Handle Big Data With SAS®DS2

SEMPRA, 21November 2024

Charu Shankar
SAS Education

Charu Shankar, SAS® Institute

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking since 2007.

Charu curates and delivers unique content on SAS, SQL, Viya, etc. to support users in the adoption of SAS software.

When not coding, Charu teaches yoga and loves to explore Canadian trails with her husky Miko.

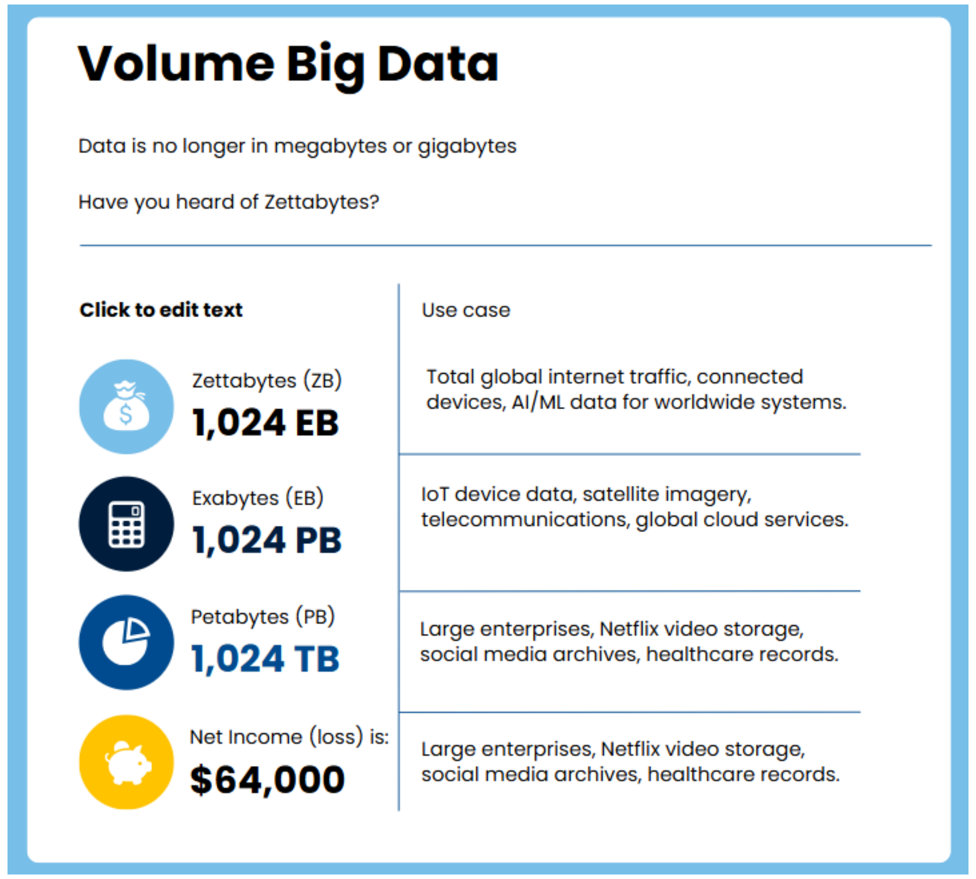
Agenda

- Describe the DS2 programming language
- Understand basic DS2 Syntax
- Create Methods to reuse
- Know When to use DS2
- Q &A

1Describe the DS2 Programming Language

What is Big Data

When volume, velocity and variety of data exceeds an organization's storage or compute capacity for accurate and timely decision-making

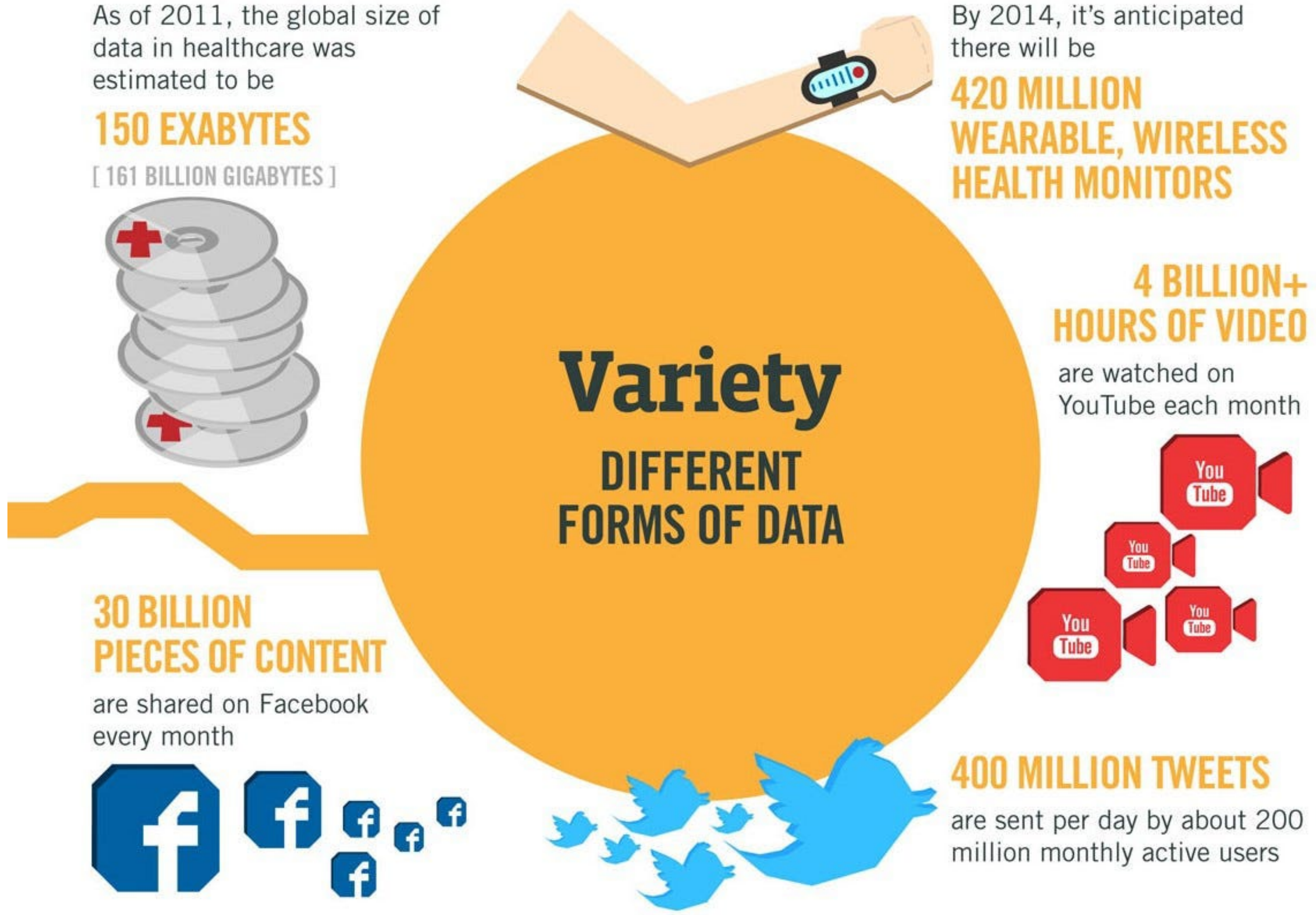


Volume Big Data

Data is no longer in megabytes or gigabytes

Have you heard of Zettabytes?

Click to edit text	Use case
Zettabytes (ZB) **1,024 EB**	Total global internet traffic, connected devices, AI/ML data for worldwide systems.
Exabytes (EB) **1,024 PB**	IoT device data, satellite imagery, telecommunications, global cloud services.
Petabytes (PB) **1,024 TB**	Large enterprises, Netflix video storage, social media archives, healthcare records.
Net Income (loss) is: **$64,000**	Large enterprises, Netflix video storage, social media archives, healthcare records.



Variety

DIFFERENT FORMS OF DATA

As of 2011, the global size of data in healthcare was estimated to be

150 EXABYTES

[161 BILLION GIGABYTES]

30 BILLION PIECES OF CONTENT are shared on Facebook every month

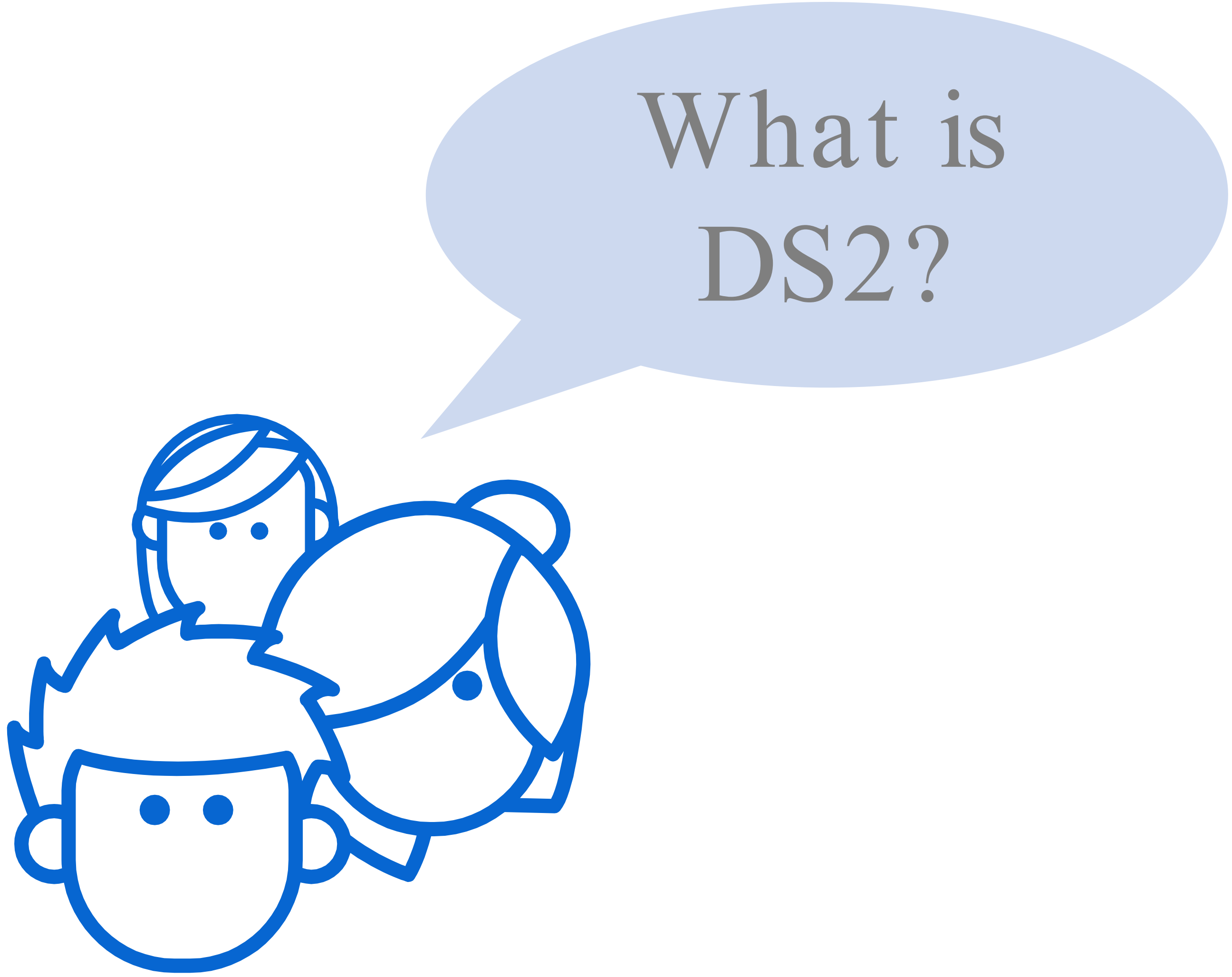
By 2014, it's anticipated there will be

420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO are watched on YouTube each month

400 MILLION TWEETS are sent per day by about 200 million monthly active users

What is DS2?



Advanced data manipulation language

DATA step-like syntax

Closely integrated with SQL

Convenient reusability

Structural Similarities and Differences – Data Step and DS2

Base SAS DATA Step

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

DS2 Data Block

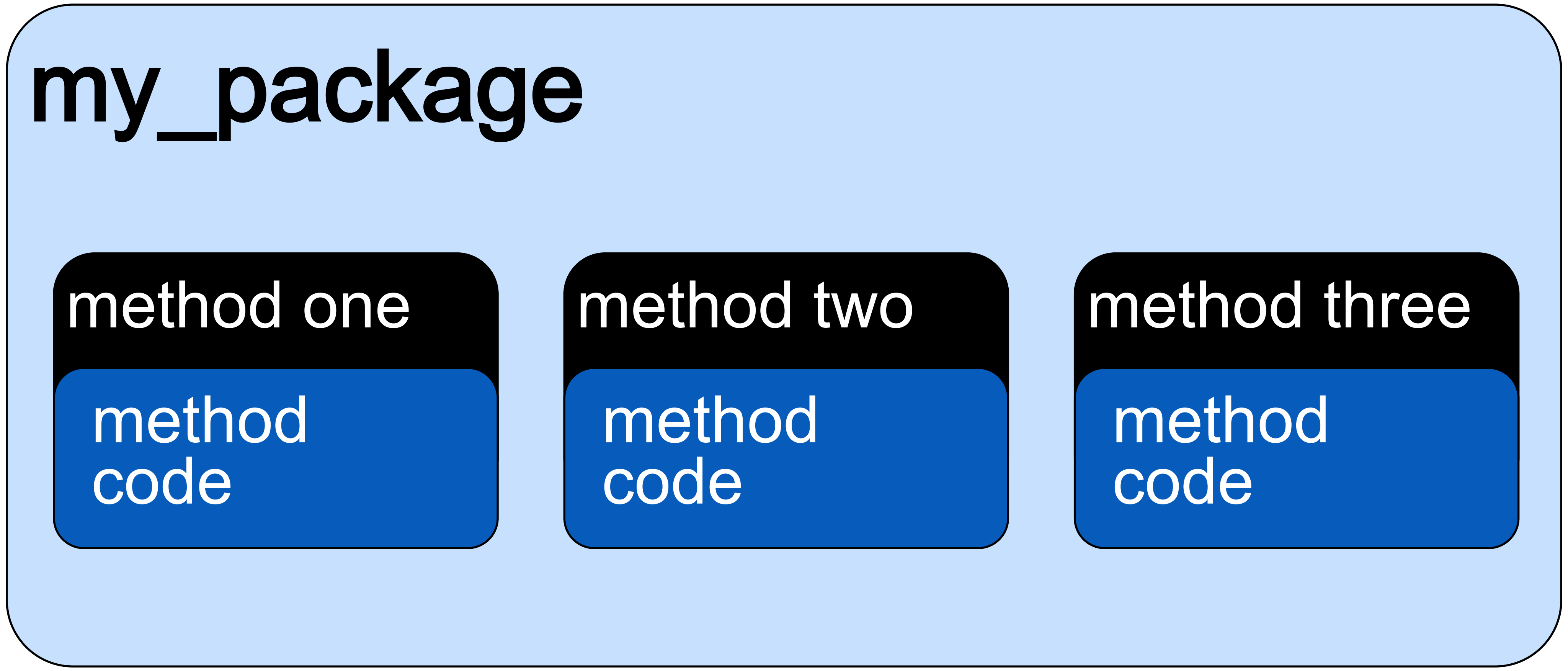
```
data _null_;
   method init();
      Text='Hello, World!';
      put Text=;
   end;
enddata;
run;
```

OOPs –Object Oriented Programming

SAS DS2 is considered object-oriented because it introduces concepts such as methods, packages, and inheritance, which are key features of object-oriented programming (OOP).

```
proc ds2;
   data bees / overwrite=yes;
      dcl package bumblebee_class bee; /* Declare an instance of the
bumblebee_class package */
      method init();
         bee = _new_ bumblebee_class("Bombus", "pensylvanicus");
      end;
      method run();
         bee.describe();
      end;
   enddata;
run;
quit;

/* Defining the bumblebee_class package */
proc ds2;
   package bumblebee_class;
      dcl char(20) species;
      dcl char(30) scientific_name;

      /* Constructor method to initialize values */
      method bumblebee_class(char(20) sp, char(30) sci_name);
         species = sp;
         scientific_name = sci_name;
      end;

      /* Method to display bee information */
      method describe();
         put "Species: " species "Scientific Name: " scientific_name;
      end;
   endpackage;
run;
quit;
```

DS2- Object Oriented Programming



my_package

method one

method code

method two

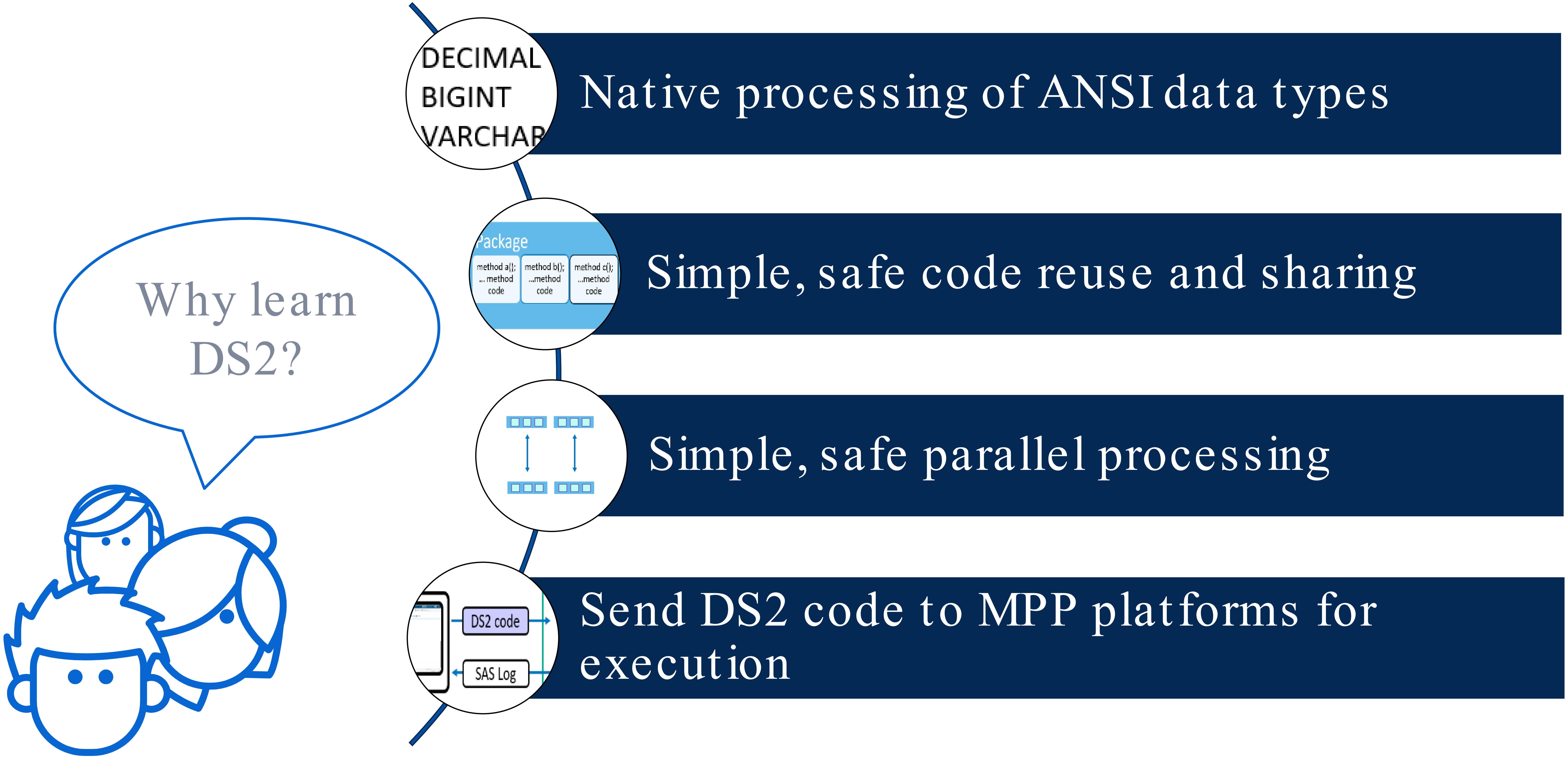
method code

method three

method code

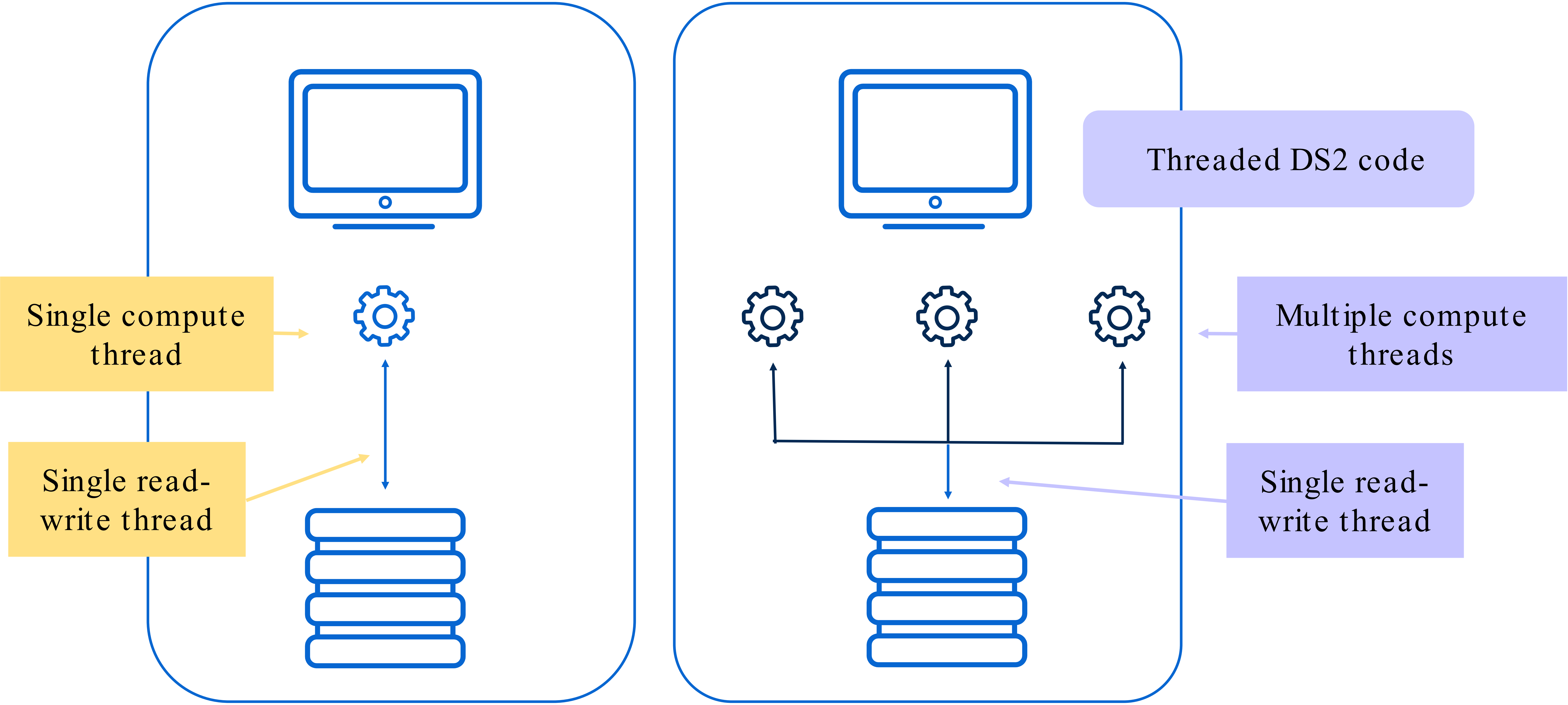
```
dcl package my_package my;
    …
A=my.two;
```

DS2 Superpowers


Why learn DS2?
DECIMAL
BIGINT
VARCHAR
Package
method a(); ... method code
method b(); ...method code
method c(); ...method code
DS2 code
SAS Log

- Native processing of ANSI data types
- Simple, safe code reuse and sharing
- Simple, safe parallel processing
- Send DS2 code to MPP platforms for execution

Threaded Processing – Data Step & DS2



Single compute thread

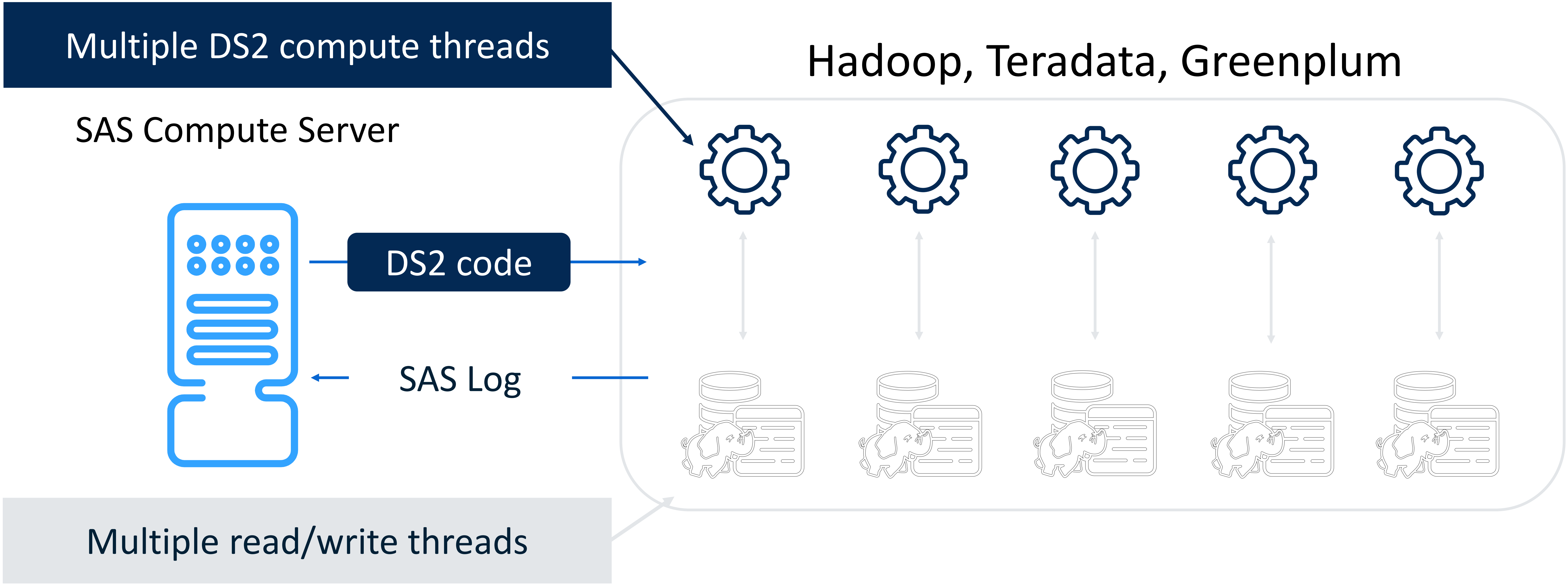
Single read-write thread

Threaded DS2 code

Multiple compute threads

Single read-write thread

DS2 – Massively Parallel Processing System



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

ANSI SQL Data Type Support - Extended Data Types

The syntax of the DS2 language intersects with the SAS DATA step but also includes additional data types, ANSI SQL types, programming structure elements, and user-defined methods and packages

BIGINT

BINARY(*n*)

CHAR(*n*)

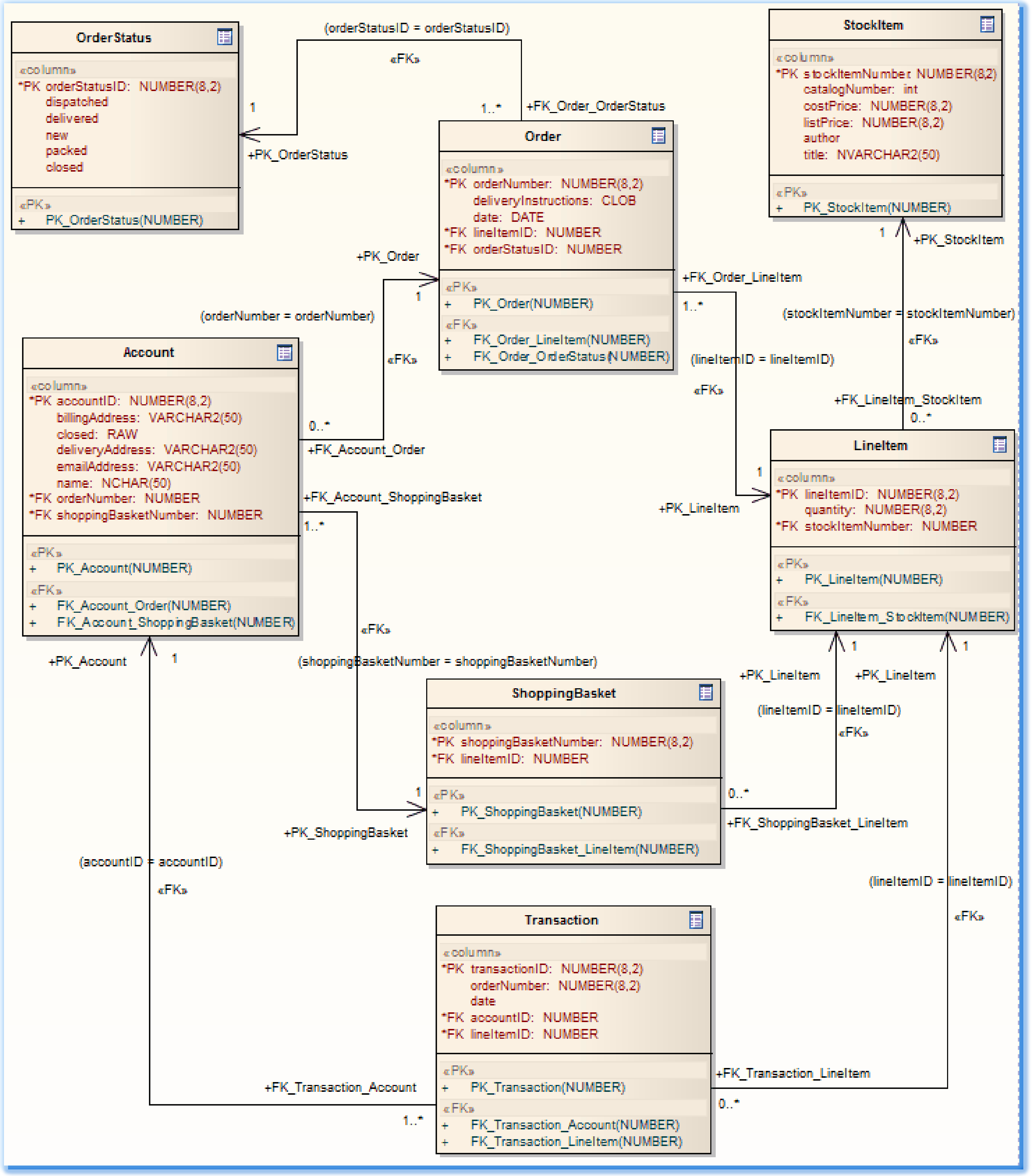
DATE

DOUBLE

FLOAT(*p*)

INTEGER

NCHAR(*n*)



```
OrderStatus
«column»
*PK orderStatusID: NUMBER(8,2)
    dispatched
    delivered
    new
    packed
    closed
«PK»
+   PK_OrderStatus(NUMBER)

Order
«column»
*PK orderNumber: NUMBER(8,2)
    deliveryInstructions: CLOB
    date: DATE
*FK lineItemID: NUMBER
*FK orderStatusID: NUMBER
«PK»
+   PK_Order(NUMBER)
«FK»
+   FK_Order_LineItem(NUMBER)
+   FK_Order_OrderStatus(NUMBER)

StockItem
«column»
*PK stockItemNumber: NUMBER(8,2)
    catalogNumber: int
    costPrice: NUMBER(8,2)
    listPrice: NUMBER(8,2)
    author
    title: NVARCHAR2(50)
«PK»
+   PK_StockItem(NUMBER)

Account
«column»
*PK accountID: NUMBER(8,2)
    billingAddress: VARCHAR2(50)
    closed: RAW
    deliveryAddress: VARCHAR2(50)
    emailAddress: VARCHAR2(50)
    name: NCHAR(50)
*FK orderNumber: NUMBER
*FK shoppingBasketNumber: NUMBER
«PK»
+   PK_Account(NUMBER)
«FK»
+   FK_Account_Order(NUMBER)
+   FK_Account_ShoppingBasket(NUMBER)

LineItem
«column»
*PK lineItemID: NUMBER(8,2)
    quantity: NUMBER(8,2)
*FK stockItemNumber: NUMBER
«PK»
+   PK_LineItem(NUMBER)
«FK»
+   FK_LineItem_StockItem(NUMBER)

ShoppingBasket
«column»
*PK shoppingBasketNumber: NUMBER(8,2)
*FK lineItemID: NUMBER
«PK»
+   PK_ShoppingBasket(NUMBER)
«FK»
+   FK_ShoppingBasket_LineItem(NUMBER)

Transaction
«column»
*PK transactionID: NUMBER(8,2)
    orderNumber: NUMBER(8,2)
    date
*FK accountID: NUMBER
*FK lineItemID: NUMBER
«PK»
+   PK_Transaction(NUMBER)
«FK»
+   FK_Transaction_Account(NUMBER)
+   FK_Transaction_LineItem(NUMBER)
```

NVARCHAR(*n*)

REAL

SMALLINT

TIME(*p*)

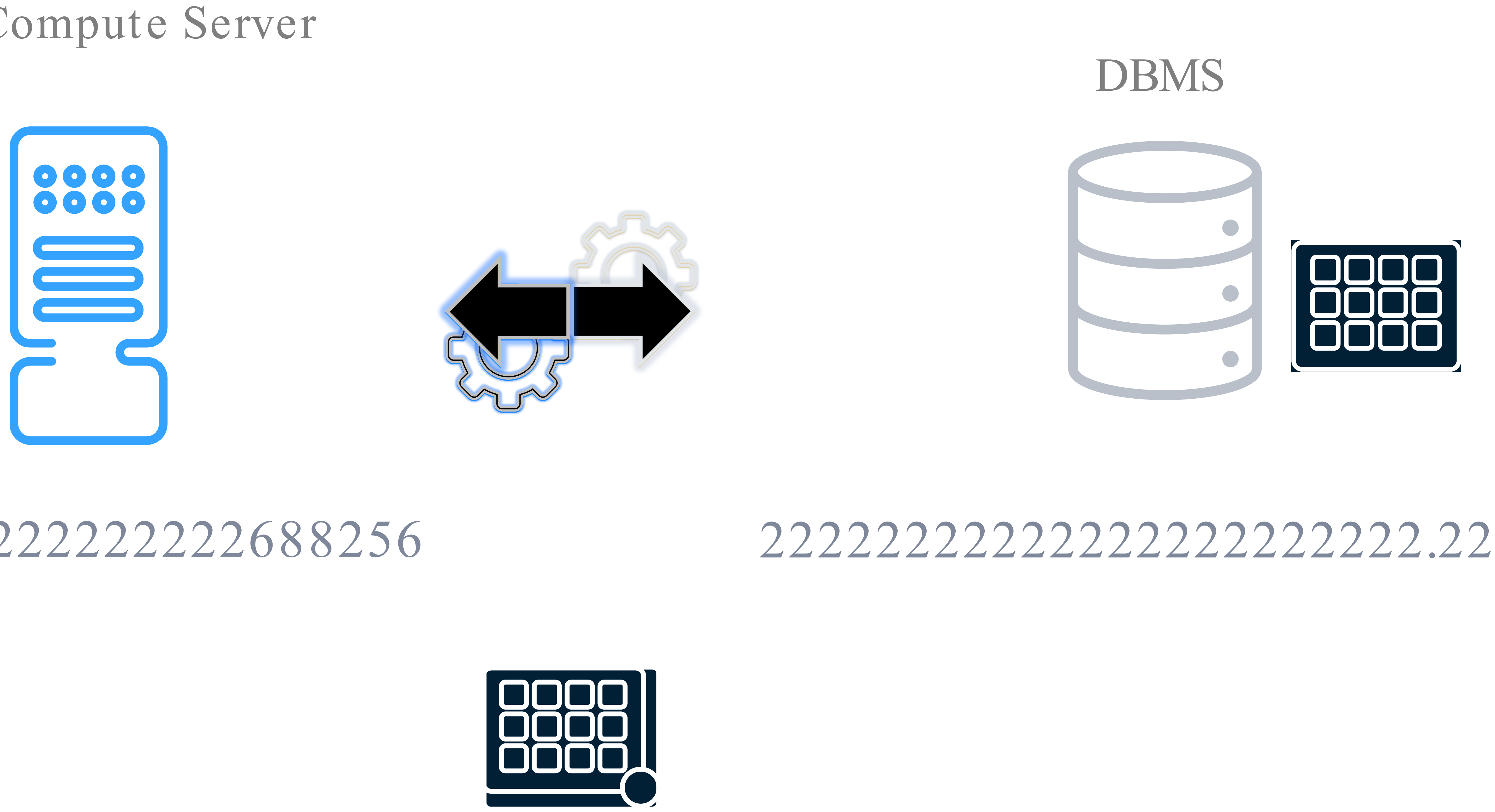
TIMESTAMP(*p*)

TINYINT

VARBINARY(*n*)

VARCHAR(*n*)

Reading Data from a Database - SAS/Access Engine Behavior



SAS Compute Server

DBMS

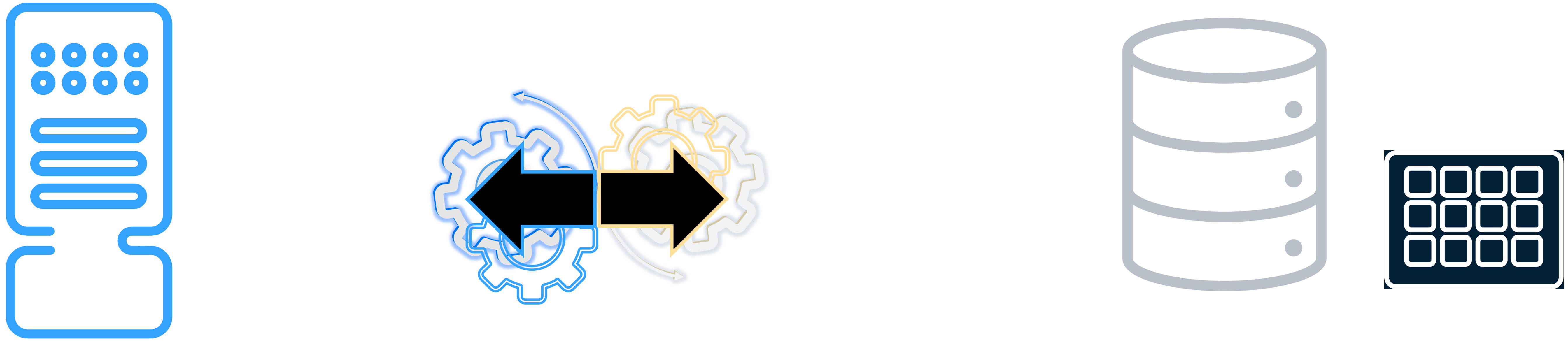
22222222222222222688256

2222222222222222222222222.22

Reading Data from a Database - DS2 Behavior

SAS Compute Server

DBMS



1234567890 12345678

1234567890 12345678

DS2 Is Not A Data Step Replacement

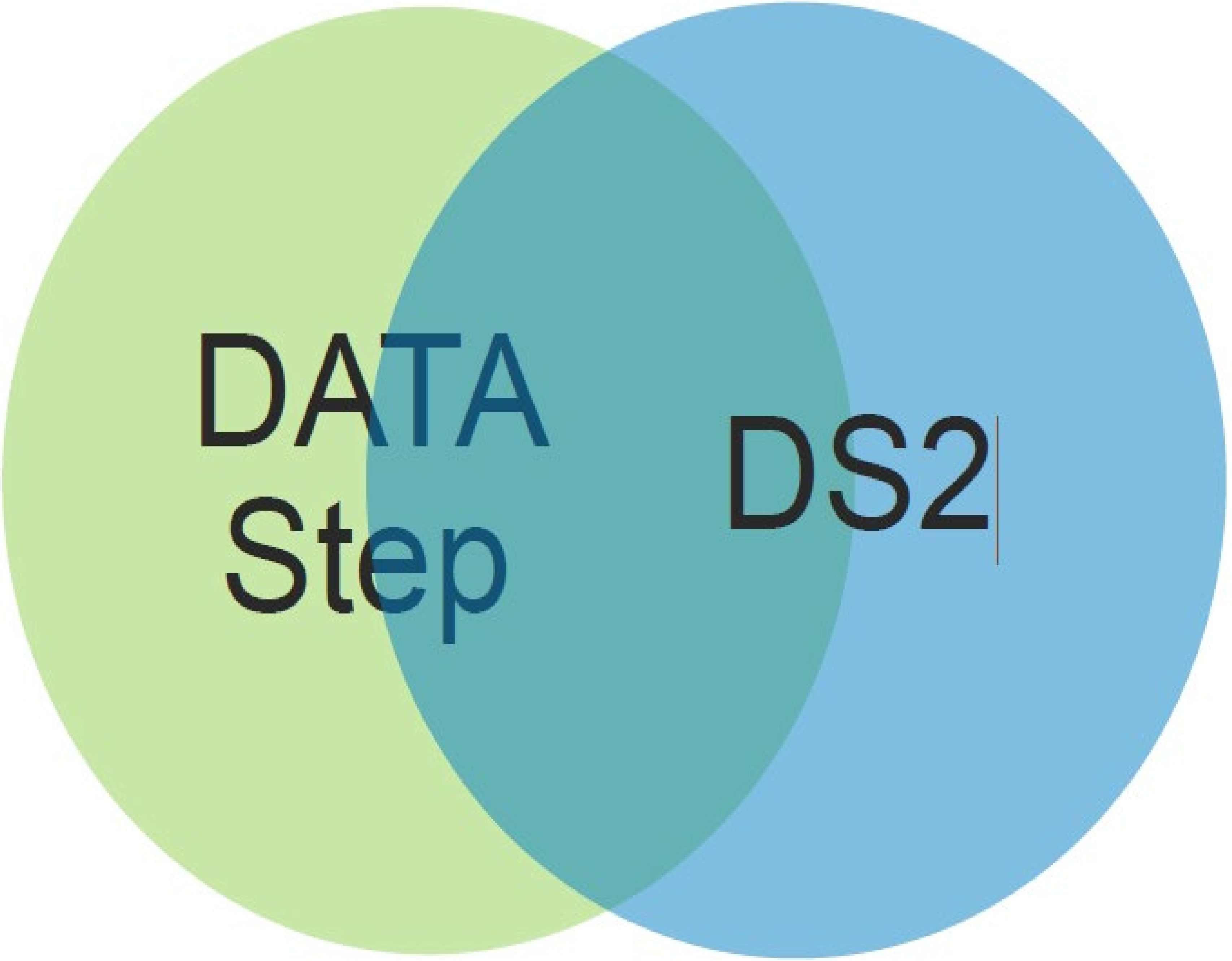
□DS2 Complements the DATA Step

DS2 has

- DATA and SET statements
- IF..THEN..ELSE, DO loops
- Expressions and Functions
- Future: Debugger

DS2 does not have

- INFILE and INPUT statement
- MERGE, UPDATE
- MODIFY / REPLACE
- POINT=



2 Understand Basic DS2 Syntax

Hello, World

DS2 Syntax

– related to the DATA step

– but different

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

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

ds02d01

A DS2 “Hello, World” Program

This demonstration illustrates a few differences between Base SAS DATA step and DS2 DATA step programs.

Idea Exchange

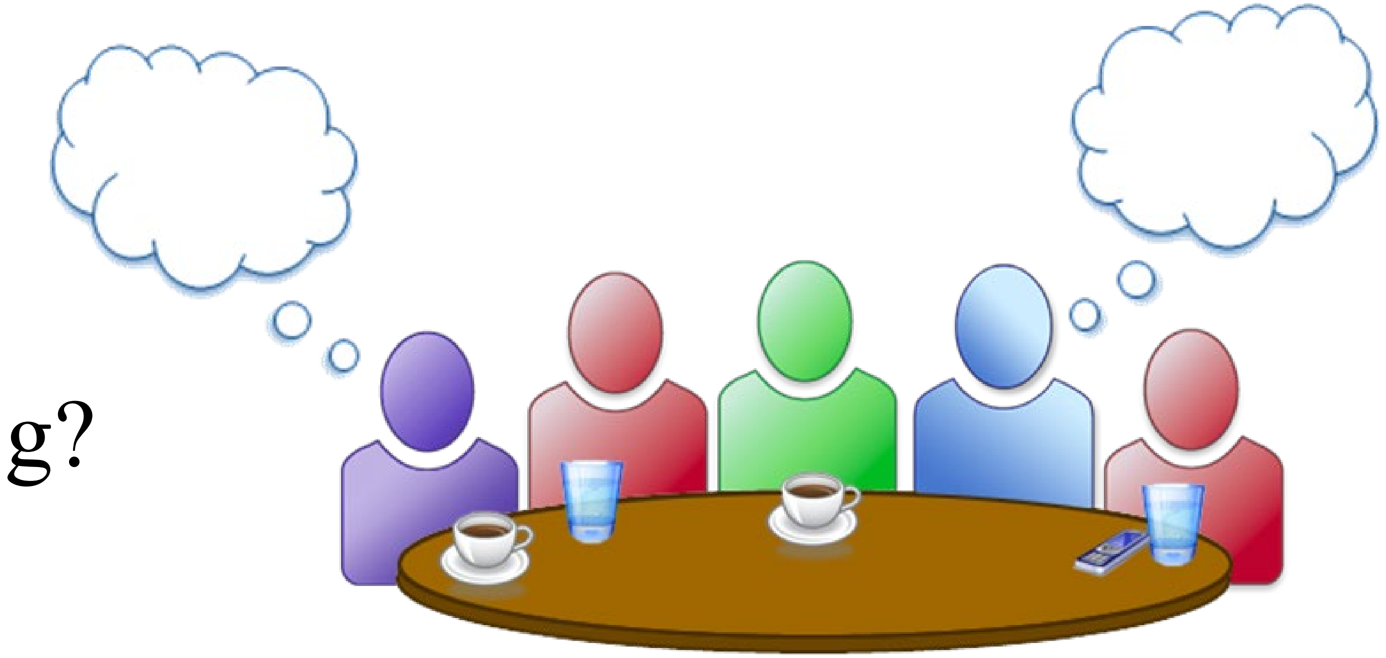
SAS Code

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

SAS Log

```
WARNING: No DECLARE for assigned-to variable Text;
         assuming type char.
```

- Have you seen this type of warning before?
- What caused the warning?



Object Based Syntax – User Defined Methods and Packages

```
PROC DS2;
data _null_;
   method c2f(double Tc) returns double;
   /* Celsius to Farenheit */
      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;
```

Define method

Call method

DS2 - Data Blocks

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

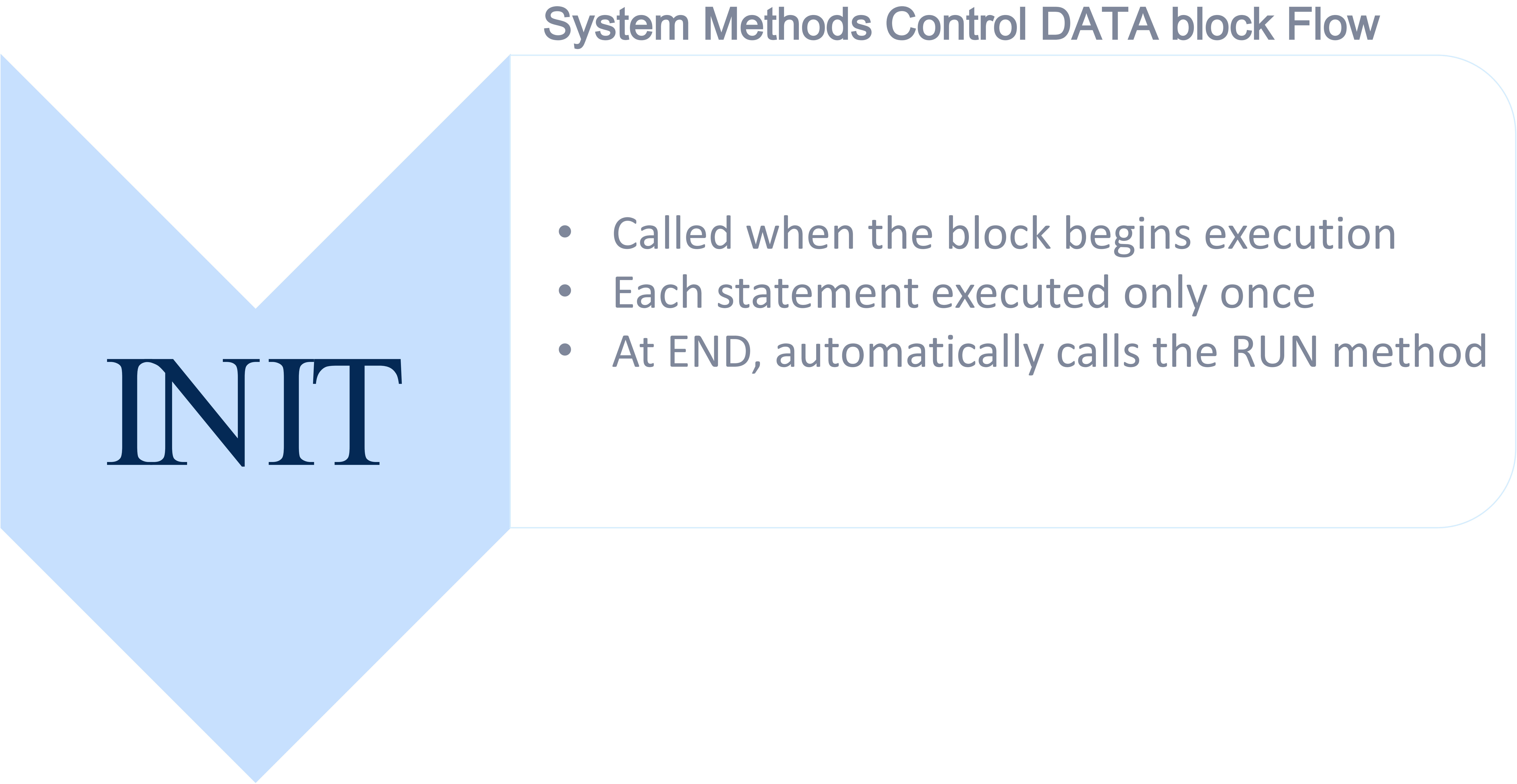
```
data my.table;






enddata;
run;
```

DS2 - Methods

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

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

INIT Method



System Methods Control DATA block Flow

INIT

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

RUN Method

RUN

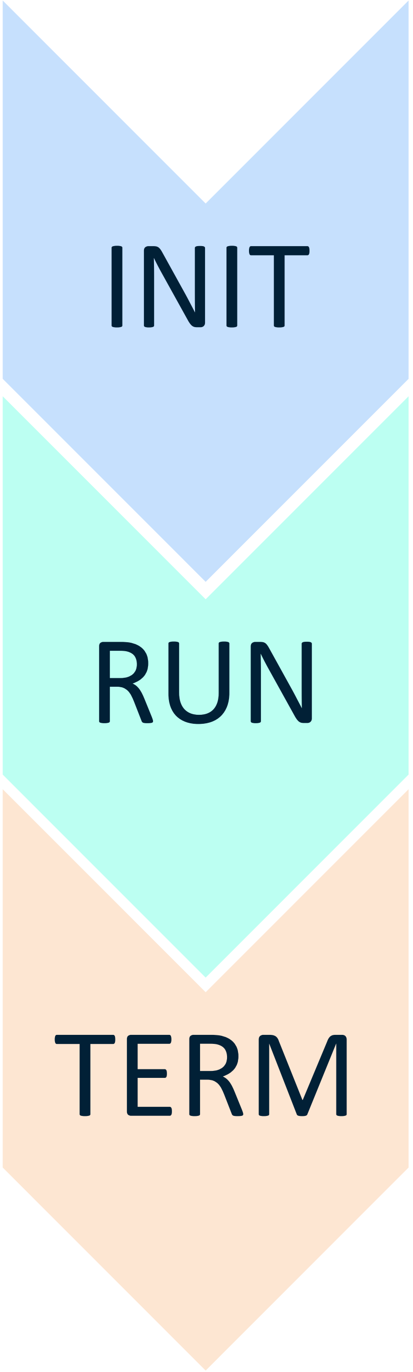
- Starts only when called by INIT
- At END, implicit OUTPUT and RETURN
- Loops until input data is exhausted
- When out of data, automatically calls TERM

TERM Method

Term

- Starts only when called by RUN
- Each statement executed only once
- END terminates program block execution

Null Method



```
data;
   method init();
   end;
   method run();
      set my.table;
   end;
   method term();
   end;
enddata;
run;
```

NULL Method

NULL Method

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;
```

SAS Data Step Declarative Statements

Variable Declaration: Variables are automatically created in the DATA step when they are first referenced, but here's an example of declaring and initializing them explicitly	data example;
	  length name $20;
	  name = 'John Doe';
	run;

Array Declaration: Arrays group variables for easier processing.	data example;
	  array scores[3] score1-score3;
	  do i = 1 to 3;
	    scores[i] = i * 10;
	  end;
	run;

Format Declaration: The FORMAT statement defines how variables should be displayed.	data example;
	  salary = 50000;
	  format salary dollar8.;
	run;

Label Declaration: Adds descriptive labels to variables for reports or displays.	data example;
	  length id $5;
	  label id = 'Employee ID';
	  id = 'E123';
	run;

Retain Declaration: Keeps the value of variables across iterations of the DATA step.	data example;
	  retain counter 0;
	  counter + 1;
	run;

Keep/Drop Declaration: Specifies variables to include or exclude in the output dataset.	data example;
	  keep age;
	  age = 25;
	run;

Informat Declaration: Assigns input formats to read raw data into the SAS dataset correctly.	data example;
	  informat birthdate date9.;
	  input birthdate $;
	  format birthdate date9.;
	datalines;
	15JAN1990
	;
	run;

SAS DS2 Declarative Statements

Variable Declaration (length): In DS2, you explicitly declare all variables using dcl (declare) instead of relying on length or implicit declarations .

```
proc ds2;
  data example;
    dcl varchar(20) name;
    dcl double age;
    method run();
      name = 'John Doe';
      age = 30;
    end;
  enddata;
run;
quit;
```

Array Declaration: DS2 supports arrays, but they are defined differently. DS2 arrays are more like fixed-size collections that don’t have to be declared as array explicitly.

```
proc ds2;
  data example;
    dcl double scores[3];
    method run();
      do i = 1 to 3;
        scores[i] = i * 10;
      end;
    end;
  enddata;
run;
quit;
```

Format Declaration: In DS2,, formatting in DS2 is applied to variables within the DECLARE statement itself, as shown below:

```
proc ds2;
  data example;
    dcl double salary format=dollar8.;
    method run();
      salary = 50000;
    end;
  enddata;
run;
quit;
```

Variable Declaration

DECLARE|DCL*data-type variable-lis*<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';
```

Variable Declaration

```
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.
```

ds2_d04

Variable Declaration

DATA Step

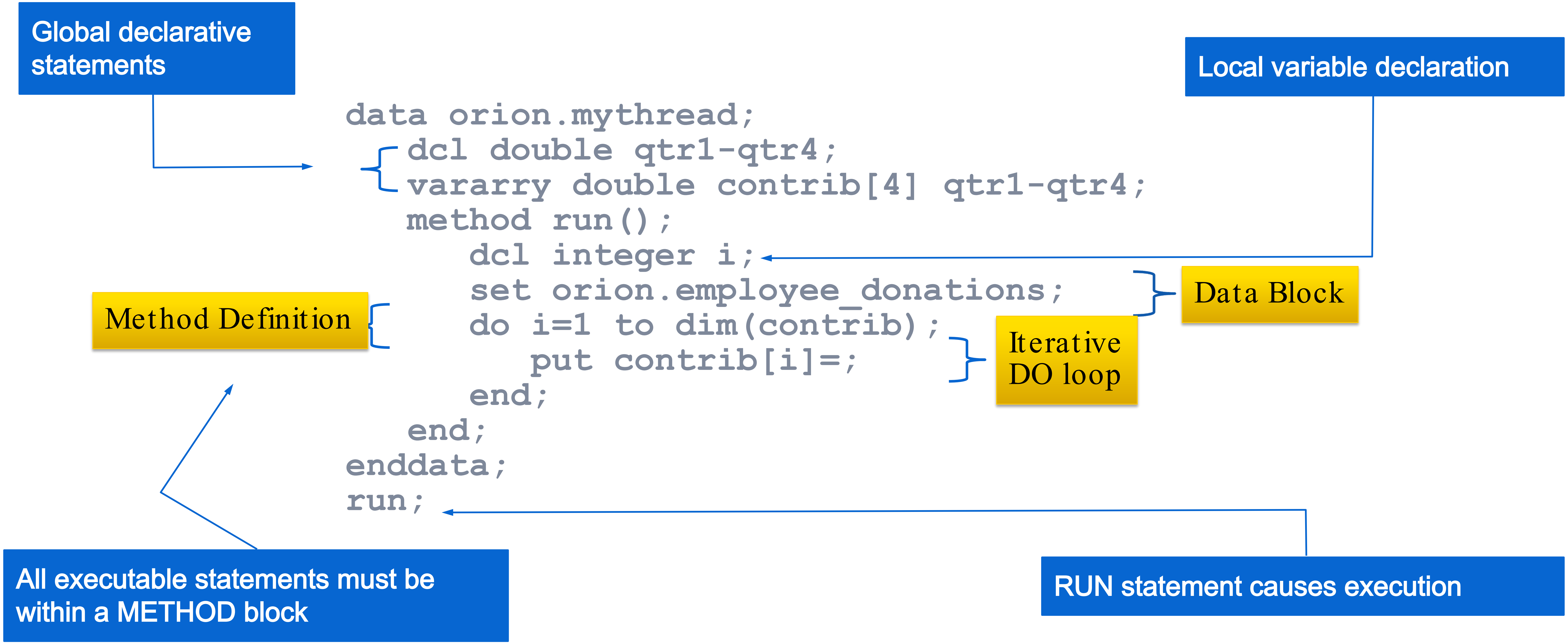
- KEEP
- DROP
- RETAIN

DS2

- KEEP
- DROP
- RETAIN

```
ERROR: Compilation error.
```

DS2 Data Block Structure



```
data orion.mythread;
   dcl double qtr1-qtr4;
   vararry double contrib[4] qtr1-qtr4;
   method run();
      dcl integer i;
      set orion.employee_donations;
      do i=1 to dim(contrib);
         put contrib[i]=;
      end;
   end;
enddata;
run;
```

Global declarative statements

Local variable declaration

Method Definition

Data Block

Iterative DO loop

All executable statements must be within a METHOD block

RUN statement causes execution

Convert a Base SAS DATA step program to DS2.

Convert a DATA step to DS2, and leverage the new programming structures and capabilities.

```
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;
```

```
proc ds2;
data _null_;








enddata;
run;
quit;
```

ds02d05

Quiz

Which DS2 system method should be used to execute sections 1, 2, and 3 of this DATA step?

```
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;
```

Section	System Method
	INIT
	RUN
	TERM

Quiz – Correct Answer

Which DS2 system method should be used to execute sections 1, 2, and 3 of this DATA step?

```
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;
```

Section	System Method
1	INIT
2	RUN
3	TERM

Converting to DS2

Converting Section 1

```
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;
```

```
proc ds2;
data _null_;
   method init();
      Text='**> Starting';
      put Text;
   end;









enddata;
run;
quit;
```

ds02d05

Converting to DS2

Converting Section 2

```
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;
```

```
proc ds2;
data _null_;
   method init();
      Text='**> Starting';
      put Text;
   end;

   method run();
      set orion.banks;
      put _all_;
   end;






enddata;
run;
quit;
```

ds02d05

Converting to DS2

Converting Section 3

```
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;
```

```
proc ds2;
data _null_;
   method init();
      Text='**> Starting';
      put Text;
   end;

   method run();
      set orion.banks;
      put _all_;
   end;

   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
quit;
```

ds02d05

Converting to DS2

- Partial SAS log

```
**> Starting
NAME=Carolina Bank and Trust     Text=**> Starting _N_=1 RATE=0.0318
NAME=State Savings Bank          Text=             _N_=2 RATE=0.0321
NAME=National Savings and Trust  Text=             _N_=3 RATE=0.0328
**> All done
WARNING: No DECLARE for assigned-to variable text; assuming type char.
NOTE: Execution succeeded. No rows affected.
```

The variable **Text** is not desired in the RUN method output.

ds02d05

Setup for the Poll

What is the scope of the undeclared variable **Text**?

```
proc ds2;
data _null_;
   method init();
      Text='**> Starting';
      put Text;
   end;

   method run();
      set orion.banks;
      put _all_;
   end;

   method term();
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
quit;
```

ds02d05

Poll

What is the scope of the undeclared variable Text?

a. local

b. global

c. cannot be determined

Poll – Correct Answer

- What is the scope of the undeclared variable **Text**?

  a.local

  [✓ b.global]

  c.cannot be determined

- **Undeclared variables have a global scope.**

Converting to DS2

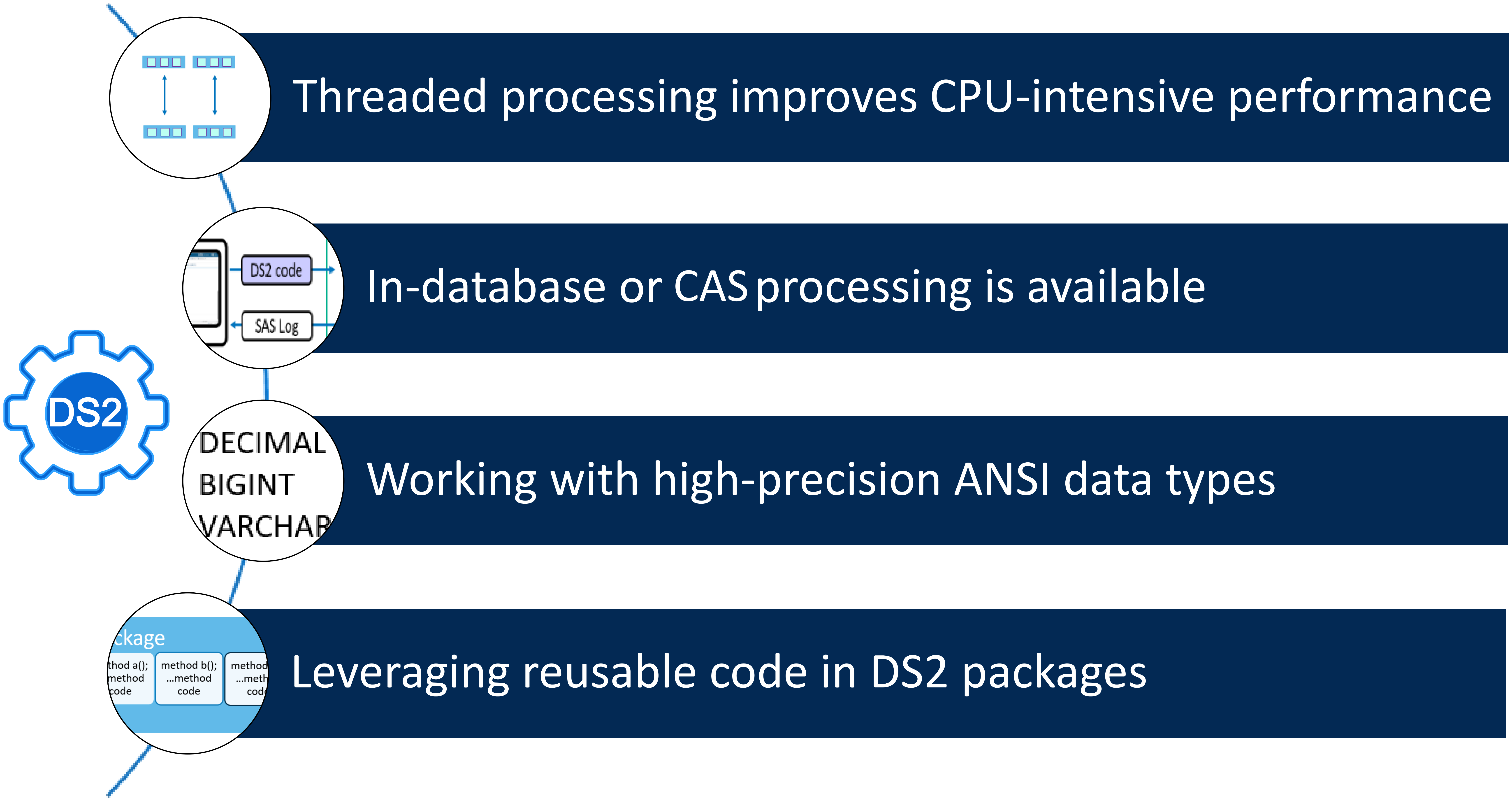
Declaring the variable **Text** as local excludes it from the PDV.

```
proc ds2;
data _null_;
   method init();
      dcl char(15) Text;
      Text='**> Starting';
      put Text;
   end;
   method run();
      set orion.banks;
      put _all_;
   end;
   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
run;
enddata;
quit;
```

ds02d05

3 Know When To Use DS2

When to Use DS2



Threaded processing improves CPU-intensive performance

In-database or CAS processing is available

DECIMAL
BIGINT
VARCHAR

Working with high-precision ANSI data types

Leveraging reusable code in DS2 packages

Idea Exchange

What real-world benefit do you see resulting from converting this particular program from a Base SAS DATA step to a DS2 DATA step?

```
proc ds2;
data _null_;
   method init();
      dcl varchar(20) Text;
      Text='**> Starting';
      put Text;
   end;
   method run();
      set orion.banks;
      put _all_;
   end;
   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
quit;
```

Quiz

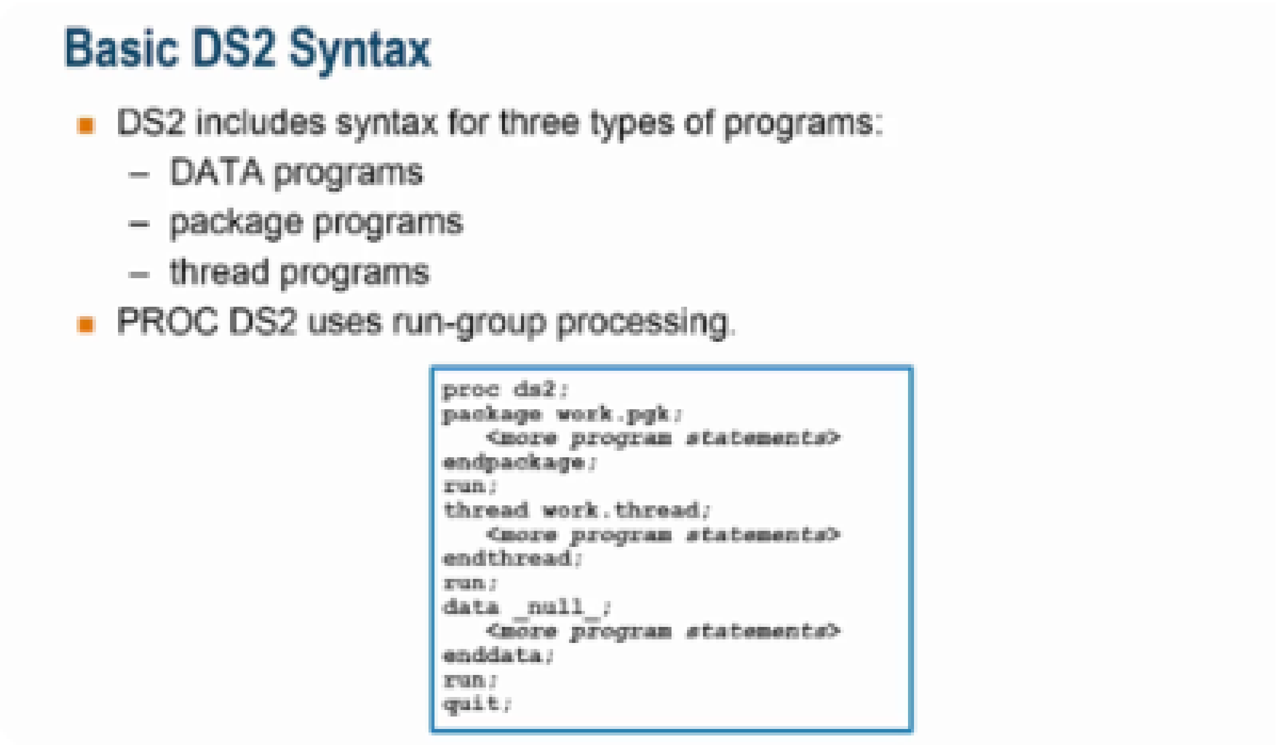
Sharpen your Pencils

Question 1

PROC DS2 creates which types of programs?

A. DATA, Package and Thread

B. DATASTEP, Package and Multi-Thread



Basic DS2 Syntax

- DS2 includes syntax for three types of programs:
  - DATA programs
  - package programs
  - thread programs
- PROC DS2 uses run-group processing.

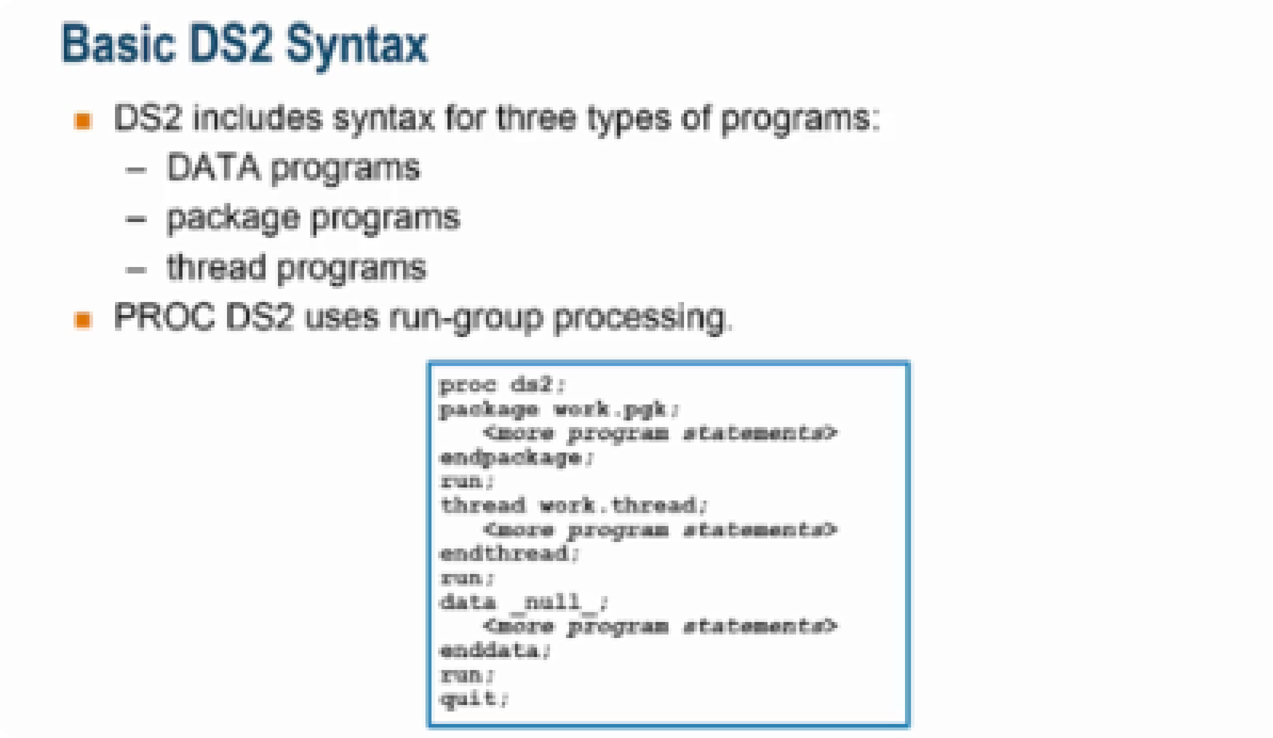
```
proc ds2;
package work.pgk;
   <more program statements>
endpackage;
run;
thread work.thread;
   <more program statements>
endthread;
run;
data _null_;
   <more program statements>
enddata;
run;
quit;
```

Answer 1

PROC DS2 creates which types of programs?

[✓ A.] DATA, Package and Thread

B. DATASTEP, Package and Multi-Thread



Basic DS2 Syntax

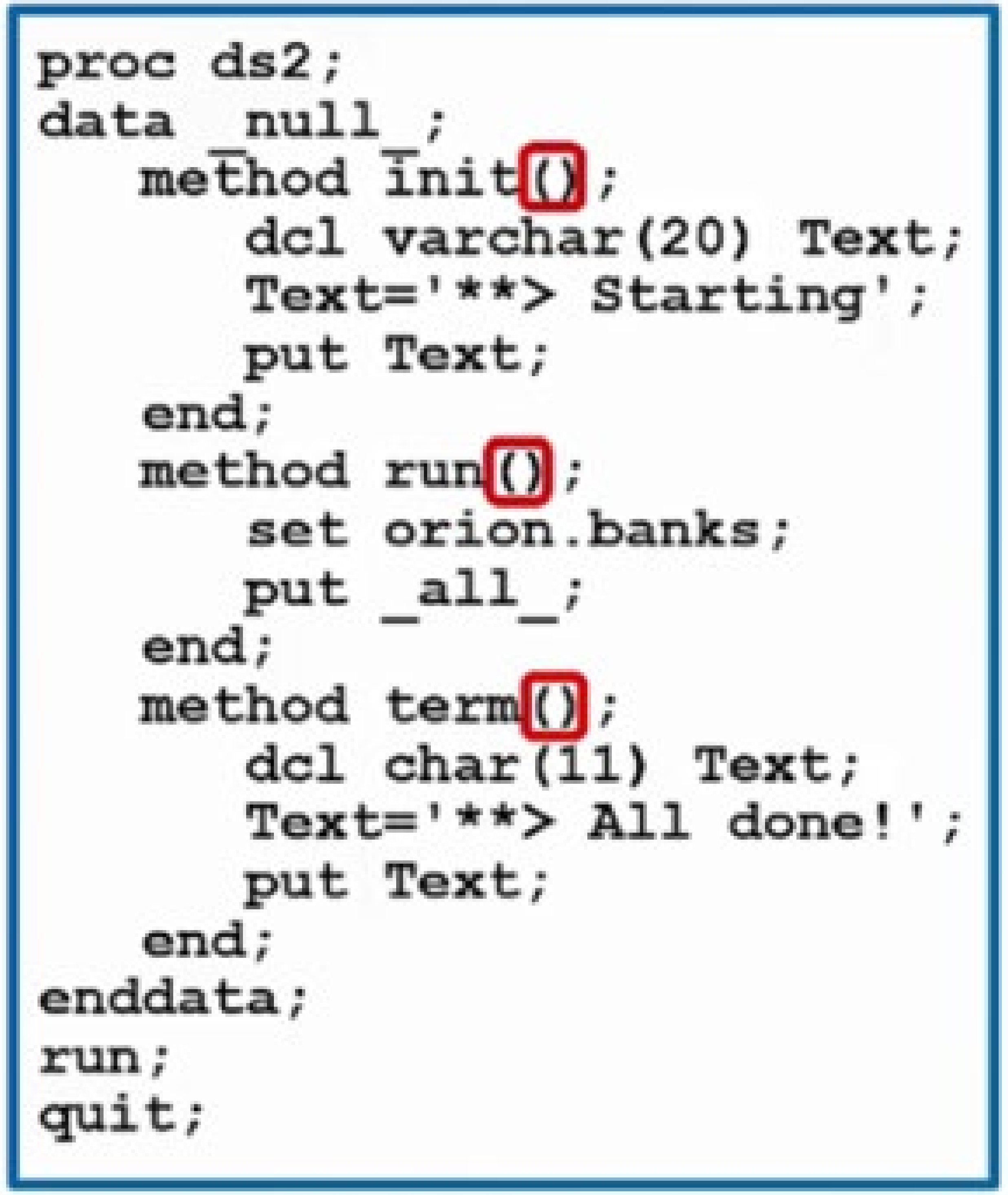
- DS2 includes syntax for three types of programs:
  - DATA programs
  - package programs
  - thread programs
- PROC DS2 uses run-group processing.

```
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;
```

Question 2

How many methods are required to run the code in the image below?

A. 1
B. 3
C. 0
D. None of the above



```
proc ds2;
data _null_;
   method init();
      dcl varchar(20) Text;
      Text='**> Starting';
      put Text;
   end;
   method run();
      set orion.banks;
      put _all_;
   end;
   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
quit;
```

Answer 2

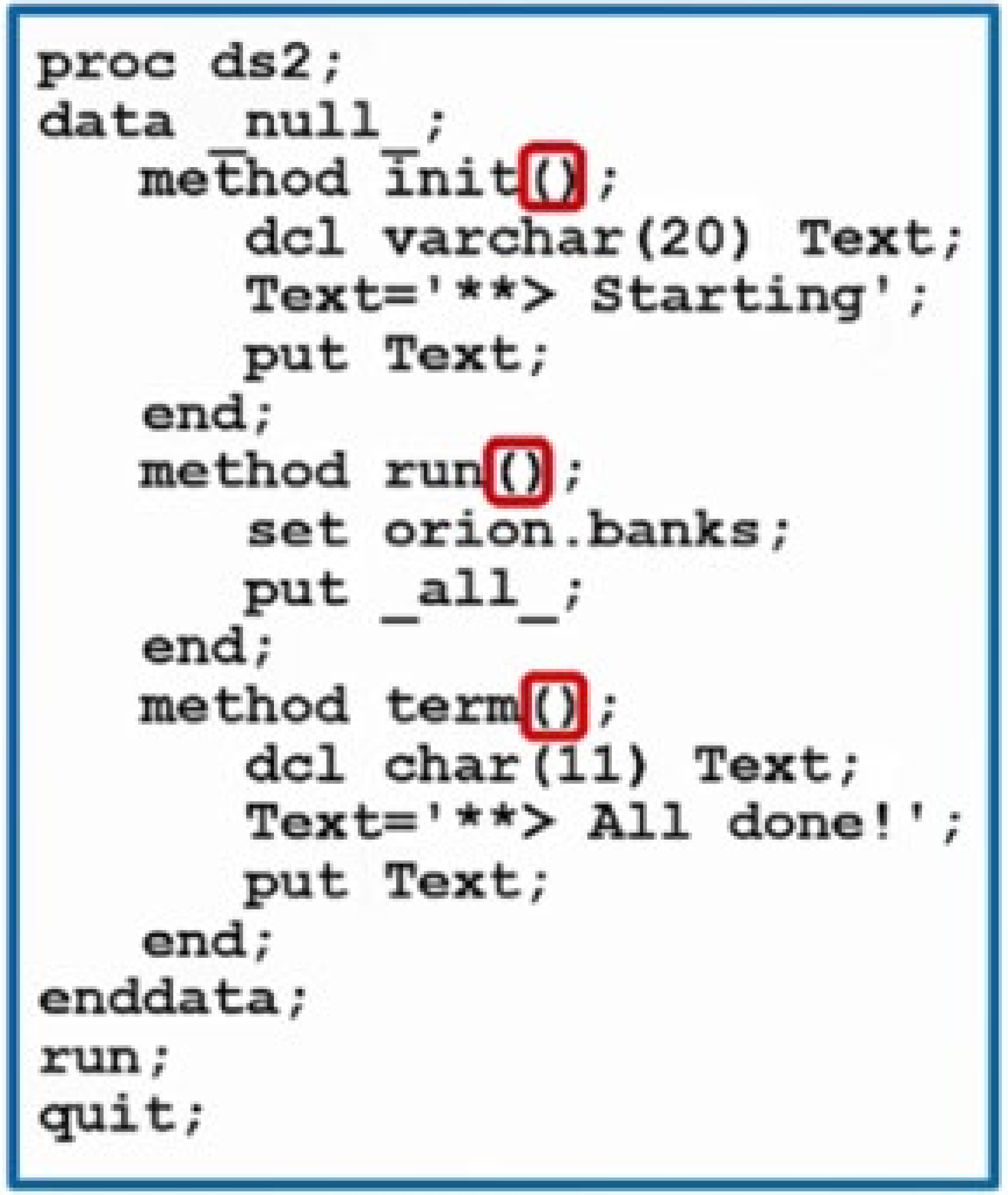
How many methods are required to run the code in the image below?

[✓ A.] 1

B. 3

C. 0

D. None of the above



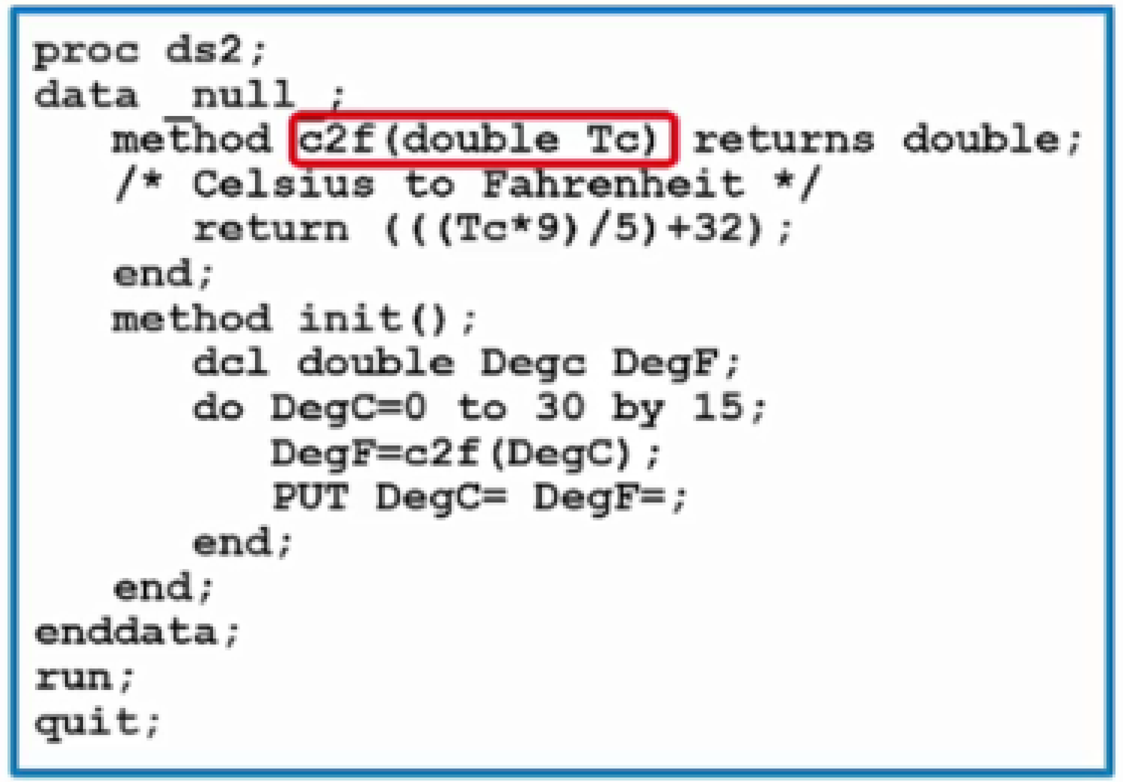
```
proc ds2;
data _null_;
   method init();
      dcl varchar(20) Text;
      Text='**> Starting';
      put Text;
   end;
   method run();
      set orion.banks;
      put _all_;
   end;
   method term();
      dcl char(11) Text;
      Text='**> All done!';
      put Text;
   end;
enddata;
run;
quit;
```

Question 3

What type of method is defined below in the image?

A. System

B. User Defined



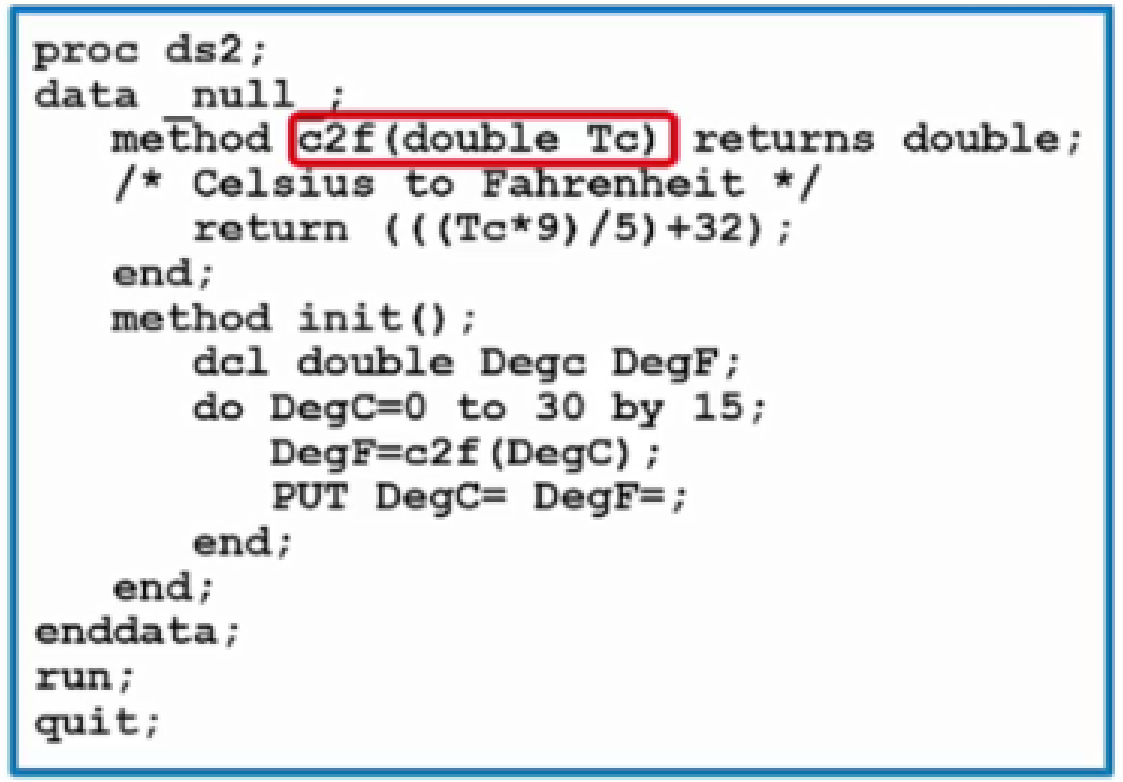
```
proc ds2;
data _null_;
   method c2f(double Tc) returns double;
   /* Celsius to Fahrenheit */
      return (((Tc*9)/5)+32);
   end;
   method init();
      dcl double Degc DegF;
      do DegC=0 to 30 by 15;
         DegF=c2f(DegC);
         PUT DegC= DegF=;
      end;
   end;
enddata;
run;
quit;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

Answer 3

What type of method is defined below in the image?

A. System

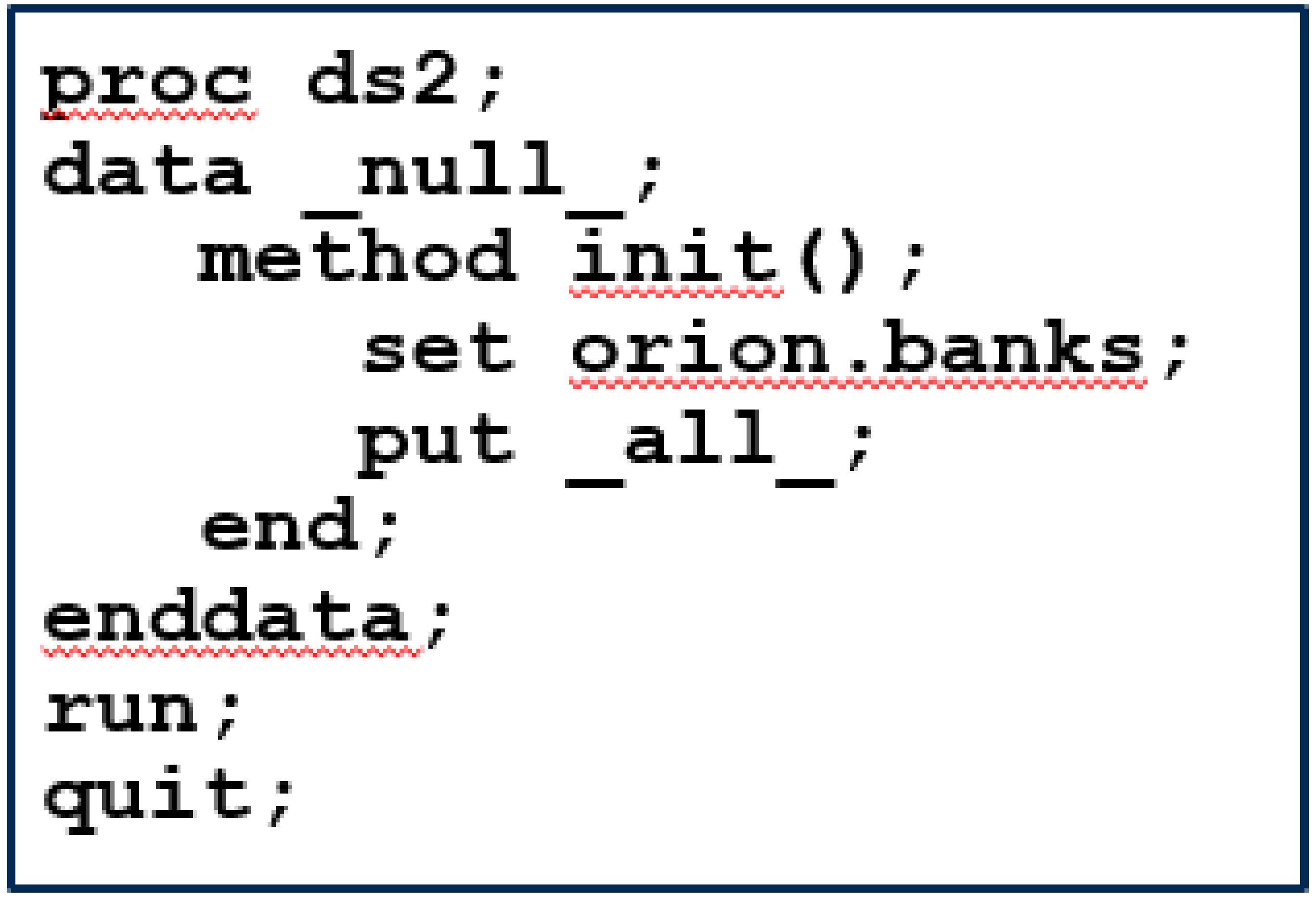
[✓ B. User Defined]



```
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;
```

Question 4

This DS2 DATA step INIT method contains a SET statement that reads orion.banks. If the data set contains three observations, how many times is the method executed?



```
proc ds2;
data _null_;
   method init();
      set orion.banks;
      put _all_;
   end;
enddata;
run;
quit;
```

a. 0

b. 1

c. 3

d. cannot be determined from the information given

Answer 4

This DS2 DATA step INIT method contains a SET statement that reads **orion.banks**. If the data set contains three observations, how many times is the method executed?

```
proc ds2;
data _null_;
   method init();
      set orion.banks;
      put _all_;
   end;
enddata;
run;
quit;
```

a. 0

[✓ b.] 1

c. 3

d. cannot beetermined from the information given

The INIT system method automatically executes only once, when the DS2 DATA step first begins execution.

✓ Did you enjoy this session, Let us know in the evaluation

Charu Shankar

EMAIL	Charu.shankar@sas.com
BLOG	https://blogs.sas.com/content/author/charushankar/
TWITTER	CharuYogaCan
LINKEDIN	https://www.linkedin.com/in/charushankar/

sas®