ABAP Cheat Sheet

A concise reference for ABAP (Advanced Business Application Programming) syntax, statements, and concepts, useful for quick lookup and understanding.

ABAP Basics

Data Types

`C`	Character (fixed length)
`N`	Numeric character (fixed length, only digits)
`D`	Date (YYYYMMDD)
`T`	Time (HHMMSS)
`I`	Integer
`F`	Floating point number
`P`	Packed number (for monetary values)
`STRING`	Character string (variable length)
`XSTRING`	Byte string (variable length)

Basic Syntax

`WRITE`: Output statement.

Example:
`WRITE 'Hello, ABAP!'.

`DATA`: Declare variables.

Example:
`DATA: lv_name TYPE string.`

`PARAMETERS`: Input parameters for reports.

Example:
`PARAMETERS: p_carrid TYPE s_carr_id.`

`SELECT`: Read data from database tables.

Example:

```
SELECT * FROM sflight INTO TABLE @lt_flights.
```

`LOOP AT`: Loop through internal tables.

Example:

```
LOOP AT lt_flights INTO ls_flight.
  WRITE: / ls_flight-carrid, ls_flight-connid.
ENDLOOP.
```

`IF...ELSEIF...ELSE...ENDIF`: Conditional logic.

Example:

```
IF sy-subrc = 0.
  WRITE: / 'Success'.
ELSE.
  WRITE: / 'Failure'.
ENDIF.
```

Internal Tables

Table Types

Standard Table	Unsorted table with linear index. Fast for sequential access. Default table type.
Sorted Table	Table sorted by key. Faster access by key. Requires `SORT` statement for initial fill.
Hashed Table	Table with hash algorithm for key access. Fastest access by key. Cannot be accessed by index.

Table Operations

`APPEND` : Add a row to an internal table.

Example:
`APPEND ls_flight TO lt_flights.`

`INSERT` : Insert a row into an internal table.

Example:
`INSERT ls_flight INTO TABLE lt_flights INDEX 1.`

`MODIFY` : Modify a row in an internal table.

Example:
`MODIFY TABLE lt_flights FROM ls_flight TRANSPORTING carrid connid WHERE carrid = 'AA'.`

`DELETE` : Delete a row from an internal table.

Example:
`DELETE TABLE lt_flights WHERE carrid = 'AA'.`

`READ TABLE` : Read a row from an internal table.

Example:
`READ TABLE lt_flights WITH KEY carrid = 'AA' connid = '17' INTO ls_flight.`

`SORT` : Sort an internal table.

Example:
`SORT lt_flights BY carrid connid.`

`CLEAR` : Clear an internal table (header line, if applicable).

Example:
`CLEAR lt_flights.`

`REFRESH` : Delete all rows from an internal table, but keep header line.

Example:
`REFRESH lt_flights.`

ABAP Objects

Class Definition

Start with `CLASS <class_name> DEFINITION.` and end with `ENDCLASS.`.

Example:

```
CLASS lcl_my_class DEFINITION.
  PUBLIC SECTION.
    METHODS: constructor,
             display.
  PRIVATE SECTION.
    DATA: mv_name TYPE string.
ENDCLASS.
```

Visibility sections: `PUBLIC`, `PROTECTED`, `PRIVATE`.

Methods are defined using the `METHODS` statement.

Attributes (variables) are defined using the `DATA` statement.

Class Implementation

Start with `CLASS <class_name> IMPLEMENTATION.` and end with `ENDCLASS.`.

Example:

```
CLASS lcl_my_class IMPLEMENTATION.
  METHOD constructor.
    mv_name = 'Initial Name'.
  ENDMETHOD.

  METHOD display.
    WRITE: / mv_name.
  ENDMETHOD.
ENDCLASS.
```

Method implementations are defined within the class implementation.

Constructor: Special method called when an object is created.

Object Creation

Create an object using `CREATE OBJECT`.

Example:

```
DATA: lo_my_object TYPE REF TO lcl_my_class.
CREATE OBJECT lo_my_object.
```

Call methods using `lo_my_object->method_name( ).`

Example:

```
lo_my_object->display( ).
```

Release object using `FREE OBJECT lo_my_object.`

ABAP Dictionary

Data Dictionary Objects

Tables	Define database tables, their structure and relationships.
Data Elements	Define elementary data types, field labels, and documentation.
Domains	Define technical attributes of data types (e.g., length, data type).
Structures	Combine multiple fields into a single unit (like a record).
Views	Virtual tables that combine data from one or more tables.
Search Helps	Provide value help (F4 help) for input fields.

Table Creation

Use transaction `SE11` to create and maintain dictionary objects.
Define fields, data types, and key fields for a table.
Specify technical settings (e.g., data class, size category).
Activate the table to make it available in the system.

Data Element and Domain

Data elements reference a domain, providing semantic information.
Domains define data type, length, and value range.
Use transaction `SE11` to create data elements and domains.