SAP ABAP Introduction

What is ABAP?

ABAP initially abbreviated as Allgemeiner Berichts-Aufbereitungs-Prozessor, which is a German for "generic report preparation processor". It was later converted to the English and change name as “Advanced Business Application Programming”.

ABAP is fourth-generation languages (4GLs) originally first established in the 1980s. ABAP was initially developed as a report language for SAP R/2 platform that was used by large organizations to develop mainframe business applications for materials handling, financial and accounting.

The ABAP language was used by developers to develop the SAP R/3 platform. It was also planned to be used by SAP customers to enhance SAP applications. The language was promoted towards more technical customers with programming experience.

Why ABAP?

- ABAP supports both procedural and Object-oriented programming.
- ABAP aimed to use by SAP customers to enhance SAP applications.
- ABAP allows us to develop custom reports and interfaces with ABAP programming.
- ABAP supports portability and can easily ported from one platform to another.
- ABAP is easy for the people who are coming from programming background.
- ABAP was one of the first languages to include the concept of Logical Databases (LDBs), which supports every platform, language, and units.
- ABAP used by developers to develop the SAP R/3 platform.

Specify the ABAP version history?

- ABAP is one of the 4GLs (fourth-generation languages) first developed in the 1980s.
- ABAP becomes as the language for developing programs for the client–server R/3 system, which SAP was first released in the year 1992.

- In 1999, SAP released an object-oriented extension known as ABAP Objects along with R/3 release 4.6.
- In 2001, all but the most basic functions were written in ABAP.
- In 2006, 7.0 version released that offer switch framework feature.
- In 2012, 7.4 version released with Table expressions feature.
- In 2015, 7.5 version released that start supporting Open SQL expressions.
- In 2017, 7.52 version released with virtual sorting of the internal table feature.
- In 2019, 7.54 (current) version released with many revisions of existing features.

What is ABAP Runtime Environment?

The runtime system is a part of SAP kernel controls the ABAP programs execution. The runtime system is responsible for processing ABAP statements, controlling the screens flow logic and responding to events (For example - a user clicking on a button existed on the screen).

A key component of the ABAP runtime system is the Database Interface, that converts database-independent ABAP statements ("Open SQL") into statements understood by the underlying DBMS ("Native SQL"). The database interface manages all the communication with the relational database on behalf of ABAP programs. Database interface also contains features such as buffering of tables and frequently accessed data in the local memory of the application server.

How ABAP program stored?

All ABAP programs exists inside the SAP database and they are not stored as a separate external file like Java or C++ programs.

In the database, each ABAP program code stores in two forms - **Source code** and **Generated code**.

Source code can view and edited with the ABAP Workbench tools.
Generated code is a binary representation that is slightly similar with Java bytecode.

What are the types of ABAP programs-based execution?

An ABAP program is either an executable unit or a collection, that offers reusable code to other programs. ABAP programs are basically two types -

- **Executable programs –** These are independently executable.
- **Non executable programs –** These are not independently executable.

Executable programs –

ABAP divides **executable programs** are two types. Those are -

- Reports
- Module pools

Reports follows a simple programming model where a user enters a set of input parameters optionally and the program uses those input parameters to produce a report in the form of an interactive list. Due to the "list-oriented" nature of the program output, these programs called as “reports”.

Module pools explains challenging ways of user interaction using a collection of screens. The term “screen” specifies the actual, physical image that the user can see. Each screen has a "flow logic", that calls to the ABAP code indirectly. This flow logic is divided into two sections – one is "PBO" (Process Before Output) and another is "PAI" (Process After Input). The combination of the screen and its flow logic is called as “dynpro” (dynamic program).

Non-executable programs –

The **non-executable program** types are divided as below -

- **INCLUDE modules –** Includes into calling unit during the generation time and mostly used to split the larger programs.
- **Subroutine pools -** Contain ABAP subroutines that are coded in between FORM/ENDFORM and invoked with PERFORM.

- **Function groups -** Are libraries of function modules that are coded in between FUNCTION/ENDFUNCTION and invoked by CALL FUNCTION).
- **Object classes –** Defines set of methods and attributes.
- **Interfaces –** Contains “empty” method definitions to manually provide the code by developer.
- **Type pools -** Defines set of data types and constants.

How to create development environment for ABAP?

The ABAP development can be done in two possible days. Those are –

- ABAP Workbench
- ABAP Development Tools

The **ABAP Workbench** is part of the ABAP system and accessed via SAP GUI. It contains various tools for editing programs. Some of the ABAP workbench tools are -

- **ABAP Editor** used for writing and editing reports, module pools, includes and subroutine pools. Transaction code is SE38.
- **ABAP Dictionary** used for processing database table definitions and retrieving global types. Transaction code is SE11.
- **Menu Painter** used for designing the user interface like menu bar, standard toolbar, application toolbar, function key assignment. Transaction code is SE41.
- **Screen Painter** used for designing screens and flow logic. Transaction code is SE51.
- **Function Builder** used for function modules. Transaction code is SE37.
- **Class Builder** used for ABAP Objects classes and interfaces. Transaction code is SE24.

The **Object Navigator** provides a single integrated interface into these various tools. Transaction code is SE80.

The **ABAP Development Tools (ADT)** also known as "ABAP in Eclipse" with a set of plugins to develop ABAP.

SAP ABAP R/3 Architecture -

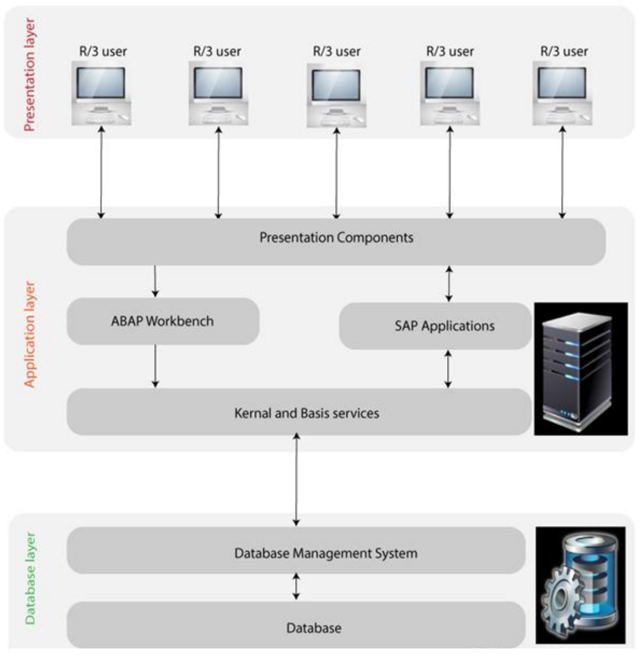
What are the views of R/3 System?

The R/3 system is the platform for all other applications in the R/3 System. In R/3, R stands for Real-time and 3 stands for 3-tier architecture. The R/3 System can have different set of views in different prospects. The main views among them are -

- Logical view
- Software Oriented View

What is the logical view of R/3 System?

The below diagram represents the logical view of the R/3 system -



The difference between “logical view” and "hardware or software-based view” is, not all the above components assigned to a specific hardware or software unit. The above diagram explains how the R/3 basis system constructs a central platform within the R/3 System. The tasks of three logical components tasks of the R/3 Basis system are described below -

Kernel and Basis Services –

The kernel and basis services is a runtime environment for all R/3 applications (i.e. hardware, operating system, and database specific). The runtime environment is mainly written in C and C++. However, some parts are also written in ABAP.

The tasks of the kernel and basis services are - “running applications”, “user and process administration”, “database access”, “communication” and “system monitoring and administration”.

ABAP Workbench –

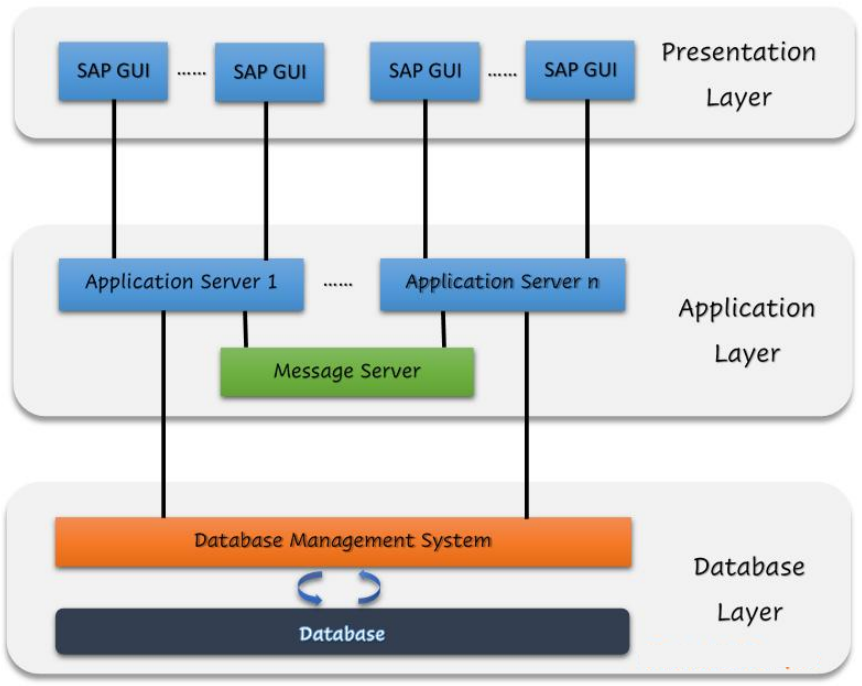
The ABAP Workbench is a complete development environment for applications in the ABAP language. We can create, edit, test, and manage application developments using workbench. Workbench is itself written in ABAP and is fully integrated in the R/3 system.

Presentation Components –

The presentation components are responsible for the interaction between the R/3 System and the user.

What is Software Oriented View of R/3 system?

The software-oriented view specifies the different software components (i.e. all the SAPgui components and application servers) that produce R/3 system. The below diagram represents a software-oriented view of the R/3 System.



SAP R/3 system is a multi-tier client-server architecture. The SAP R/3 architecture is basically divided into three layers and those are -

- Presentation layer
- Application layer
- Database layer

Presentation layer –

Presentation layer is an interface between the R/3 system and its end-users. Presentation layer is a distributed to the workstations of end users. The end users can be client or customer or individual users.

Presentation layer receives the data from the input devices and sends it to the application layer (server) to process the data. And, responsible to receive the processed data from the application layer and sends the same to the devices where the result requires to display.

The data or input can be entered by using the input devices. The input devices can be browsers, mobile devices, font-end GUI systems and so on.

The presentation layer is normally distributed across several presentation servers. The presentation layer analyzes user inputs and transfers these to the application layer.

Application layer –

The application layer is an intermediate layer. This layer process the inputs from the presentation layer and one messaging server. The application layer is implemented using application servers. Application server(s) can be one or multiple. Each Application server can contain either one or multiple services to run an R/3 system.

It communicates to the database layer with the input received from the presentation layer, processes input and sends the results back to the presentation layer. The application layer contains the ABAP runtime environment where the ABAP programs are executed.

All the services are distributed to the more than one application server. Messaging server contains the information about the application groups and load balancing information. Messaging server is responsible for the communication between the existing application servers.

Messaging server is responsible for -

- Passes the requests from one application server to another.
- Assigns appropriate application server when users logged in.

Database layer –

Database layer is layer contains the central database system of all the data in R/3 system. Normally, this database layer installs on the different server due to the security, performance and processing issues.

This layer is responsible for processing the requests from the application layer and sends back the required information. In general, either SQL or ORACLE servers can be used.

All R/3 data stores in the database. Database layer contains two components -

- Database Management Systems
- Database

The communication flow happens from the presentation layer to database layer and vice versa. Technical distribution of the system is independent of the layers. I.e., All the three layers can be installed on the same machine or can be installed on different individual machines.

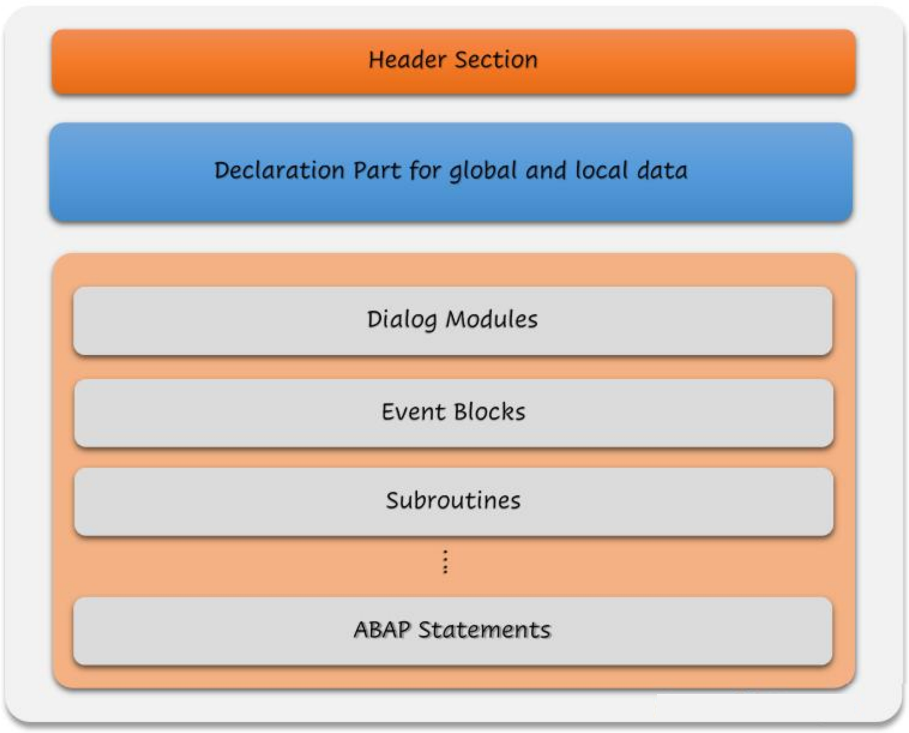
SAP ABAP Programs Structure -

What is the ABAP program Structure?

ABAP programs are mostly responsible for data processing within the different blocks of an application program. i.e. program cannot create as a single unit however program should divide into smaller modules that are assigned to individual blocks.

Each module (set of statements) in the program is called as a processing block. The set of processing blocks gets executed when we run the program. The processing blocks cannot be nested. A processing block contains a set of ABAP statements.

The following diagram shows the structure of an ABAP program –



Each ABAP program consists of below two parts -

- Header section
- Declaration Part for Global Data, Classes and Selection Screens
- Processing Blocks container

Header section –

Header section contains the detailed information about the program. Header section is standard template that is generated when we create the program and it is editable. The ABAP program first statement starts with PROGRAM or REPORT. The header section is mandatory.

The first statement automatically inserted by system when creating a program –

- For module pools – **PROGRAM <program-name>**
- For executable pools – **REPORT <report-name>**

Declaration Part for Global Data, Classes and Selection Screens –

Declaration part for global data, classes, and selection screens is the first part of ABAP program. This part consists of – all declaration statements. i.e. declaration statements for global data, all selection screen definitions, all class definitions, and declaration statements in procedures (methods, subroutines, function modules).

Processing Blocks Container –

Processing block container of an ABAP program contains all processing blocks for the program.

The ABAP program is combination of different processing blocks. Each processing block contains the business logic and its declaration. The processing blocks of ABAP programs are –

- Dialog Modules
- Event blocks for selection screens
- Event blocks for lists
- Event blocks for executable programs/Reports
- Subroutines
- Function Modules
- Methods
- All ABAP statements (except the declarative statements in the declaration part)

Let us discuss them one by one in detail -

What is Dialog module?

Dialog module is used to code a business logic that relates to screens flow. Dialog modules can be called from the screen flow logic/screen command logic.

Dialog modules are coded in between the MODULE and ENDMODULE statements.

Fields should have same name on both dialog screen and related ABAP program and data can be passed between the matching named fields in the program.

What is Event block for selection screens?

A selection screen is a type of dialog. The different events in a selection screen like PAI, PBO, user input is controlled by a selection screen processor. The selection screen processor controls the flow logic of the selection screen.

What is Event block for Lists?

Lists are special screens that outputs arranged data. We can create list in any processing block of an ABAP program using a set of commands such as WRITE, NEW-PAGE and so on. The list processor displays the list on the screen and handles user actions within lists. The list processor controls the flow logic of the list.

What is Event block for Executable Programs (Reports)?

Runtime environment controls the execution of program (type 1). Type 1 programs are event driven.

What is Subroutines?

A subroutine is a reusable section of code. We can define subroutines in any ABAP program and can call subroutines using the PERFORM statement from ABAP programs. Subroutines are defined using the FORM statement and coded in between FORM…ENDFORM statement.

What is Function Modules?

Function modules are external functions with a well-defined interface. We can call function modules from ABAP programs using the CALL FUNCTION statement.

Function modules are defined using the FUNCTION statement and coded in between FUNCTION...ENDFUNCTION statement.

What is Method?

Methods define the functions of classes in ABAP Objects. Methods have a defined interface. We can call methods from ABAP programs using the CALL METHOD statement.

Methods are defined using the METHOD statement and coded in between the METHOD…ENDMETHOD statement.

What is the ABAP Statement?

The ABAP statements that are not part of declaration part is called as source code.

The ABAP program source code consists of comments and ABAP statements. The ABAP statements and comments can be discussed in further chapters.

What are the Naming conventions for ABAP programs?

Below are some rules for naming ABAP programs -

- An ABAP program name can be in between **1** and **30** characters long.
- Program name cannot contain any of the characters - **Period (.)**, **comma (,)**, **space ()**, **parentheses ()**, **apostrophe (‘)**, **inverted commas (")**, **equals sign (=)**, **asterisk (*)**, **accented characters** or **German umlauts (à, é, ø etc.)**, **percentage signs (%)**, or **underscores (_)**.
- User defined programs should start with **“Y”** or **“Z”**.
- Program should not start with **"A"** or **"X"** that are reserved for SAP Programs.

What are the requirements to create a program in real time?

- **Program name –** Every user-defined program should start with **“Y”** or **“Z”**. Check the naming standards in the project and get the name approved from the client or business analyst.
- **Program Type –** Once the name approved, understand the type of program suitable for. The available program types are – executable program, include program, module pool, function group, subroutine pool, interface pool, class pool, type pool or XSLT program.
- **Package –** Each requirement is scheduled to certain release in the real time project. Each release has its set of packages (may be one or more than one). Check with the lead for appropriate package and assign the package while saving the program.

What are the characteristics of ABAP program?

- ABAP statements begins with an ABAP keyword and ends with a period(.).
- The words (keywords, variables, operators etc.) are always separated by at least one space. If words have more than one space in between, it will be considered as one.
- ABAP statement can be more than one line long however it is not recommended. ABAP supports a single line contain more than one ABAP statement.
- There are no rules for ABAP statement starting position or indentation. ABAP statement can start in any column of the line.
- ABAP statement can be in any case (i.e. either lower case or upper case). Both are considered as same and no differentiation between upper and lower case for keywords, additions, and operands.
- ABAP program is only case sensitive while comparison.

SAP ABAP Statements -

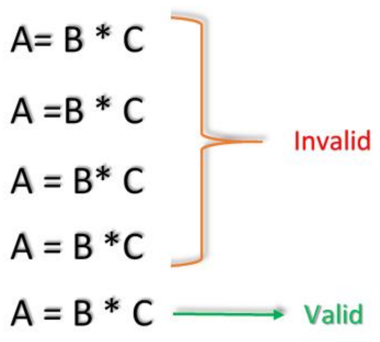
ABAP programs are written with individual sentences (statements) and each sentence or line of code in the ABAP program is called as ABAP statement. In other words, statement is a combination of keywords, operands, operators, and expressions and so on.

Below are some characteristics about ABAP statements -

- ABAP statements begins with an ABAP keyword and ends with a period(.).
  For example -

```
Write 'Hello World, Welcome to KIIT'.
```

-
- The words (keywords, variables, operators etc.) are always separated by at least one space. If words have more than one space in between, it will be considered as one.
  For example -



- ABAP statement can be more than one line long however it is not recommended. ABAP supports a single line contain more than one ABAP statement.
  For example -

Write 'Hello World, Welcome to KIIT'.

-
- There are no rules for ABAP statement starting position or indentation. ABAP statement can start in any column of the line.
  For example -

```
  Write
            'Hello World, Welcome to KIIT'.
```

-
- ABAP Statements are not case-sensitive. ABAP statement can be in any case (i.e. either lower case or upper case). Both are considered as same and no differentiation between upper and lower case for keywords, additions, and operands. ABAP considers all cases (upper, lower, and mixed cases) statements are valid and does not throws any error.
  For example, all the statements below are considered as valid even though they are in mixed case.

Write 'This statement is valid'.wRItE 'This statement is valid'.WRITE 'This statement is valid'.write 'This statement is valid'.WritE 'This statement is valid'.

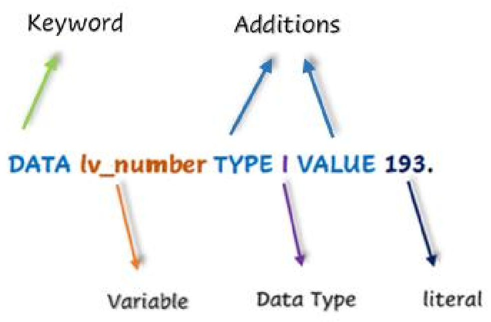
-

- ABAP statements are formed with keywords, variables, data types and objects.
  For example, -

DATA lv_number TYPE I VALUE 193.

-

Let us take an example of ABAP statement coded in the declaration part of program.



The above ABAP statement is used to declare variable lv-number of type integer(I) with a value 193(literal).

All elements (keywords, variables, data types and literals) are highlighted in the above diagram of the ABAP statement.

Let us take an example of ABAP statement coded in the processing block of program.

PROGRAM Z_FIRST_PROG.WRITE 'Hello World, KIIT'.

In the above example, PROGRAM and WRITE are the keywords. The program displays the list on the screen. In this case, the list consists of ‘Hello World, Welcome to KIIT’.

The above diagram describes the structure of the program and all elements (keywords and literals) are highlighted.

What are the types of Statements?

The first element of an ABAP statement is the ABAP keyword. An ABAP keyword determines the category of the statements. The different statement categories are -

- Declarative Statements
- Modularization Statements
- Control Statements
- Call Statements
- Operational Statements
- Database Statements

Declarative Statements -

Declarative statements used to declare data objects that are used by other ABAP statements in the program or routine. Declarative keywords to form declarative statements.

Some of the declarative statements are –

```
TYPES: BEGIN OF lv_table,
           ……END OF lv_table.
DATA lv_variable TYPE lv_table.
TABLES table_wa.
```

Modularization Statements -

Modularization statements used to define the processing blocks in an ABAP program. There are two types of statements in modularization keywords and those are –

Event Statements –

Used to define event blocks. There are no special end statements for event statements and ends automatically when next processing block started.

Some of the Modularization statements are -

```
AT SELECTION-SCREEN ON <parameter_name>
START-OF-SELECTION.
<statements-block>.
AT USER-COMMAND.
<statements-block>.
```

Defining statements –

Used to define subroutines, function modules, methods, and dialog modules. The END statements are the end statement for defining statements.

Some of the Defining statements are -

```
FORM <subroutine_name>.
          <statements-block>ENDFORM.
FUNCTION <function_name>.
          <statements-block>ENDFUNCTION.
MODULE <module_name> {OUTPUT|[INPUT]}.
 ...ENDMODULE.
```

Control Statements -

Control statements used to control the flow of an ABAP program within a processing block based on the specified conditions.

Some of the Control statements are -

```
IF <condition>.
  <statements-block>.   ENDIF.
WHILE <logical-expression>
          <statements-block>.  ENDWHILE.
CASE <variable>. WHEN <value1>.
  <statements-block>.WHEN <value2>.
  <statements-block>....... ...... ......  WHEN <valuen>.
  <statements-block>. WHEN OTHERS.
  <statements-block>.  ENDCASE.
```

Call Statements -

Call statements used to call processing blocks that are already defined using modularization statements. The calling blocks can either be in the same ABAP program or in a different program.

Some of the Call statements are -

```
PERFORM <subroutine-name>
  USING <variables>.
CALL <function-name>/<method-name>.
SET USER-COMMAND <function-code>.
SUBMIT <report_name>
LEAVE PROGRAM.
```

Operational Statements -

Operational statements process(displays, moves etc,) the data.

Some of the Operational statements are -

```
WRITE <data-object>/<field-symbol>/<formal-parameter>/<text-symbol>
MOVE <field1> TO <field2>.
ADD <field1> TO <field2>.
```

Database Statements -

Database statements use the database interface to access the tables from the central database system. There are two kinds of database statement in ABAP - Open SQL and Native SQL.

Some of the Database statements are -

```
SELECT INSERT DELETE
```

SAP ABAP Keywords -

What is a keyword?

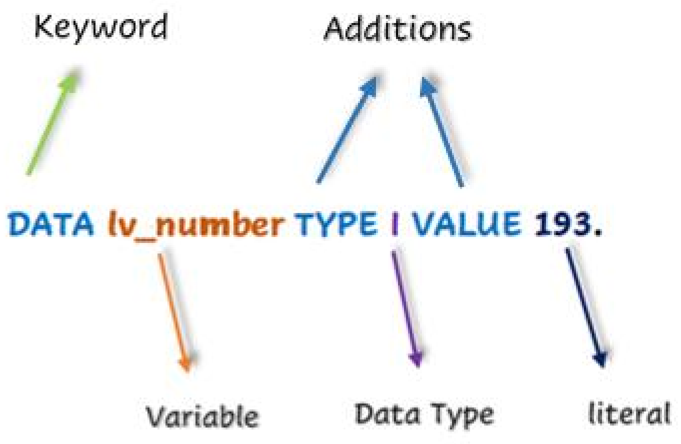
The first element of the ABAP statement is called as ABAP keyword. A keyword determines the ABAP statement category.

ABAP keywords are the vocabulary of the ABAP language. ABAP keywords are the elements of an ABAP statement. ABAP statements are composed of keywords, operands, and operators according to the predefined syntax rules.

ABAP keywords can also contain hyphens (-) to form multi-word expressions. For example, ADD-CORRESPONDING, END-OF-FILE, END-OF-PAGE etc,.

ABAP keywords are not reserved names. Although the use of an ABAP word for naming conventions is not restricted, however it should be avoided if possible.

Let us take an example of ABAP statement coded in the declaration part of program with a keyword.



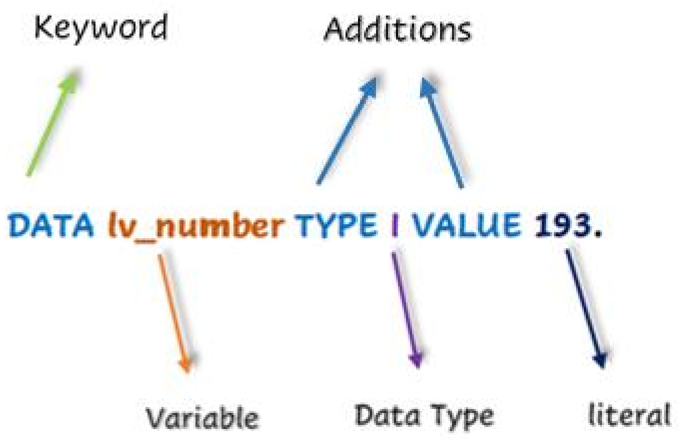
The above ABAP statement used to declare variable lv-number of type integer(I) with a value 193(literal). ABAP statement contains a keyword DATA and additions TYPE, VALUE. All elements (keywords, variables, data types and literals) are highlighted in the above diagram of the ABAP statement.

Let us take an example of ABAP statement coded in the processing block of program.

PROGRAM Z_FIRST_PROG.WRITE 'Hello World, Welcome to KIIT'.

In the above example, PROGRAM and WRITE are the keywords.

The program displays the list on the screen. In this case, the list consists of 'Hello World, Welcome to KIIT'.



The above diagram describes the structure of the program and all elements (keywords and literals) are highlighted.

What are types of keywords?

Below are the list of different keyword categories –

- Declarative Keywords
- Modularization keywords

- Control Keywords
- Call Keywords
- Operation Keywords
- Database statements

Declarative Keywords –

Declarative statements use the declarative keywords to declare the data objects that are used by another ABAP statements.

Some of declarative keywords are -

TYPES DATA TABLES

Modularization Keywords –

Modularization statements uses modularization keyword to define the processing blocks in ABAP program. The modularization keywords are two types –

Event Keywords –

Event statements uses event keywords to define event blocks. There are no special end statements and automatically ends when another block starts.

Some of event keywords are -

AT SELECTION SCREENSTART-OF-SELECTIONAT USER-COMMAND

Defining Keywords –

Defining statements uses defining keywords to define subroutines, dialog modules, functional modules, and methods. Defining statements ends with END keywords.

Some of definitive keywords are -

FORM ..... ENDFORM.FUNCTION ... ENDFUNCTION.MODULE ... ENDMODULE.

Control keywords -

Control statements uses control keywords to controls the flow of an ABAP program within a processing block based on the specified conditions.

Some of control keywords are -

IF WHILE CASE

Call Keywords -

Call statements uses call keywords to call processing blocks. The processing blocks can either be in the same program or in a different program.

Some of call keywords are -

PERFORM CALL SET USER-COMMAND SUBMIT LEAVE TO

Operational keywords -

Operational statements uses the operational keywords to process the data.

Some of operational keywords are -

WRITE MOVE ADD

Database Keywords -

Database statements uses the database keywords to access the tables in the central database system. There are two kinds of database statement in ABAP - Open SQL and Native SQL. Both types of statements use the below DML statements to access the database.

Some of database keywords are -

SELECT INSERT UPDATE DELETE

SAP ABAP Syntax -

ABAP program is set of processing blocks. Each processing block contains one or more statements. The Syntax of ABAP programming language consists of the following –

- Statements
- Formatting ABAP statements (Indentation)
- Chained statement or colon notation
- Comments

ABAP Statements –

ABAP program contains individual ABAP statements. Each statement begins with a keyword and ends with a period(.).

Formatting ABAP statements (Indentation) –

ABAP statement coding has no restriction. We can enter ABAP statement in any format. i.e. Multiple ABAP statements can be written on single line. Single ABAP statement can be split into multiple lines.

Original Program fragment -

```
PROGRAM Z_FIRST_PROG.WRITE 'Hello World, Welcome to KIIT'.
```

Could also written as –

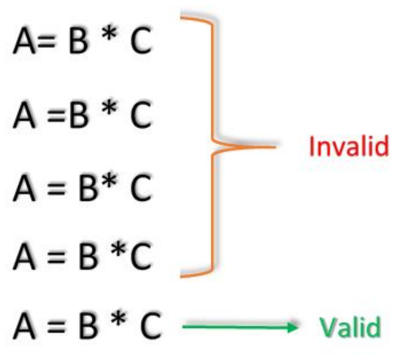
```
PROGRAM Z_FIRST_PROG. WRITE 'Hello World, Welcome to KIIT'.
```

Or as follows –

```
PROGRAM
          Z_FIRST_PROG.
                    WRITE
                              'Hello World, Welcome to KIIT'.
```

The elements of a statement should be separated by at least one blank/space or a line break. Otherwise, blanks and line breaks between tokens are not significant.

Below examples shows the valid and invalid statements -



Colon notation or Chain statement –

ABAP programming allows to concatenate consecutive statements with same keyword in the first part into a chain statement.

If any two or more statements started with same keywords coded consecutively together or one after the other, ABAP has a feasibility to combine them in a single statement. To combine two or more consecutive statements, colon (:) should be used after the keyword.

To concatenate sequence of statements, write the identical keyword only once and place a colon(:) immediately after keyword. After colon(:), write the remaining parts of the statements one after the other with a comma (,) separated. After all written, place a period (.) at the end of the statement.

Syntax -

```
ABAP_keyword : statements-comma-seperated.
```

For example, the declaration statement sequence shown below -

```
DATA v_a TYPE I.DATA v_b TYPE P.
```

Chained statement -

```
DATA: v_a TYPE I, v_b TYPE P.
```

In the chain statement, after the colon the remaining parts of the statements are comma separated. We can write the statement like below -

```
DATA: v_a TYPE I,
v_b TYPE P.
```

In a chain statement, the first part (before the colon) is not limited to the keyword of the statements. The identical part of all the statements can code before the colon(:).

For example, the processing statement sequence shown below –

```
SUM = SUM + 1. SUM = SUM + 2. SUM = SUM + 3.
```

Chain statement for the above –

```
SUM = SUM + : 1, 2, 3.
```

In the above example, "SUM = SUM + " is the identical part of all the ABAP statements. So to make the chain statement for the above, write the common part first, next code colon(:) and then remaining part of statements comma separated.

Example -

Problem should be described here

Code -

```
*&---------------------------------------------------------------------**& Report  Z_COLON_NOTATION*&----------------------------------------------------------------------**& Written by KIIT*&----------------------------------------------------------------------*
```

REPORT Z_COLON_NOTATION.

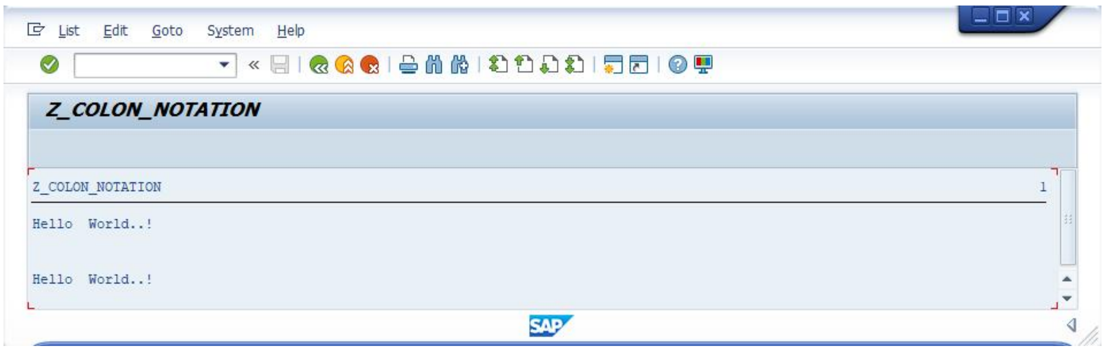
* Displaying 'Hello ' on the output. WRITE 'Hello '.

* Displaying 'World..!' on the same row of the output.WRITE 'World..!'.

* Skipping next two lines on the output.SKIP 2.

* Chained statement WRITE : 'Hello ', 'World..!'.

Output -



```
Z_COLON_NOTATION                                                    1

Hello  World..!


Hello  World..!
```

Explaining Example -

In the above example, each and every statement is preceeded with a comment to explain about the statement. Go through them to get clear understanding of example code.

WRITE 'Hello '.
WRITE 'World..!'. - write output on the line 3 and 4. **SKIP 2** - Inserts 2 empty line and places the cursor on 6th line. **WRITE : 'Hello ', 'World..!'** - Writes the output on the line 6. First two WRITE statements equal to third WRITE statement.

Comments –

Comments or Non-executable statements definition are self-explanatory. Comments generally used to understand the code easily for the programmers. Comments are do nothing statements and are ignored by the computer.

There are two types of comments that are supported in ABAP editor and those are -

- Full line comments or Comment Lines
- Partial comments or End of line comments

Full line comments or Comment Lines -

Full line comment contains only a comment for the full line and nothing else.

Full line comments can be made by using asterisk (*) or quote(") as a first character of the entire line. If the line starts with * or ", then those lines are considered as comments and ignored by the compiler while generating program.

For **example**, let us write a full line comment below –

```
* Full line comment in SAP.
```

Partial comments or End of line comments –

If user creates a comment at the end of the executable statement (after a period), those called as a partial comment. Partial comment should get started with double quote (“). If any information starts with “, then those are ignored by the compiler while generating program. Otherwise, compilation errors occurred.

For **example**, let us write a partial comment below –

```
Write 'Hello World..!'.  "Starting of a Partial comment
```

Note! ABAP Editor not capitalize the comment either it may be a full or partial comment code.

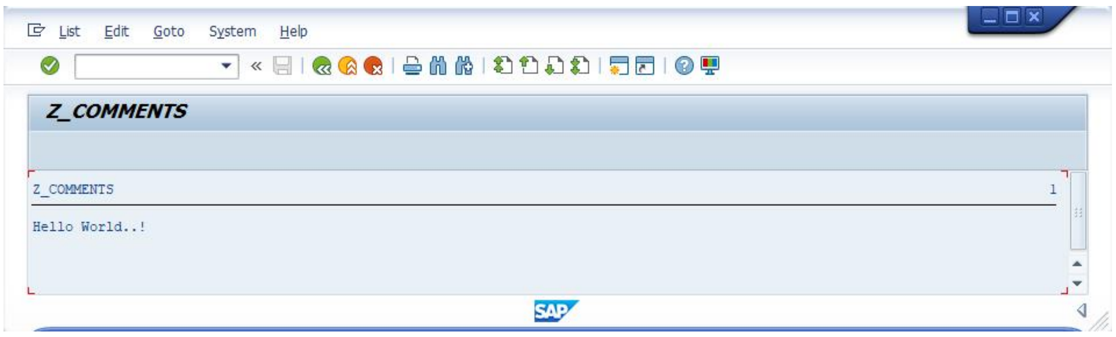
Example -

Write a program to show case full and partial comments.

Code -

```
*&---------------------------------------------------------------------**& Report  Z_COMMENTS*&---------------------------------------------------------------------**& Written by KIIT*&---------------------------------------------------------------------*
REPORT  Z_COMMENTS.
* Full line commentWRITE 'Hello World..!'. "partial comment
```

Output -



Z_COMMENTS

```
Z_COMMENTS                                                        1
Hello World..!
```

Explaining Example -

In the above example, each and every statement is preceeded with a comment to explain about the statement. Go through them to get clear understanding of example code.

*** Full line comment -** Full line comment in the program. **" partial comment -** Partial line comment in the program.