Sets:

Sets are composite data types that store a collection of unique, unordered elements. Each element in a set is distinct, and the order of elements is not important. Sets are useful for tasks that involve testing membership or performing set operations like union, intersection, and difference. In programming languages, sets are often implemented using data structures like arrays, linked lists, or hash tables.

The type definition has this structure:

TYPE <set-identifier> = SET OF <Basetype>

The variable definition for a set includes the elements of the set.

DEFINE <identifier> (value1, value2, value3, ... ) : <set-identifier>

A set of vowels could be declared as follows:

TYPE Sletter = SET OF CHAR

DEFINE vowel ('a', 'e', 'i', 'o', 'u') : letters

Example

TYPE FruitsSet = SET OF STRING

// Declaring a set of fruits

DEFINE fruits: FruitsSet ("apple", "banana", "orange")

// Checking membership

IF "apple" IS IN fruits THEN

OUTPUT "Apple is in the set"

ENDIF

// Performing set operations

DECLARE all\_fruits: FruitsSet

ADD "apple" TO all\_fruits

ADD "banana" TO all\_fruits

ADD "orange" TO all\_fruits

ADD "grape" TO all\_fruits

ADD "melon" TO all\_fruits

DECLARE common\_fruits: FruitsSet

COMMON ELEMENTS OF fruits AND all\_fruits INTO common\_fruits

// Displaying the common fruits

FOR EACH fruit IN common\_fruits DO

OUTPUT fruit

ENDFOR

python

# Creating a set

fruits = {"apple", "banana", "orange"}

# Adding elements to a set

fruits.add("grape")

# Checking membership

if "apple" in fruits:

print("Apple is in the set")

# Performing set operations

all\_fruits = {"apple", "banana", "orange", "grape", "melon"}

common\_fruits = fruits.intersection(all\_fruits)

Records:

* Records, also known as structs or tuples in some languages, are composite data types that store a collection of related values with different data types.
* Each value in a record is typically referred to as a field or attribute.
* Records provide a way to group related pieces of data together, making it easier to organize and manipulate complex data structures.
* The fields in a record can have different types and can be accessed individually.

*Classes*

* A class is a composite data type that includes variables of given data types and
* methods (code routines that can be run by an object in that class). An object is
* defined from a given class; several objects can be defined from the same class.
* Classes and objects will be considered in more depth in Chapter 20.