How to create a function module in SAP ABAP: concepts, Function Builder, and a working SPELL_AMOUNT example

This guide covers the basics you need before you create a function module in SAP ABAP: what function modules are, what they contain, and how to open the Function Builder. You then build a complete report that prompts for a number and calls the predefined SPELL_AMOUNT function module to convert it into words. The example shows how to set up the function call, handle the return code, and run the finished report.

Prerequisites

- Access to the SAP Easy Access screen.
- The ability to open transactions SE37 (Function Builder) and SE38 (ABAP Editor).

Step 1: Understand what function modules are

Before you create a function module in SAP ABAP, it helps to know how function modules work:

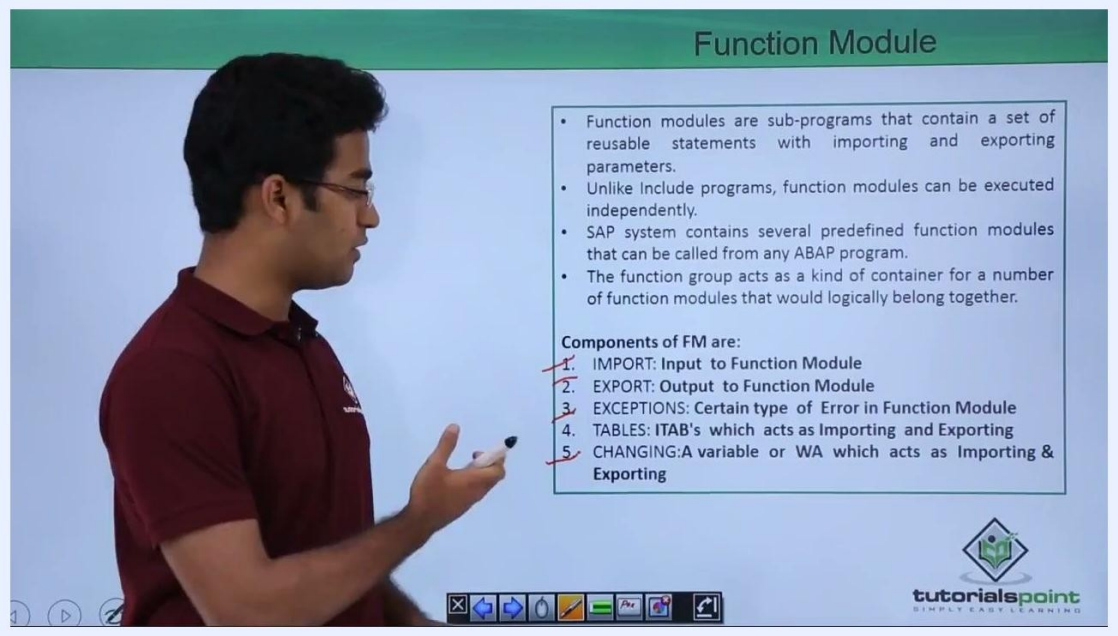
- Function modules are sub-programs that contain a set of reusable statements with importing and exporting parameters.
- Unlike include programs, function modules can be executed independently.
- SAP provides several predefined function modules that you can use in different programs.
- You can also create custom function modules.
- Function modules are grouped into function groups, which act as containers for logically related modules.

Step 2: Learn the components of a function module

Every function module is built from the following components:

- **IMPORT**: Input to the function module.
- **EXPORT**: Output from the function module.
- **EXCEPTIONS**: Handle specific types of errors in the function module.
- **TABLES**: Internal tables (ITABs) that act as importing and exporting parameters.

- **CHANGING**: A variable or work area (WA) that acts as both an importing and an exporting parameter.



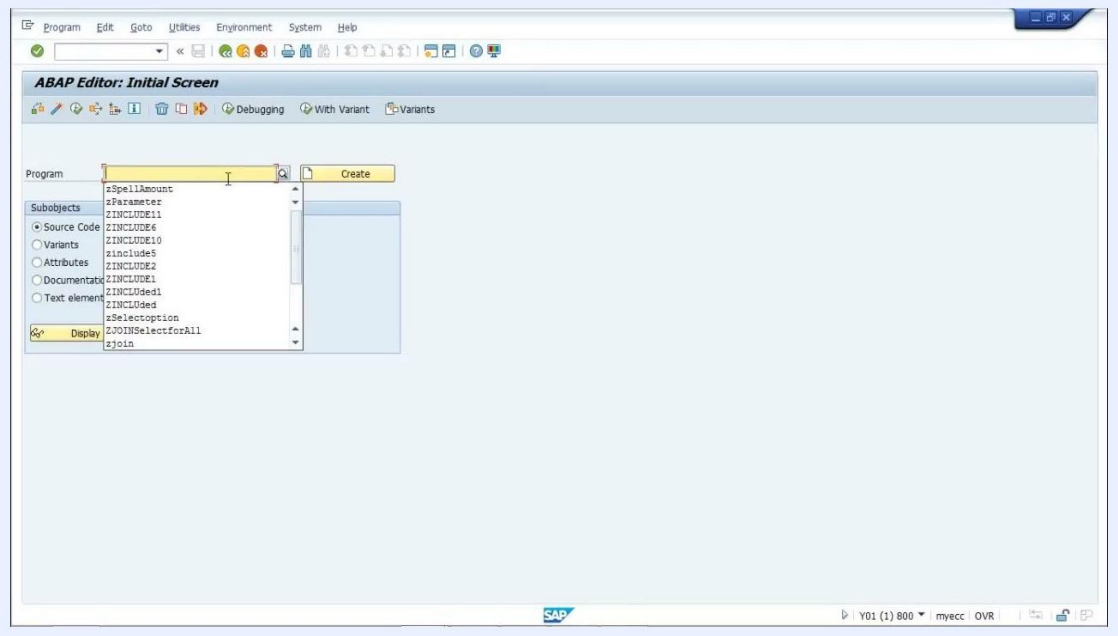
Slide titled "Function Module" listing the characteristics of function modules and a numbered list of their components

Step 3: Open the Function Builder to start working with function modules

The Function Builder is where you view and create function modules in SAP.

1. Go to the SAP Easy Access screen.

2. Enter SE37 in the command field to open the Function Builder.



SAP Easy Access screen with se37 entered in the command field to open the Function Builder

Step 4: Create the program for the function module example

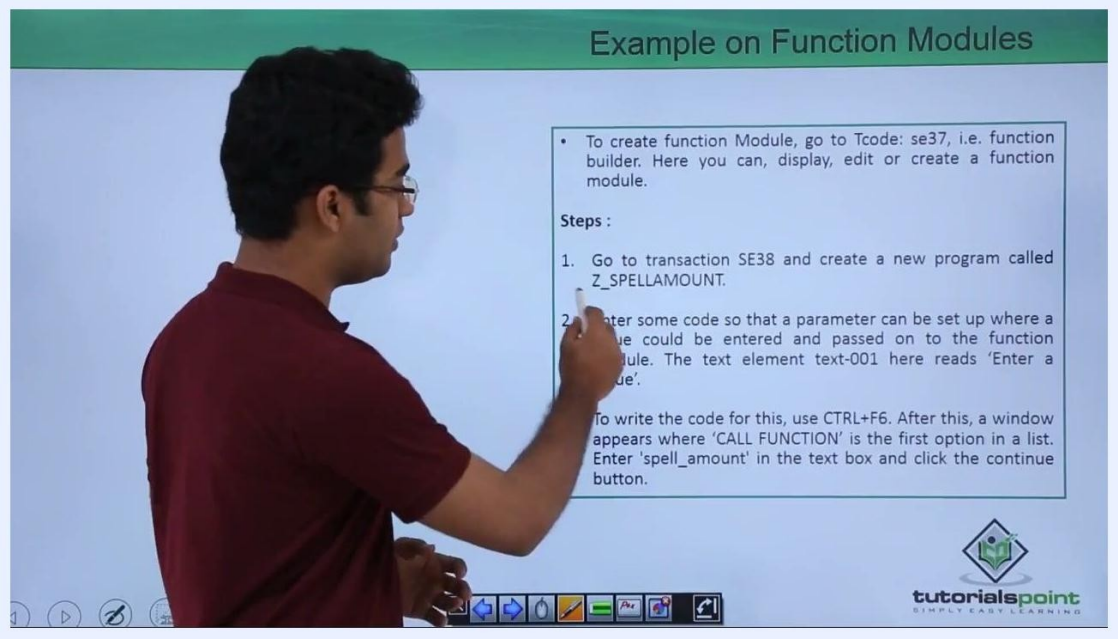
In this example, the report prompts the user for a numeric amount. When the user runs the report, the SPELL_AMOUNT function module converts the entered number into words.

1. Go to transaction SE38 to open the ABAP Editor.
2. Create a new program named Z_SPELLAMOUNT.

The "Example on Function Modules" slide summarizes the overall approach:

- Create the program in SE38.
- Enter code that sets up a parameter, so a value can be entered and passed to the function module. The text element text-001 reads "Enter a Value".

- To write the function call, use CTRL+F6. In the window that appears, CALL FUNCTION is the first option in the list. Enter spell_amount in the text box and click the continue button.



Slide titled "Example on Function Modules" with the steps for creating and coding the Z_SPELLAMOUNT program in SE38

Step 5: Switch to change mode

If the report opens in display mode, switch to change mode so you can edit it. Confirm that the title bar reads ABAP Editor: Change Report ZSPELLAMOUNT.

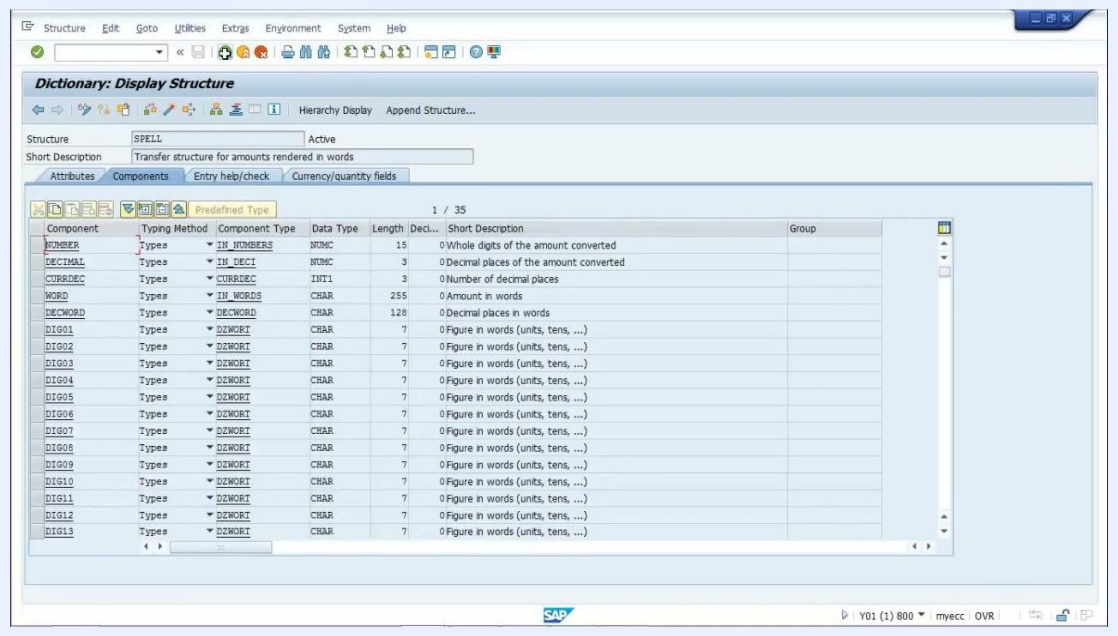
Step 6: Define a result structure for the output

Declare a variable above the function call to hold the output of the function module.

1. Enter the following line at the top of your report:

```
DATA result LIKE spell.
```

2. Note that this creates a variable result based on the predefined structure spell, which the output requires.



ABAP Editor with the line "Data result like spell." added above the function call

For reference, SPELL is a predefined SAP structure that stores the output of SPELL_AMOUNT. It contains fields such as NUMBER, DECIMAL, CURRDEC, and WORD, which represent the amount in words and related details. You can view its definition in the Data Dictionary.

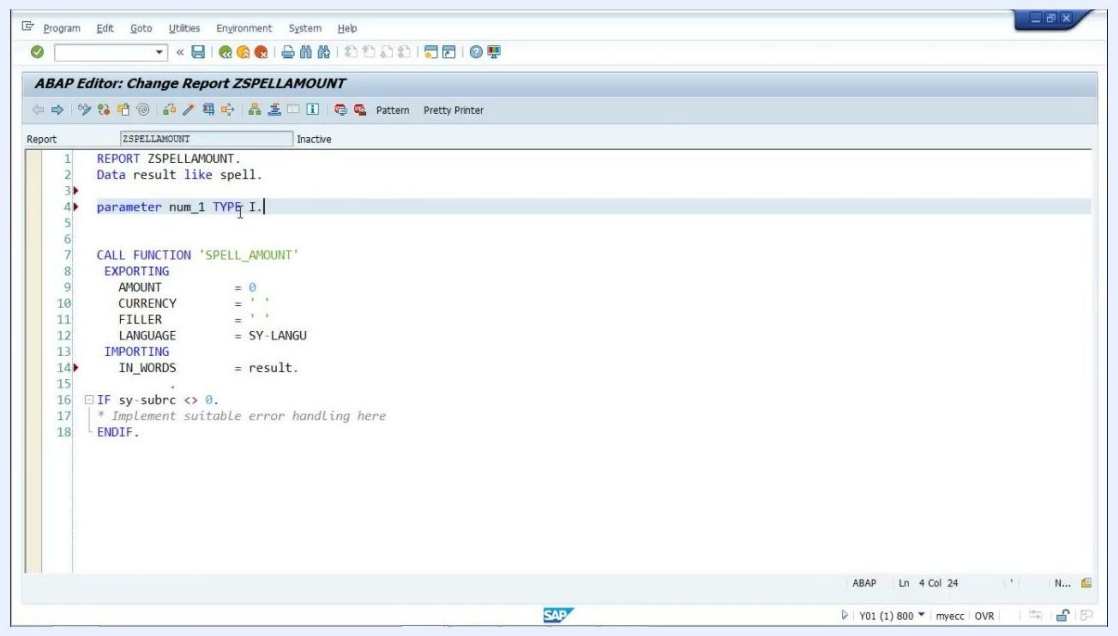
Step 7: Add a parameter for user input

Define a parameter so the user can enter the amount to convert.

1. Add the following line near the top of your report:

```
PARAMETERS: num_1 TYPE I.
```

2. Note that this creates an input field where the user enters an integer value.



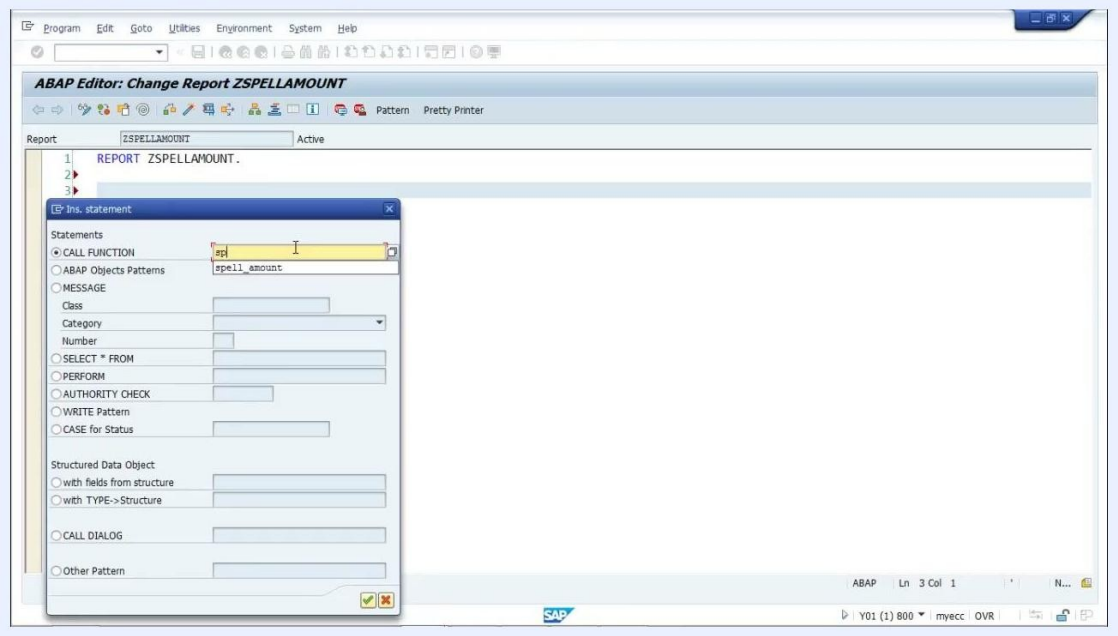
ABAP Editor with the line "parameter num_1 TYPE I." added to provide a user input field

Step 8: Insert the function module call

Use the pattern tool to add the function module call to your function module example.

1. Place the cursor where you want the function call.
2. Click the **Pattern** button in the ABAP Editor toolbar.
3. In the **Ins. statement** dialog, select **CALL FUNCTION**.

4. Begin typing spell_amount in the function name field and select it from the dropdown.



ABAP Editor with the Ins. statement dialog open, CALL FUNCTION selected, and spell_amount being entered

Click the green checkmark (Continue/Enter) to insert the template. The editor adds a code block for CALL FUNCTION 'SPELL_AMOUNT' with exporting, importing, and exception parameters, mostly commented out.

Step 9: Configure the function module call

Edit the inserted template so it passes your input and captures the result.

1. Uncomment the EXPORTING and IMPORTING parameter sections.
2. Remove the EXCEPTIONS section if your program doesn't need it.
3. Set the parameters as follows:

    - AMOUNT = num_1
    - CURRENCY = '' (leave blank if not required)
    - FILLER = '' (leave blank if not required)
    - LANGUAGE = SY-LANGU
    - IN_WORDS = result

The code should look like this:

abap

```
CALL FUNCTION 'SPELL_AMOUNT'
  EXPORTING
    AMOUNT   = num_1
    CURRENCY = ''
    FILLER   = ''
    LANGUAGE = SY-LANGU
  IMPORTING
    IN_WORDS = result.
```

Step 10: Add error handling and output logic

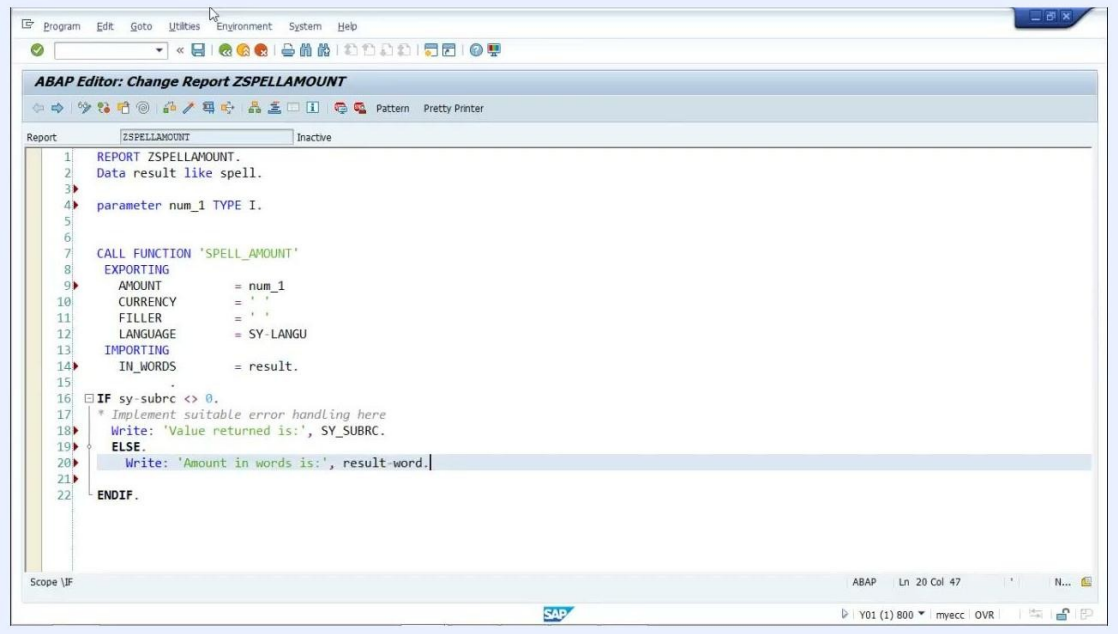
After the function call, check the return code sy-subrc to see whether the function module ran successfully.

- If sy-subrc is not equal to 0, the program outputs the returned value.
- If sy-subrc is 0, the program outputs the converted amount using the result-word field.

Enter the following code after the function call:

abap

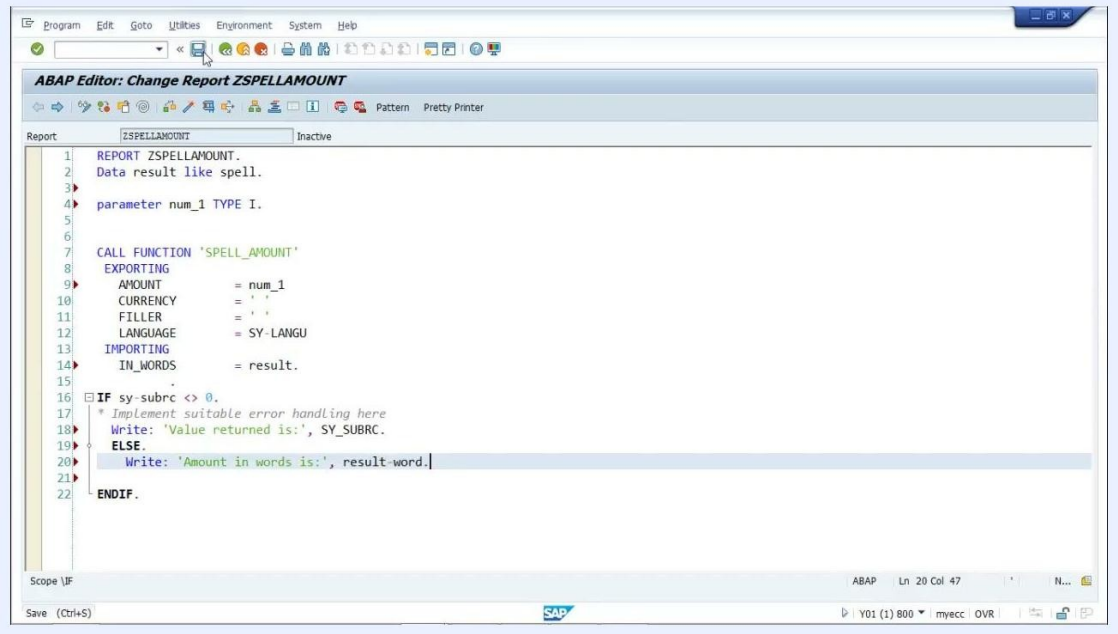
```
IF sy-subrc <> 0.
  WRITE: 'Value returned is:', SY-SUBRC.
ELSE.
  WRITE: 'Amount in words is:', result-word.
ENDIF.
```



ABAP Editor showing the completed IF/ELSE logic with the WRITE statements for the amount in words and the returned value

Step 11: Save the report

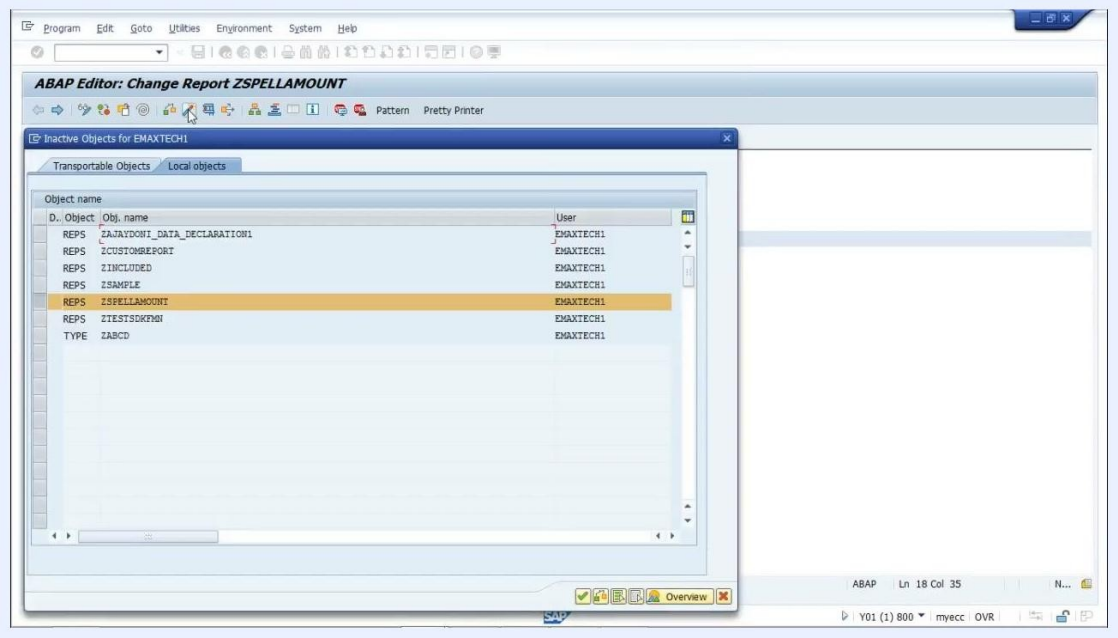
Click the save icon (floppy disk) in the ABAP Editor toolbar, or press Ctrl+S. Make sure there are no syntax errors before you continue.



ABAP Editor with the save icon highlighted and the completed code visible

Step 12: Activate the report

If prompted, activate the report so you can run it. In the **Inactive Objects** dialog, select your report (ZSPELLAMOUNT) and confirm the activation.

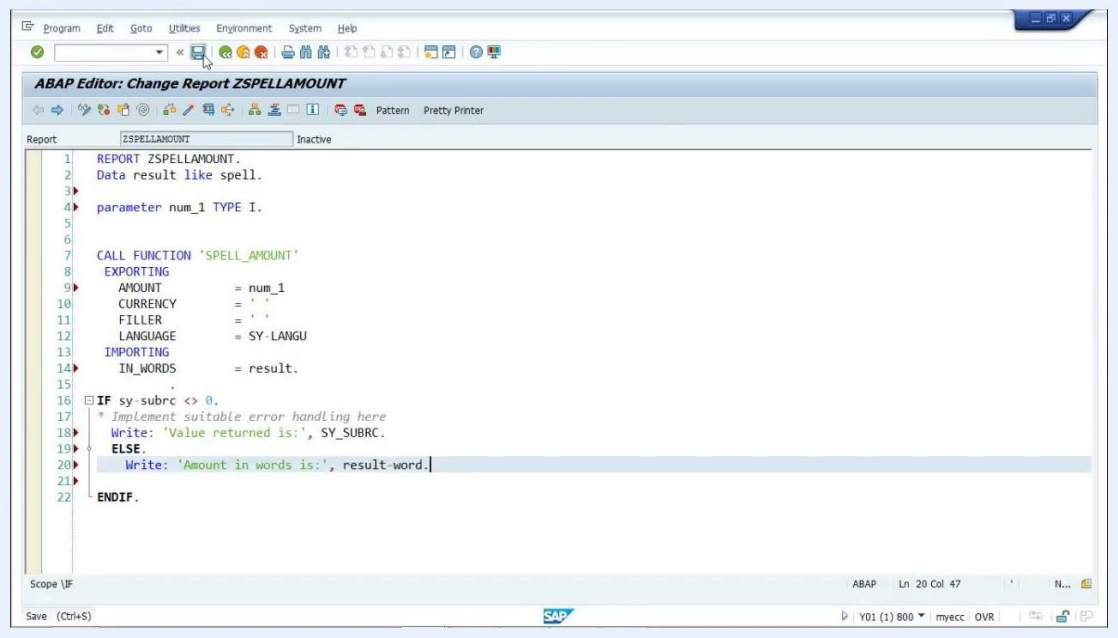


Inactive Objects dialog with ZSPELLAMOUNT selected for activation

Step 13: Execute the report and verify the output

1. Run the report.
2. Enter a numeric value when prompted, for example 2314.
3. Check the output screen. It should display the amount in words:

```
Amount in words is:
TWO THOUSAND THREE HUNDRED FOURTEEN
```



SAP output screen showing "Amount in words is: TWO THOUSAND THREE HUNDRED FOURTEEN"

Summary

By following these steps, you have:

- Reviewed what function modules are and how their components work.
- Opened the Function Builder with SE37.
- Built a report that calls the SPELL_AMOUNT function module with a user-entered value.
- Implemented error handling for the function module call.
- Displayed the result in words when the function executes successfully.
- Saved, activated, and executed the report to verify it works.

Your ABAP report now lets users enter a number and receive its value in words. The example shows how parameters enable user input, how SPELL_AMOUNT does the conversion, and how modularization lets a function module run independently of the report that calls it.