ASSIGNMENT-1

1) List and Explain R objects with examples.

* R objects:-

R consists of a number of data objects to perform various functions. There are 6 types of objects in R programming. They include: 1) Vector
2) List
3) Matrix
4) Array
5) factor
6) Data Frame

(1) vector:-

→ Vector is a basic data structure in R. It contains elements of the same type, the data type can be logic, integer, double, character, complex and raw.

→ A vector type can be checked with the typeOf() function

→ The no. of elements in the vector can be checked with the function length().

• Creation of vector:-

→ Vectors are generally created using the c() function.

→ A vector must have elements of the same type this function will type and coerce elements to the same type, if they are different coercion is from higher to lower lower to higher types from "logical to integer to double to character".

Ex:-

```
x <- c(1,5,4,9,0)
>typeof(x)
[1] double
>length(x)
[1] 5
```

```
> x <- c(1.5, 5, TRUE, "hello")
>typeof(x)
character.
```

• Accessing elements of a vector:-

→ Elements of a vector can be accessed using vector indexing. The vector used for indexing can be logical, integer or character vector

→ Vector index in a 'R' starts from 1, unlike programming language where index starts from 0.

→ We can use a vector of integers as to access specific elements.

Ex:-

```
x ← c (0,2,4,6,8,10)
> x [3]
[1] 4
> x [c(2,4)]
2 6
```

```
> x [-1]
2 4 6 8 10
```

• Modification of vector elements in R:-

→ We can modify a vector using the assignment operator. If we want to truncate the elements we can use reassignment.

Ex:-

```
x ← c (-3, -2, -1, 0, 1, 2)
> x
[1] -3 -2 -1 0 1 2
x [x < 0] ← 5
[1] 5 0 5 0 1 2
```

• How to delete a vector:-

→ We can delete a vector by simply assigning a "NULL" to it

Ex:-

```
> x ← NULL
> x
[1] NULL
```

(2) Lists:-

→ List is a data structure having components of its data types of a vector having all elements are a same type atomic vector but a vector having elements of different type called "Lists".

Creation of a list:-

→ List can be created using the list() function.

Ex:- x ← list("a"= 2.5, "b"= TRUE, "c"=1:3)

→ Its structure can be examined with str() function.

Ex:- > str(x)

```
[1] List of 3
 $a: num 2.5
 $b: logi TRUE
 $c: int [1:3] 1 2 3
```

• Accessing components of a list:-

Lists can be accessed in similar fashion to vectors. Integers, logical or character vector can be used for Indexing.

Ex:- > x

```
$name
[1] "John"
$age
[1] 19
$speaks
[1] "English" "French"
```

```
>x[c(1:2)]
$name
[1] "John"
```

```
> x ["age"]
$age
[1] 19
```

• Modify list:-

We can change a components of list by using assignment. Adding new components is easy as simply assign values using new tags and it will pop into action

```
> x [["married"]] ← FALSE
$married
[1] FALSE
```

• Deletion of components in a list:-

We can delete a component by simply assigning a "NULL" to it

>x[["age"]] <- NULL.

(3) Matrix:-

→ Matrix is a 2D-dimensional Datastructure in R programming. Matrix is similar to vector but additionally contains the dimension attribute.

→ All attributes of an object can be checked with the attributes() function. Dimensions can be checked directly with the "dim()". we can check if a variable is a matrix or not with the class() function.

• Creation of a matrix:-

→ A Matrix can be created using Matrix() function. Dimensions of the matrix can be defined by passing appropriate value for arguments N rows and N columns.

Ex:- matrix(1:9, nrow=3, ncol=3)

```
[1]  1  4  7
     2  5  8
     3  6  9.
```

• How to access elements of matrix?

→ We can access elements of a matrix using the square brackets [indexing method]. Elements can be accessed as var[row, col]

Ex:-

```
> x      [,1]  [,2]  [,3]
[1,]      1     4     7
[2,]      2     5     8
[3,]      3     6     9
```

```
> x[c(1,2), c(2,3)]
     [,1]  [,2]
[1,]   4     7
[2,]   5     8
```

• How to modify a matrix in R:-

→ We can combine assignment operator with the for accessing elements of matrix to modify it.

Ex: x[2,2] <- 10

> x

-Output:

	[,1]	[,2]	[,3]
[1,]	1	4	7
[2,]	2	10	8
[3,]	3	6	9

4) Array:-

→ Arrays are the 'R' data objects which can store data in more than 2-dimensional. Arrays can store only one data type.

→ Array is created using array() function it takes vectors as input and uses the values in the dim() parameters to create an array.

Ex:-

```
V1 <- c(5,9,3)
V2 <- c(10,11,12,13,14,15)
result <- array(c(V1,V2), dim = c(3,3,2))
Print(result)
```

, , 1

	[,1]	[,2]	[,3]
[1,]	5	10	13
[2,]	9	11	14
[3,]	3	12	15

, , 2

	[,1]	[,2]	[,3]
[1,]	5	10	13
[2,]	9	11	14
[3,]	3	12	15

- Accessing array elements

We can access array elements by using index method.

```
Print (result [3, , 2])        # 3rd row, all columns of 2 matrix
3  12  15
```

(5) Factors :-

→ Factor is a data structure used for predefined finite number of values.

→ We know the possible values before hand and these predefined distinct values are called Levels.

Ex:

```
>x
[1] single married married single
Levels: married single.
>class(x)
[1] "factor"
```

- Creation of factor:-

→ We can create a factor using the function factor. Levels of a factor are inferred from the data is not provided.

Ex:-

```
>x ← factor (c("single", "married", "married", "single"))
>x
[1] single married married single
    Levels: married single
```

- Accessing components of a factor :-

```
>x[3]
[1] married
Levels: married single
>x[-1]
[1] single married married single
levels: married single.
```

• How to modify a factor:-

→ components of a factor can be modified using simple assignments.
→ However, we cannot choose values outside of the predefined values.

Ex:-

x[2] ← divorced

```
>x
[1] single divorced married single
levels: married single divorced.
```

(6) Data Frames:-

→ Dataframe is a 2 Dimensional structure in R. It is a special case of a list which has each component of equal length.
→ Each component form the column and contents of the component form the rows.

Ex:-

```
>x
  SN Age Name
1  1  21 John
2  2  15 Dora
```

```
>typeof(x)
"list"
>class(x)
"data.frame"
```

* Creation of Dataframe in R:-

→ We can create a dataframe using the "data.frame()" function.

```
>x ← dataframe("SN"=1:2, "Age"=c(21,15), "Name"=c("John","Dora"))
>str(x)
'data.frame' : 2 obs of variables
 $SN : int 1 2
 $Age : num 21 15
 $Name : factor w/2
  levels: "Dora" "John" : 2 1
```

* Accessing components of a data frame :-

→ components of a dataframe can be accessed like a list (or) matrix.

Ex:-

```
x["Name"]

  Name
1 John
2 Dosa.
```

```
x$Name
[1] "John" "Dosa"

x[["Name"]]
"John" "Dosa"
```

8) What is R ? Explain characteristics of R ?

a)

* R Programming :-

→ R is a Programming language and software environment for statistical analysis, graphic representation and reporting..

→ R was created by Ross Ihaka and Robert Gentleman at the university of Auckland, New Zealand and is currently developed by the R Development Core Team.

→ The core of R is an interpreted computer language which allows branching and looping as well as modular Programming using functions.

→ R allows integration with the procedures written in C, C++, .Net, Python or FORTRAN languages for efficiency.

* Characteristics of R :-

→ R is an open-source software environment. It is free of cost and can be adjusted and adapted according to the user's and the project's requirements.

→ R can produce static graphics with production quality visualizations and has extended libraries providing interactive graphic capabilities. This makes data visualization and data representation very easy.

→ R can be used to perform simple and complex mathematical and statistical calculations on data objects of a wide variety.

→ R has effective data handling and storage facility.

→ R is machine independent. It supports the cross platform operation. Thus, it is usable on many different operating system.

→ R has a very comprehensive development environment meaning it helps in statistical computing as well as software development. R is an object-oriented programming language.

→ R is an interpreted language which means that it does not need a compiler to make a program from the code. R directly interprets provided code into lower-level calls and pre-compiled code.

→ R can be easily paired with other data processing and distributed computing technologies like Hadoop and Spark. It is possibely to remotely use a Spark cluster to process large datasets using R.

2b) Discuss R functions with an example.

→ Functions are used to logically break our code into simpler parts which become easy to maintain and understand.

Syntax:-

```
func_name <— function(argument) {
    statement
}
```

- Here, we can see that the reserved word function is used to declare a function in R.
- The statements within the curly braces from the body of the function. These braces are optional if the body contains only a single expression.
- Finally, this function object is given a name by assigning it to a variable, func_name.

Example :-

```
Pow <— function(x, y) {
    result <— x ^ y
    Print(Paste(X, "raised to the power", y, "is",
                result))
```

How to call a function:

We can call the above function as follows:

```
> Pow(8, 2)
[1] "8 raised to the power 2 is 64"
```

Here, the arguments used in the function declaration (x and y) are called formal arguments and those used while calling the function are called actual arguments.