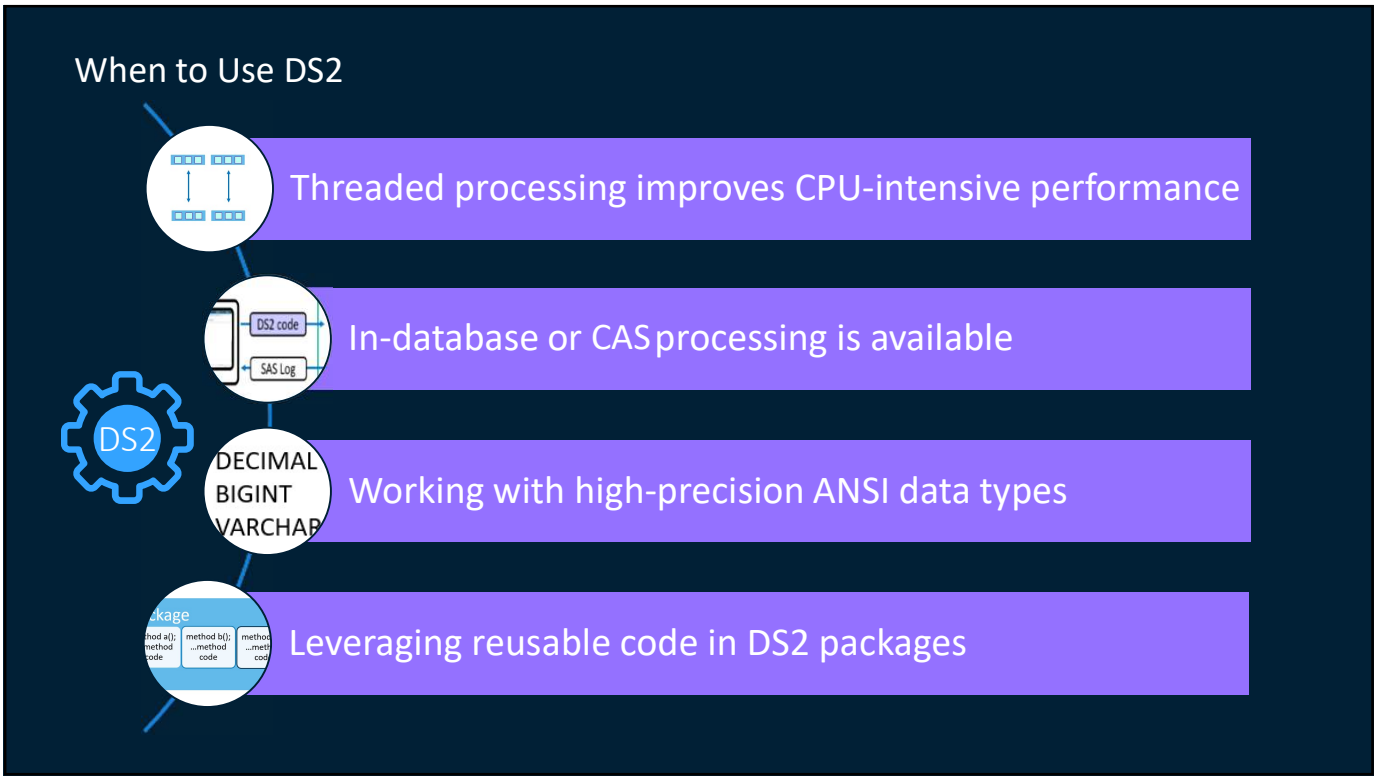
9


Multiple DS2 compute threads
SAS Compute Server
DS2 code
SAS Log
Multiple read/write threads
Hadoop, Teradata, Greenplum

10


Cloud Analytic Services (CAS)
SAS Compute Server
DS2 code
In-Memory Data

11


When to Use DS2
Threaded processing improves CPU-intensive performance
DS2 code
SAS Log
In-database or CAS processing is available
DS2
DECIMAL
BIGINT
VARCHAR
Working with high-precision ANSI data types
Leveraging reusable code in DS2 packages

12

DS2 Program Types

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

In the SAS windowing environment, program blocks without a RUN before the QUIT statement won't execute.

13

Data programs

- begin with a DATA statement
- end with an ENDDATA statement
- require a RUN statement to execute.

```
data my.table;
```

```
enddata;
run;
```

14

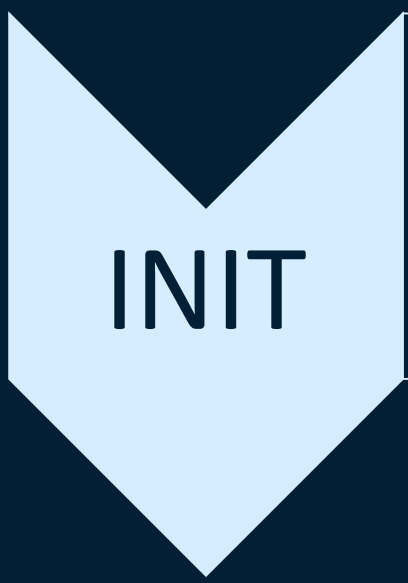
Methods

- Begin with a METHOD statement
- End with an END statement
- Contain all executable statements in the program
- System or user-defined

```
data my.table;
   method myMethod(double x);
      put 'myMethod';
   end;
   method init();
      put 'INIT';
   end;
enddata;
run;
```

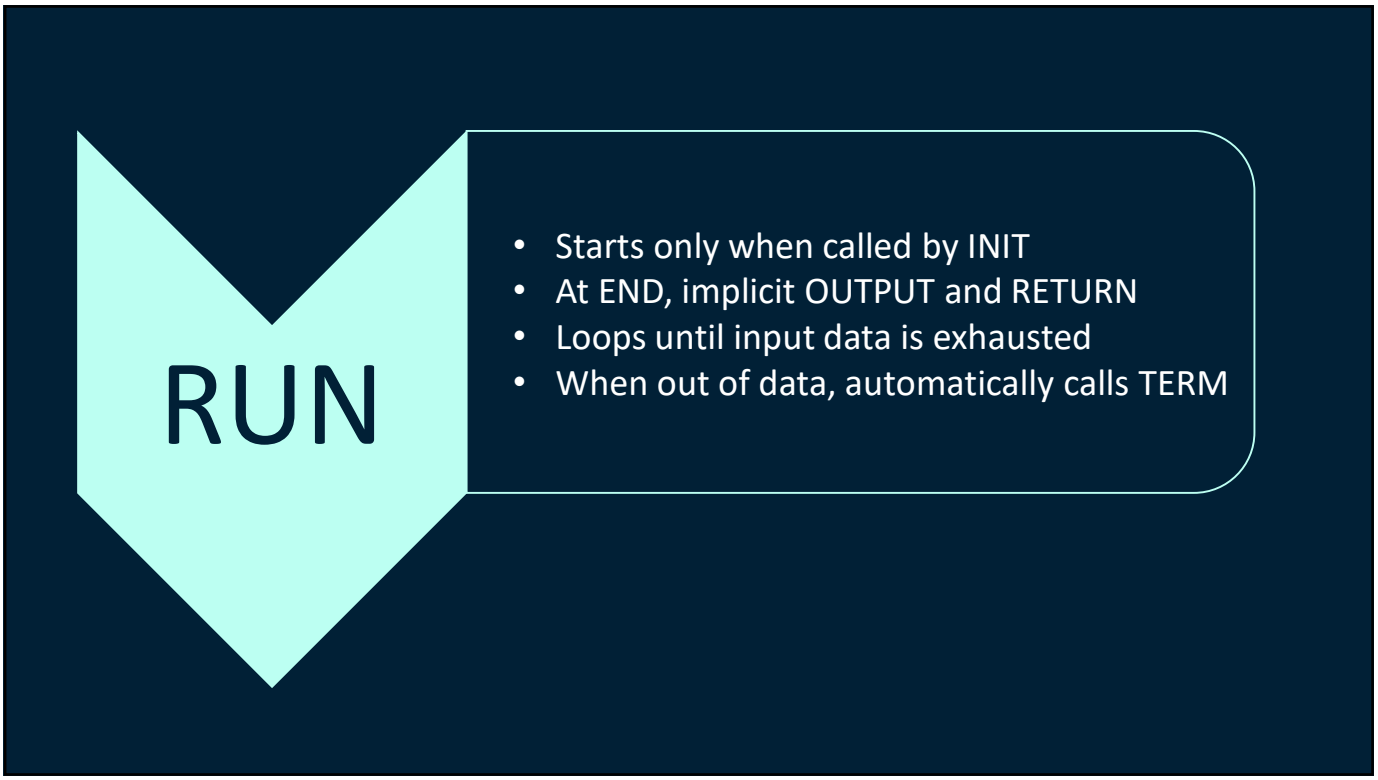
15

System Methods Control Data Program Flow



- Called when data program begins execution
- Each statement executed only once
- At END, automatically calls the RUN method

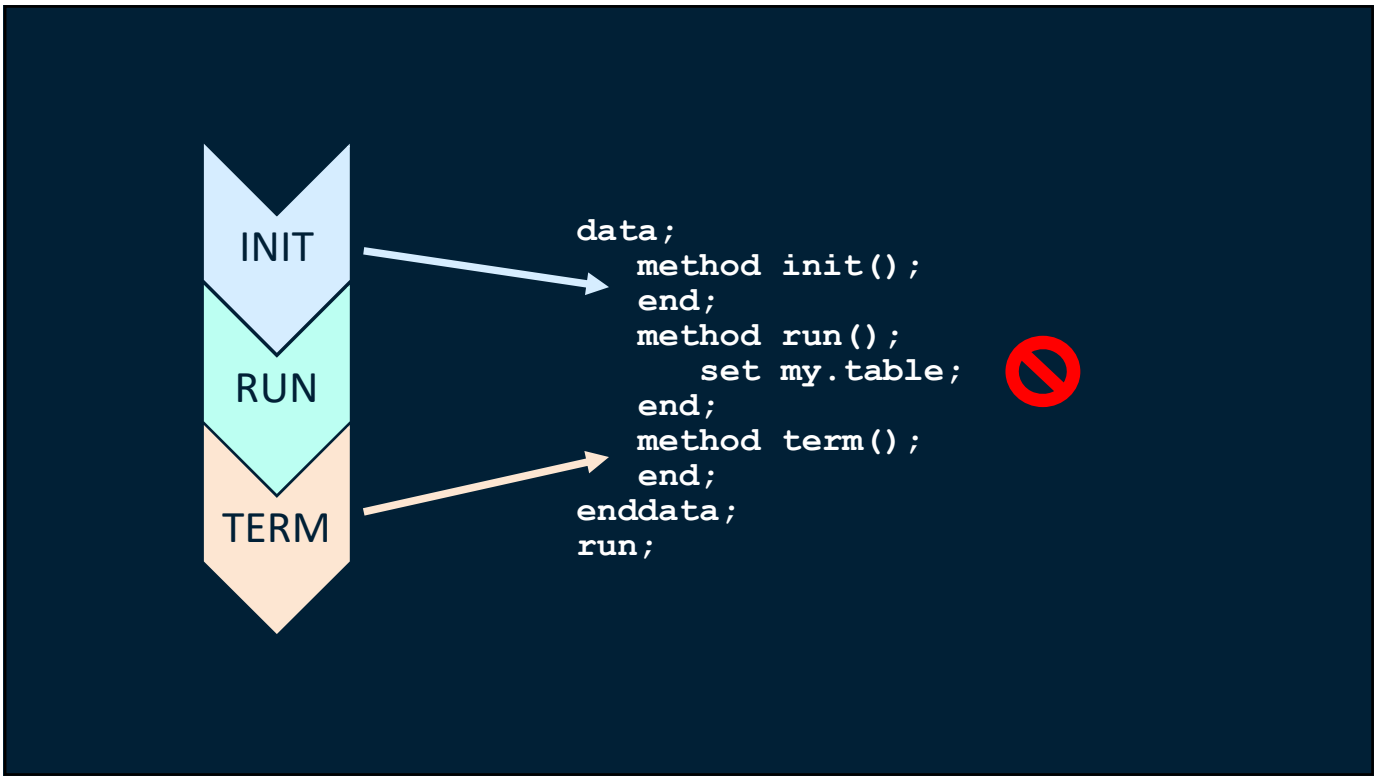
16


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

17


Term
Starts only when called by RUN
Each statement executed only once
At END, DS2 program execution terminates

18



19

Declaration Statements

```
data;
   dcl double Amount;
   retain Amount 0;
   method run();
      dcl int year;
      set orion.banks;
      amount=1000;
      do year=1 to 5;
         amount=sum(amount,amount*rate);
      end;
   end;
enddata;
```

20

Variable declaration

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

```
/* Declare three DOUBLE variables formatted dollar12.2*/
dcl double Var1 Var2 Var3 having format dollar12.2;

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

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

21

```
data;
   dcl double Value;
   method run();
      set orion.banks;
      do Month='JAN','FEB','MAR';
         do i=1 to 3;
            Value=10**i;
            if i > x then output;
         end;
      end;
   end;
enddata;
run;
```

```
WARNING: No DECLARE for referenced variable month;
         creating it as a global variable of type char(3).
WARNING: No DECLARE for referenced variable i;
         creating it as a global variable of type double.
WARNING: No DECLARE for referenced variable x;
         creating it as a global variable of type double.
```

22

DATA Step

- KEEP
- DROP
- RETAIN

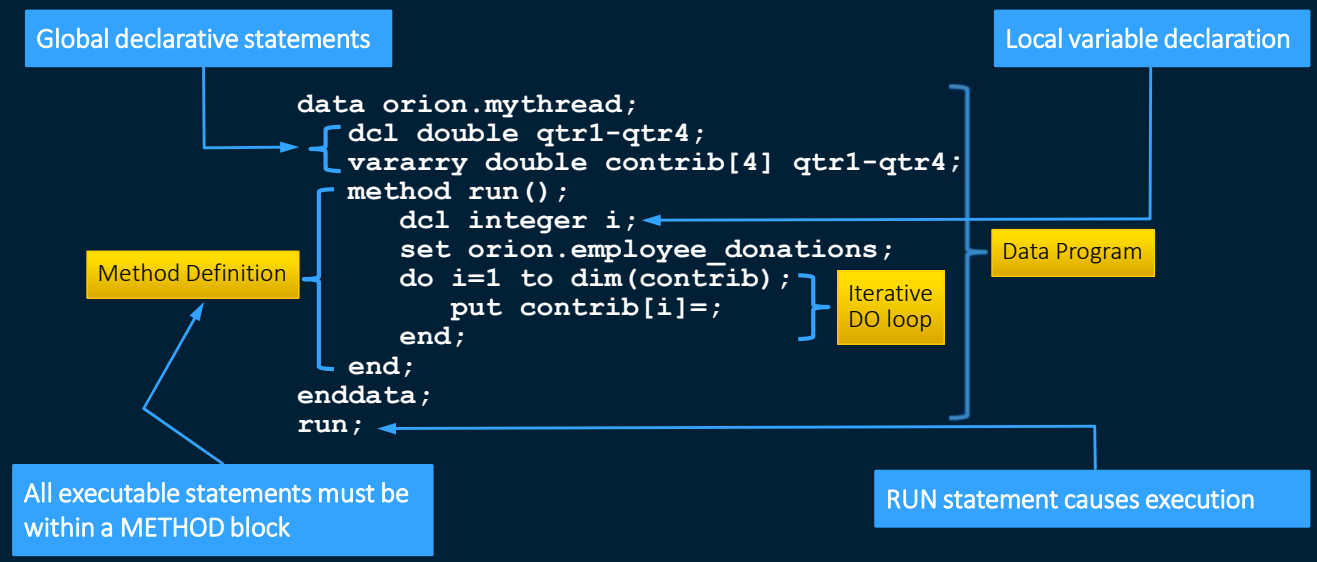
DS2

- KEEP
- DROP
- RETAIN

```
ERROR: Compilation error.
```

23

DS2 data program block structure



24

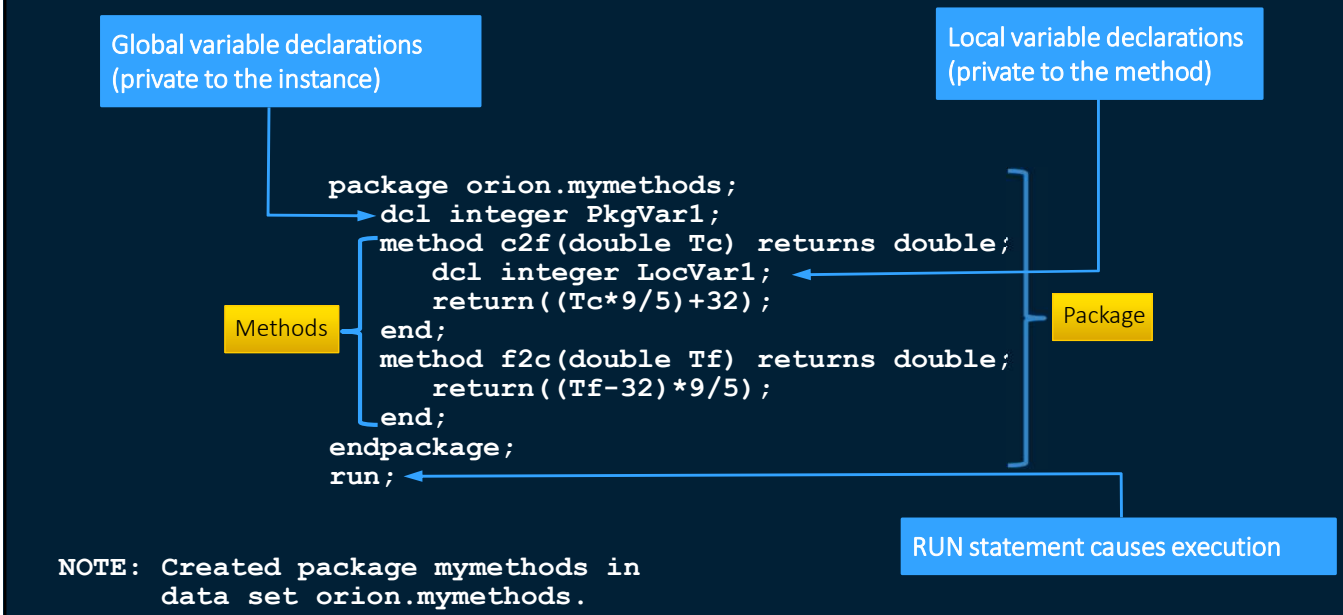
Package programs

DS2 packages are collections of methods and variables stored in SAS libraries.

- Simplify re-using code in DS2 DATA and Thread programs.
- Predefined packages ship with SAS to extend DS2 capabilities.
- User-defined packages enable easy, secure sharing of proprietary algorithms.
- Packages enable object-oriented programming for the DS2 language
  - Can accept parameters when instantiated
  - Can include user-defined constructor and destructor methods
  - Global variables in a package are private to the package instance and do not affect the PDV of the DATA program in which they are used

25

Package programs



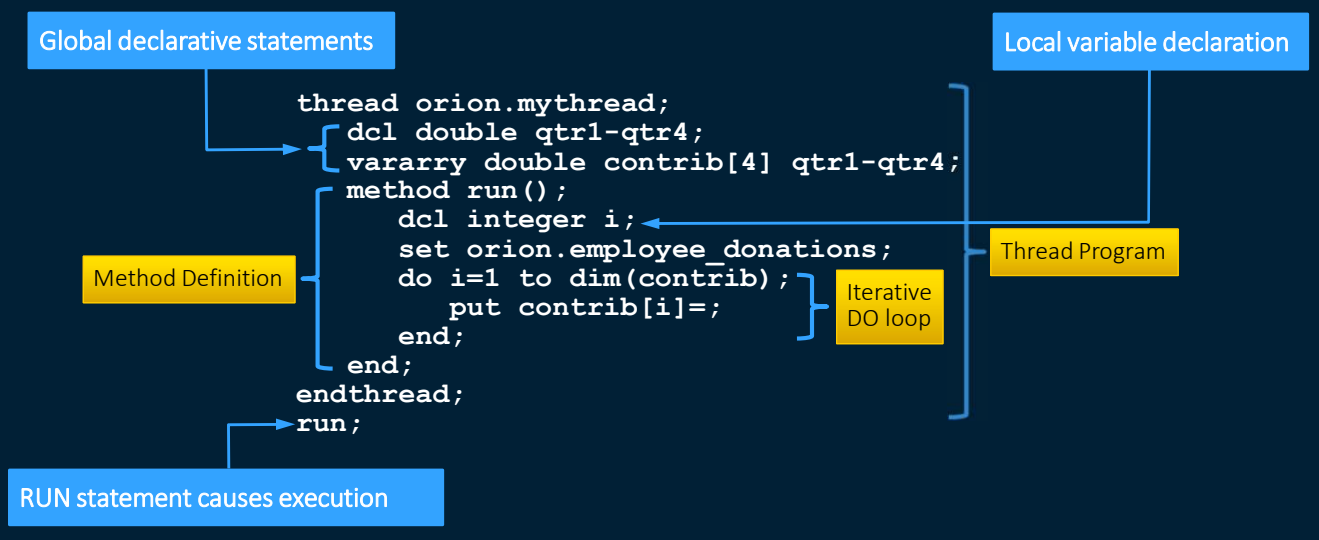
26

Thread programs

- Threads are programs stored in SAS libraries and used for parallel processing in DS2
- Similar to DATA programs in that they
  - Enforce a more structured programming syntax
  - Require all executable code to be included in a method definition
  - Require at least explicitly written system method
  - Global variables appear in the PDV of the DATA program in which they are used
- Similar to DS2 packages in that they
  - can be re-used when stored in permanent SAS libraries.
  - can contain user-defined methods.
  - can accept parameters

27

Thread programs




```
NOTE: Created thread mythread in data set orion.mythread.
```

28

User-Defined Methods

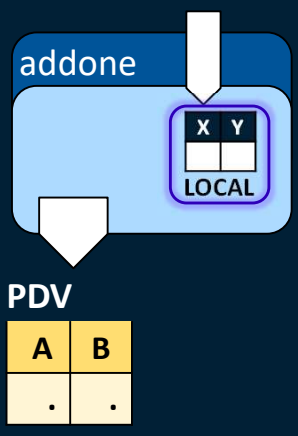
METHOD *Name* (<IN_OUT>*data-type Parameter1* …)
,<IN_OUT>*data-type ParameterN*
<RETURNS *data-type*>;
… more DS2 code …
<RETURN *value*>**;**
END;

- can accept parameters
- parameters can be either standard or IN_OUT
- IN_OUT parameter values can be modified
- can return a value
- execute only when called
- can be called multiple times

29

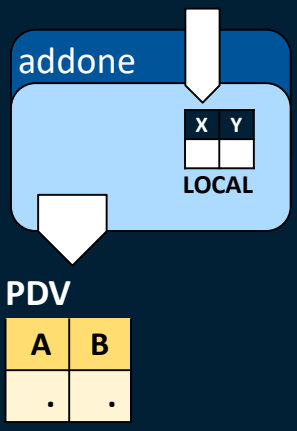
Returning a value

```
dcl int A B;
method addone(int x) returns int;
   dcl int y;
   y=x+1;
   return y;
end;
method init();
   B=3;
   A=addone(B-2);
end;
```



30

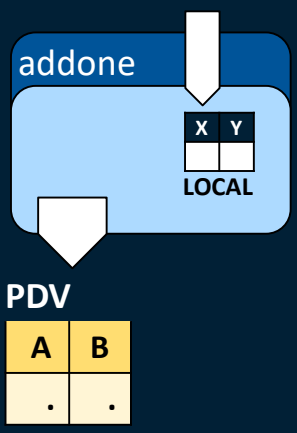
```
dcl int A B;
method addone(int x) returns int;
   dcl int y;
   y=x+1;
   return y;
end;
method init();
   B=3;
   A=addone(B-2);
end;
```



31

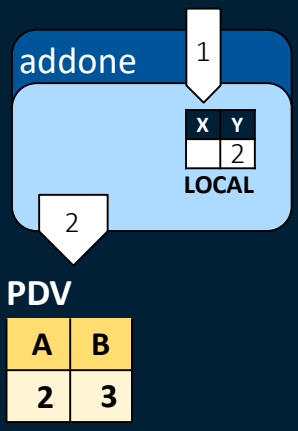
Standard Parameters

```
dcl int A B;
method addone(int x) returns int;
   dcl int y;
   y=x+1;
   return y;
end;
method init();
   B=3;
   A=addone(B-2);
end;
```



32

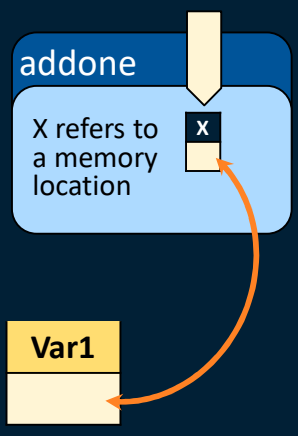
```
dcl int A B;
method addone(int x) returns int;
   dcl int y;
   y=x+1;
   return y;
end;
method init();
   B=3;
   A=addone(B-2);
end;
```



33

IN_OUT Parameters

```
method addone(in_out int X);
   x=x+1;
end;
```

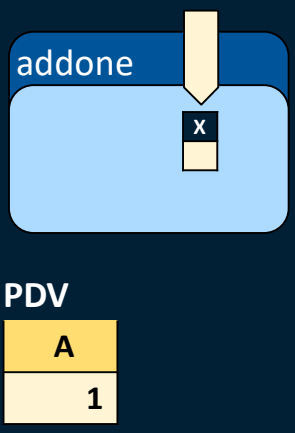



```
ERROR: Parse failed:  method addone
(in_out  >>> vararray <<<  int x[4]);

ERROR: In call of addone: argument 1 is 'in_out'; therefore, the type
       of the argument must be identical to the type of the method
       parameter (requires int, found double)
```

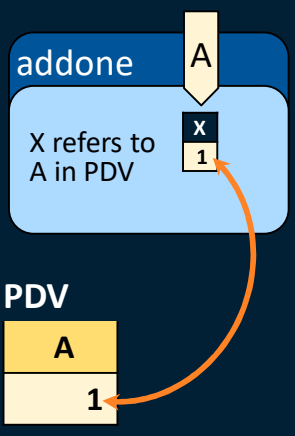
34

```
dcl int A;
method addone(in_out int X);
   x=x+1;
end;
method init();
   A=3-2;
   addone(A);
end;
```



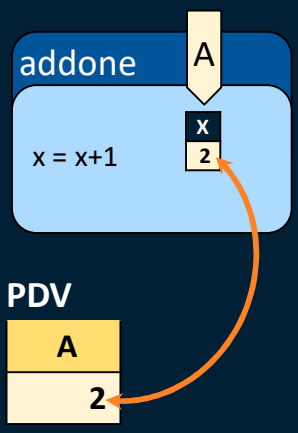
35

```
dcl int A;
method addone(in_out int X);
   x=x+1;
end;
method init();
   A=3-2;
   addone(A);
end;
```



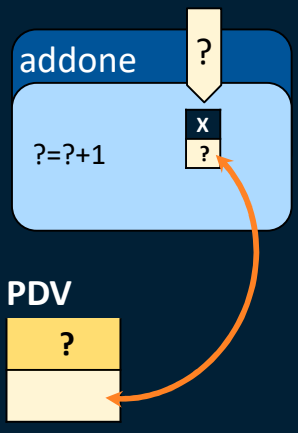
36

```
dcl int A;
method addone(in_out int X);
   x=x+1;
end;
method init();
   A=3-2;
   addone(A);
end;
```



37

```
method addone(in_out int X);
   x=x+1;
end;
method init();
   addone(3-2);
end;
```




```
ERROR: Compilation error.
ERROR: In call of addone: argument 1 is 'in_out'; therefore, the
       argument must be a modifiable value.
ERROR: Line 22: Attempt to obtain a value from a void expression.
```

38

Variable Scope

```
data;
   dcl double Value;
   method init();
      Value=5000;
   end;
   method run();
      set orion.banks;
      Total=sum(value,value*rate);
   end;
   method term();
      dcl char(12) Value;
      Value='Just Testing';
      put Value=;
   end;
enddata;
run;
```

PDV

Value N 8	Name $ 29	Rate N 8	Total N 8

39

DATA step vs. DS2 data program

```
data _null_;
   /* Section 1 */
   if _n_ =1 then do;
      Text='**> Starting';
      put Text;
   end;
   /* Section 2 */
      set orion.banks end=last;
      put _all_;
   /* Section 3 */
   if last then do;
      Text='**> All done!';
      put Text;
   end;

run;
```

```
data _null_;
 /* Section 1 */
 method init();
    Text='**> Starting';
    put Text;
 end;
/* Section 2 */
method run();
   set orion.banks;
   put _all_;
end;
/* Section 3 */
method term();
   Text='**> All done!';
   put Text;
end;
enddata;
run;
```

40

Converting to DATA Step DS2

PROC DSTODS2 IN=*fully-qualified-DATA-step-file-name*
OUT=*ds2-program-file-name*
OUTDIR=*path-for-output-file*;

```
proc dstods2 in ="&path/demos/ds203d02a.sas"
             out="ds203d02a_ds2.sas"
             outdir="&path/demos/";
run;


   set orion.banks /* END = LAST */;
```

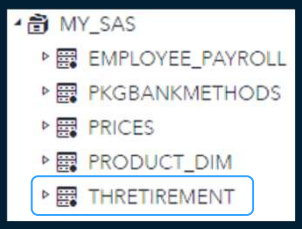
41

Converting a DS2 Data Program to a Thread

```
thread my_sas.thRetirement;
   dcl package my_sas.pkgBankMethods bank();
   dcl double valueInitial valueFinal having format dollar12.2;
   keep Employee_ID, Employee_Name value:;
   dcl int _count;
   keep Employee_ID, Employee_Name value:;
   method run();
      dcl int duration;
      set orion.employees;
      valueInitial=Salary*.1;
      Duration=year(today())-year(employee_hire_date);
      valueFinal=bank.interest(ValueInitial,.04, Duration);
   end;
   method term();
      put '*** Thread' _threadid_ 'of' _nthreads_ 'processed'
          _count 'rows on' _hostname_;
   end;
endthread;
```

42

```
NOTE: Created thread thretirement in data set my_sas.thretirement.
```



43

```
data;
   dcl thread my_sas.thRetirement ret;
   method run();
      set from ret threads=3;
   end;
enddata;
run;
```

threadid=1

Total	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4
.	.	.	.	.	.

threadid=2

Total	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4
.	.	.	.	.	.

threadid=3

Total	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4
.	.	.	.	.	.

44

The THREADS= option specifies the number of threads to execute in parallel.

- Over threading can negatively affect performance.
- &SYSNCPU is a good starting point

```
set from ret threads=&SYSNCPU;
```

45

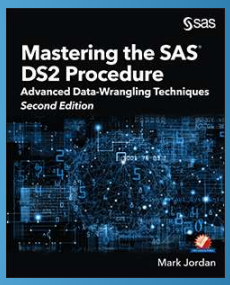
Executing DS2 Threads in Base SAS and in CAS

This demonstration illustrates improving performance of a CPU-bound DATA step using DS2 threaded processing.

46

Questions?

Email: Mark.Jordan@sas.com
Twitter: @SASJedi
Blog: http://sasjedi.tips
Author Page: http://support.sas.com/jordan



47