PRACTICAL NO. 2

Title - Study of basic syntaxes in R languaage

Introduction:

R is a programming language widely used for statistical computing, data analysis, and data visualization. In this practical, we will explore basic syntaxes in R, which will form the foundation for performing more advanced data analysis tasks. R Data types are used to specify the kind of data that can be stored in a variable.

Objective:

The objective of this practical is to familiarize yourself with the basic syntax and constructs of the R programming language. This includes understanding how to define variables, perform operations, and use basic functions.

Tools Required:

- R programming environment (R console or RStudio or VS code with R Extension)
- Basic knowledge of mathematics and programming concepts.

Theory:

Data Types in R language:

R Data types are used to specify the kind of data that can be stored in a variable. For effective memory consumption and precise computation, the right data type must be selected. R supports several fundamental data types, each designed for handling different kinds of data. These data types are essential for performing various operations in R. Each R data type has its own set of regulations and restrictions.

Data Type	Syntax Example	Example Value
Numeric	`num <- 10.5`	`10.5`
Integer	`int <- 10L`	`10L`
Character	`text <- "Hello, World!"`	`"Hello, World!"`
Logical	`flag <- TRUE`	`TRUE`
Complex	`complex_num <- 3 + 4i`	`3 + 4i`
Raw	`raw_data <- charToRaw("Hello")`	Raw byte representation

Numeric

- **Description**: Represents real numbers, including both integers and decimal numbers.
- **Syntax**: Simply assign a numeric value to a variable.
- **Example: num <- 10.5 # Numeric value**

Integer

- **Description**: Represents whole numbers. You define an integer by appending `L` to the number.
- **Syntax**: Append `L` to the numeric value to create an integer.
- **Example: int <- 10L # Integer value**

Character

- **Description**: Represents text or string data. Strings are enclosed in double quotes (`" "`) or single quotes (`' '`).
- **Syntax**: Use either double or single quotes to assign a string value.
- **Example: text <- "Hello, World!" # Character (string) value**

Logical

- **Description**: Represents boolean values: `TRUE` or `FALSE`.
- **Syntax**: Use the `TRUE` or `FALSE` keyword.
- **Example: flag <- TRUE # Logical value**

Complex

- **Description**: Represents complex numbers with a real and imaginary part.
- **Syntax**: Complex numbers are created using `i` to denote the imaginary part.
- **Example: complex_num <- 3 + 4i # Complex number (real + imaginary part)**

Raw

- **Description**: Represents raw bytes, typically used for lower-level operations.
- **Syntax**: Create raw vectors using the `charToRaw()` function or `raw()` function.
- **Example: raw_data <- charToRaw("Hello") # Raw data representation of a string**

These are the fundamental data types in R, each suited to specific types of operations. Understanding how to use them is key to performing data analysis in R.

List in R language:

In R, a **list** is a data structure that can hold a collection of different types of objects such as numbers, strings, vectors, other lists, and more. Unlike vectors, which can only hold elements of the same type, lists are more flexible and can store elements of different types.

Key Features of a List:

- A list can hold elements of different types (e.g., numeric, character, logical, etc.).
- Elements of a list can be accessed using the `[[ ]]` or `[ ]` notation.
- Lists can be named, which helps in identifying elements within the list.

Syntax for Creating a List

You create a list using the `list()` function in R.

Basic Syntax: list_name <- list(element1, element2, element3, ...)

Example of a Simple List:

```
# Creating a list with different data types

my_list <- list(42, "Hello", TRUE, 3.14)

# Print the list

print(my_list)
```

Named List

You can also create a list with named elements, which makes accessing and referencing the elements easier.

Syntax: named_list <- list(name = "John", age = 25, married = TRUE)

Example:

```
# Creating a named list

person <- list(name = "John", age = 25, married = TRUE)

# Print the list

print(person)
```

Array in R language:

An **array** in R language is a multi-dimensional data structure used to store data in more than one dimension (i.e., rows, columns, and beyond). It is similar to a matrix but can have more than two dimensions. Arrays are particularly useful when you need to represent data that involves more than two dimensions, such as multi-dimensional matrices.

Key Features of an Array:

- Arrays can have one or more dimensions (1D, 2D, 3D, etc.).
- All elements in an array must be of the same data type (numeric, character, etc.).
- Arrays are created using the `array()` function.

Syntax for Creating an Array

You create an array using the `array()` function in R.

Basic Syntax: array(data, dim = c(dim1, dim2, dim3, ...))

Example of Creating a 1D Array (Vector)

A 1-dimensional array is essentially just a vector in R.

```
# Create a 1D array (vector)

arr1D <- array(1:5)  # Numbers 1 to 5

print(arr1D)
```

Example of Creating a 2D Array (Matrix)

A 2-dimensional array can be seen as a matrix (rows and columns).

```
# Create a 2D array (matrix)

arr2D <- array(1:6, dim = c(2, 3))  # 2 rows and 3 columns

print(arr2D)
```

Example of Creating a 3D Array

A 3-dimensional array adds depth to the data structure (i.e., it has multiple "slices" or layers).

```
# Create a 3D array (3 layers, 2 rows, 2 columns)

arr3D <- array(1:12, dim = c(2, 2, 3))  # 2 rows, 2 columns, 3 layers

print(arr3D)
```

Matrices in R language:

Matrices in R Language

In R, a **matrix** is a two-dimensional data structure, similar to a table with rows and columns, where all the elements must be of the same data type (numeric, character, etc.). Matrices are useful for mathematical operations, especially linear algebra tasks.

Key Features of a Matrix:

- A matrix is a two-dimensional array.
- All elements in a matrix must be of the same data type.
- A matrix is created using the `matrix()` function.
- Matrices are indexed by row and column numbers.

Syntax for Creating a Matrix in R

You can create a matrix using the `matrix()` function, which takes the data and the dimensions (number of rows and columns) as input.

Basic Syntax: matrix(data, nrow, ncol, byrow = FALSE, dimnames

Example:

Creating a Matrix with Row and Column Names

```
# Creating a matrix with row and column names

m_named <- matrix(1:6, nrow = 3, ncol = 2, byrow = TRUE,

                  dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2")))

print(m_named)
```

Output:

```
     Col1 Col2

Row1    1    2

Row2    3    4

Row3    5    6
```

Conclusion:

In this practical, we learned the basic syntaxes and structures of the R programming language, including various Data Types, List in R, Array in R, Matrices in R language. Understanding these basics will allow you to write simple scripts and perform fundamental data analysis tasks.