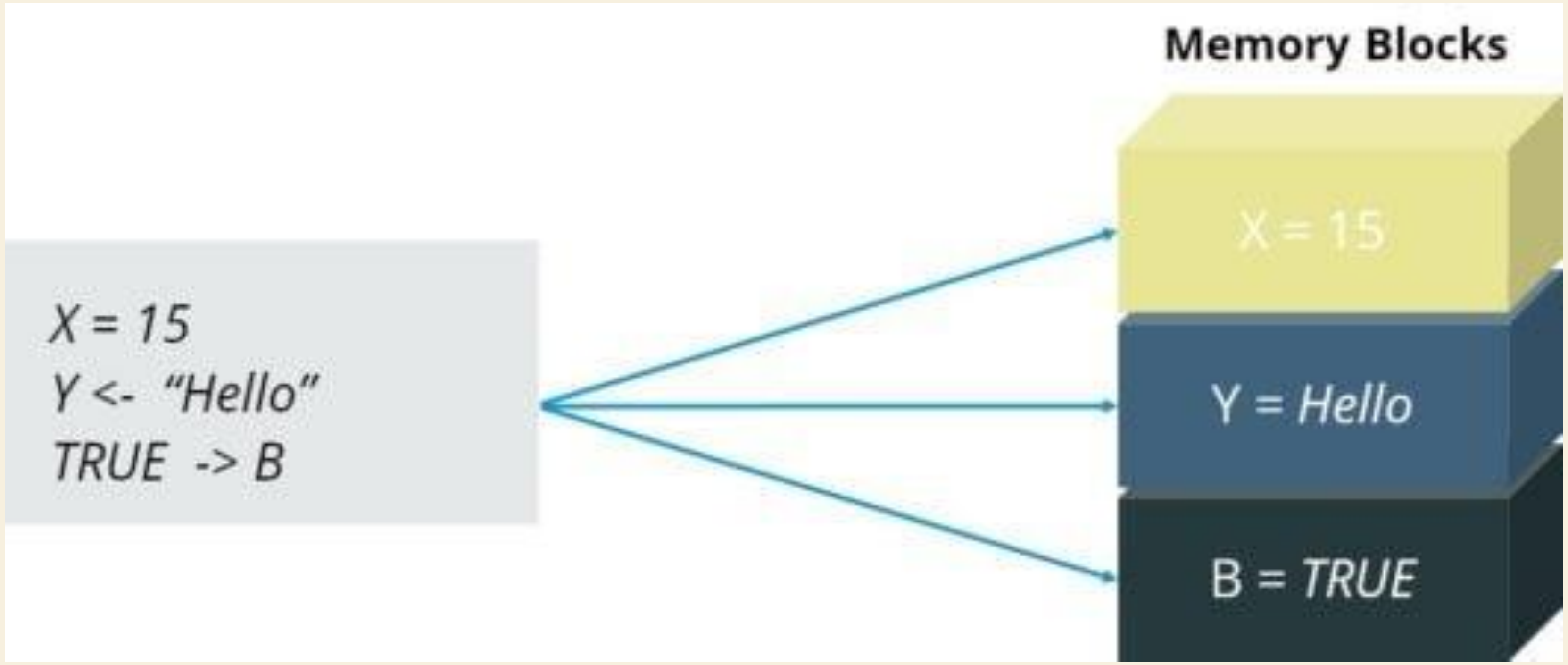
INTRODUCTION TO ‘R’

DATA TYPES AND FUNCTIONS

R PROGRAMMING: VARIABLES

- *Variable - Name to a memory location containing a value*

- *A variable in R can store* ***Numeric values, Complex Values, Words, Matrices*** *and even a* ***Table***



R PROGRAMMING: DATA TYPES

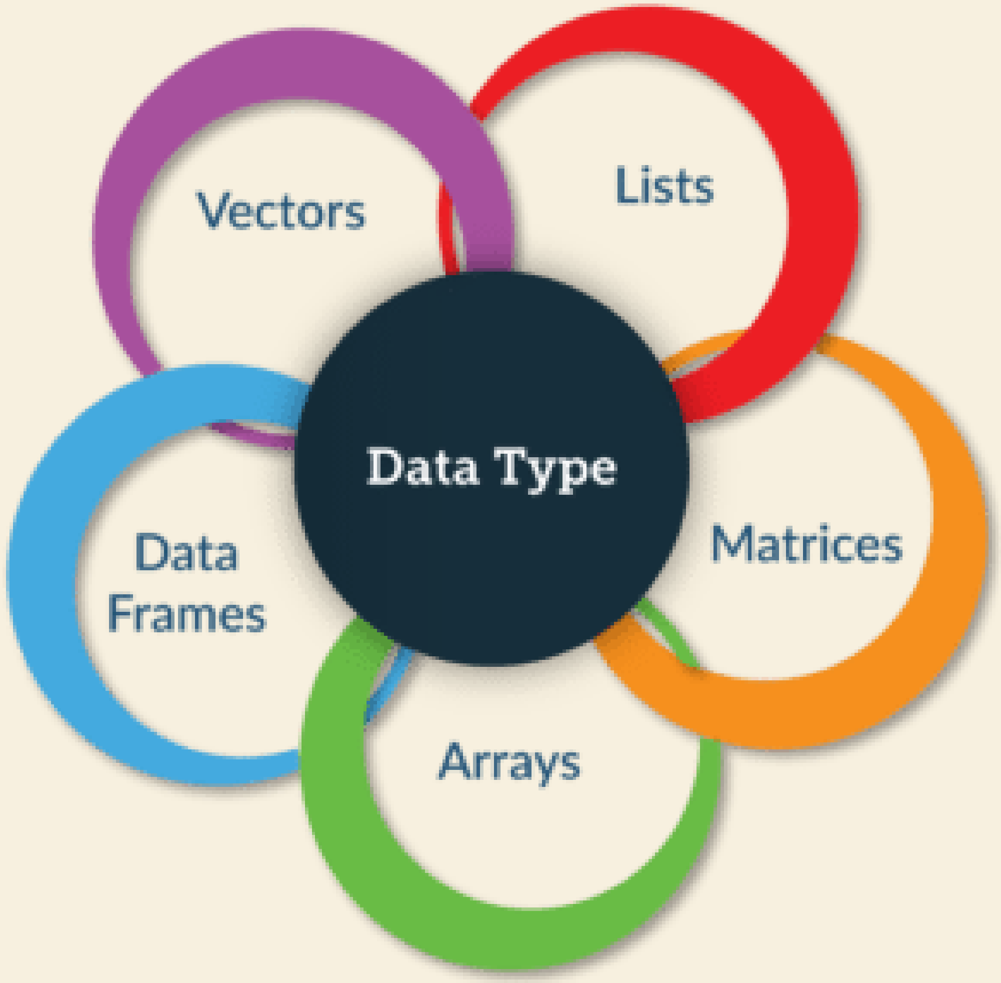
- *In R, a variable itself is not declared of any data type*
- *Rather it gets the data type of the R object assigned to it*
- *R is called a **dynamically typed language**, which means that we can change a data type of the same variable again and again when using it in a program.*

R PROGRAMMING: DATA TYPES

- *Data Types specifies*
  - *which* ***type of value a variable*** *has*
  - *what type of* ***mathematical, relational*** *or* ***logical operations*** *can be applied to it without causing an error.*

R PROGRAMMING: DATA TYPES

- *There are many data types in R, However below are the most frequently used ones:*



DATA TYPES - VECTORS

- *Vectors are the most basic R data objects*

- *A sequence of data elements of the same basic type.*

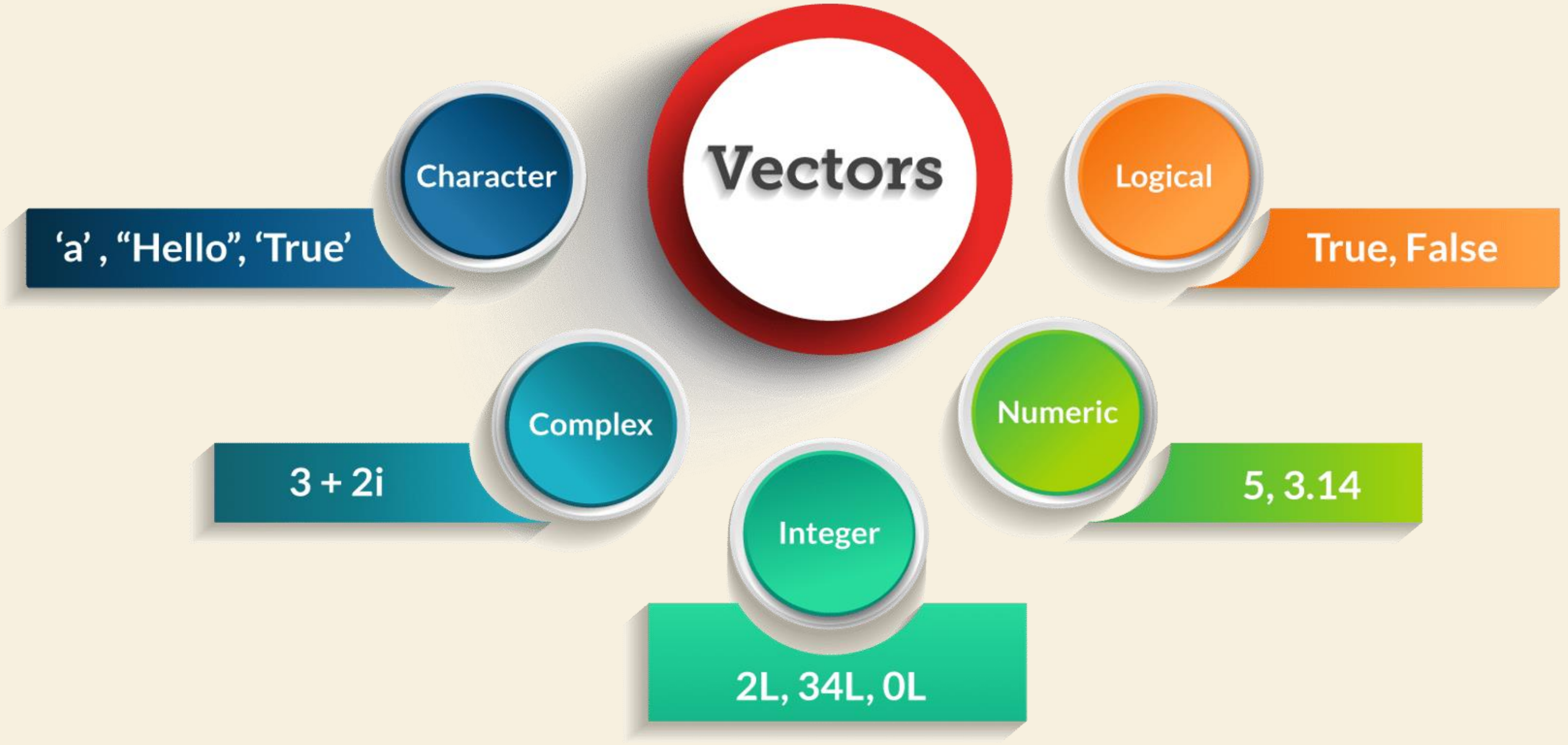
Ex:

*Defining and initializing a vector called **test_vector***

```
test_vector = c(1, 3, 5 ,79)
or
test_vector <-  c(1, 3, 5 ,79)
or
c(1, 3, 5 ,79) -> test_vector
```

DATA TYPES - VECTORS

- *There are* ***Five types*** *of atomic vectors.*



DATA TYPES - VECTORS

- *Logical:* *It is used to store logical value like* ***TRUE*** *or* ***FALSE****.*
- *Numeric:* *It is used to store* ***both positive*** *and* ***negative numbers*** *including* ***real number***
  - *Eg: 25, 7.1145 , 96547*
- *Integer:* *It holds all the integer values i.e.* ***all the positive*** *and* ***negative whole numbers***
  - *Eg: 45, -856, 0*

DATA TYPES - VECTORS

- *Complex: These are of the form* ***x + yi****, where x and y are numeric and i represents the square root of -1.*
  - *Eg: 4+3i*

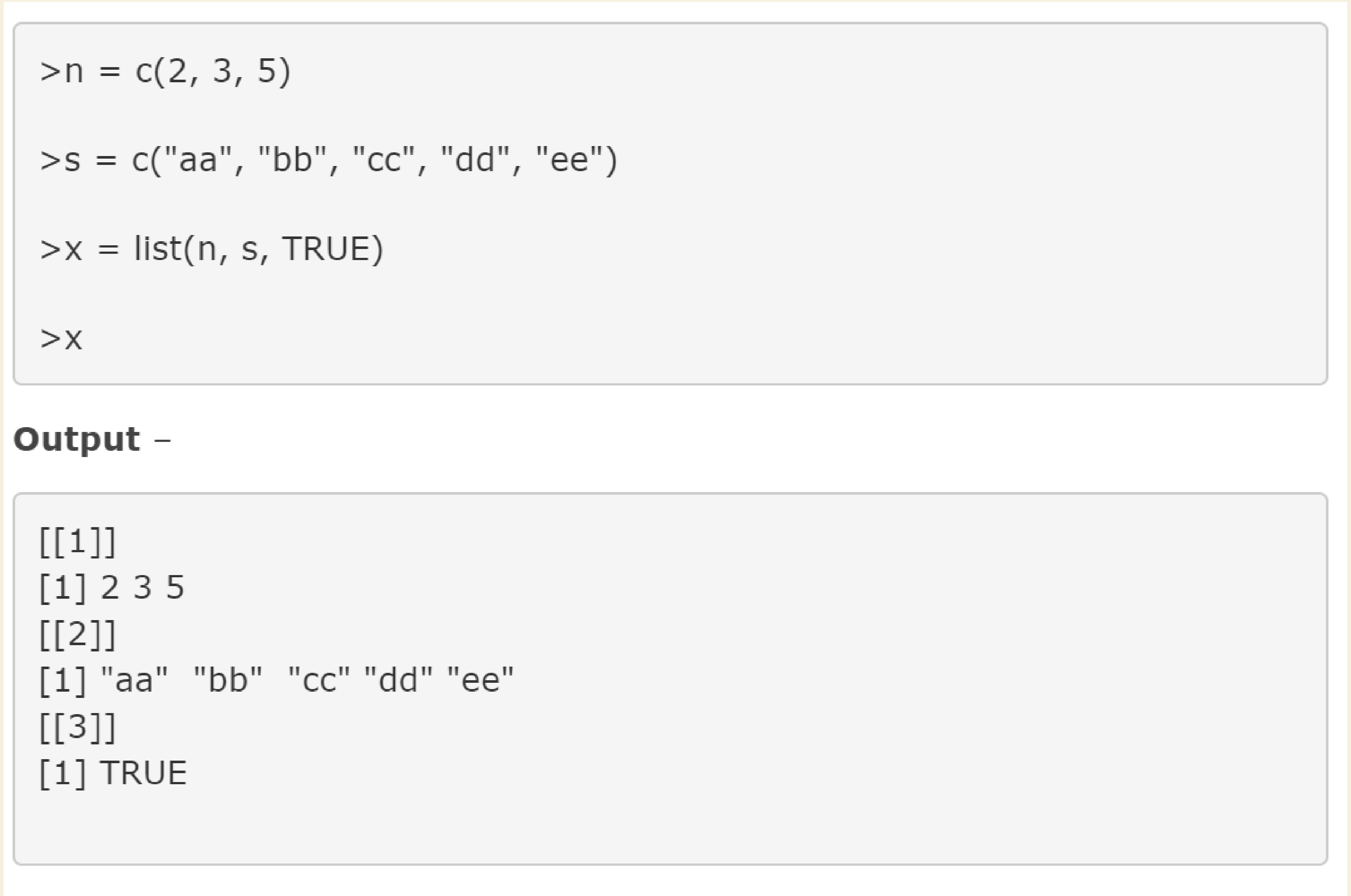
- *Character:*
  - *It is used to store a single character,*
  - *group of characters(words)*
  - *a group of words together.*
  - *They may be defined in either single quotes or double quotes.*

  - *Eg: “VIT-AP", 'R is Fun to learn'.*

DATA TYPES - LISTS

- *Lists are quite similar to vectors with a small difference*
- *Lists are the R objects which can contain elements of different types like*
  - *numbers*
  - *strings*
  - *vectors*
  - *list*

DATA TYPES - LISTS



```
>n = c(2, 3, 5)

>s = c("aa", "bb", "cc", "dd", "ee")

>x = list(n, s, TRUE)

>x
```

Output –

```
[[1]]
[1] 2 3 5
[[2]]
[1] "aa"  "bb"  "cc" "dd" "ee"
[[3]]
[1] TRUE
```

DATA TYPES - ARRAYS

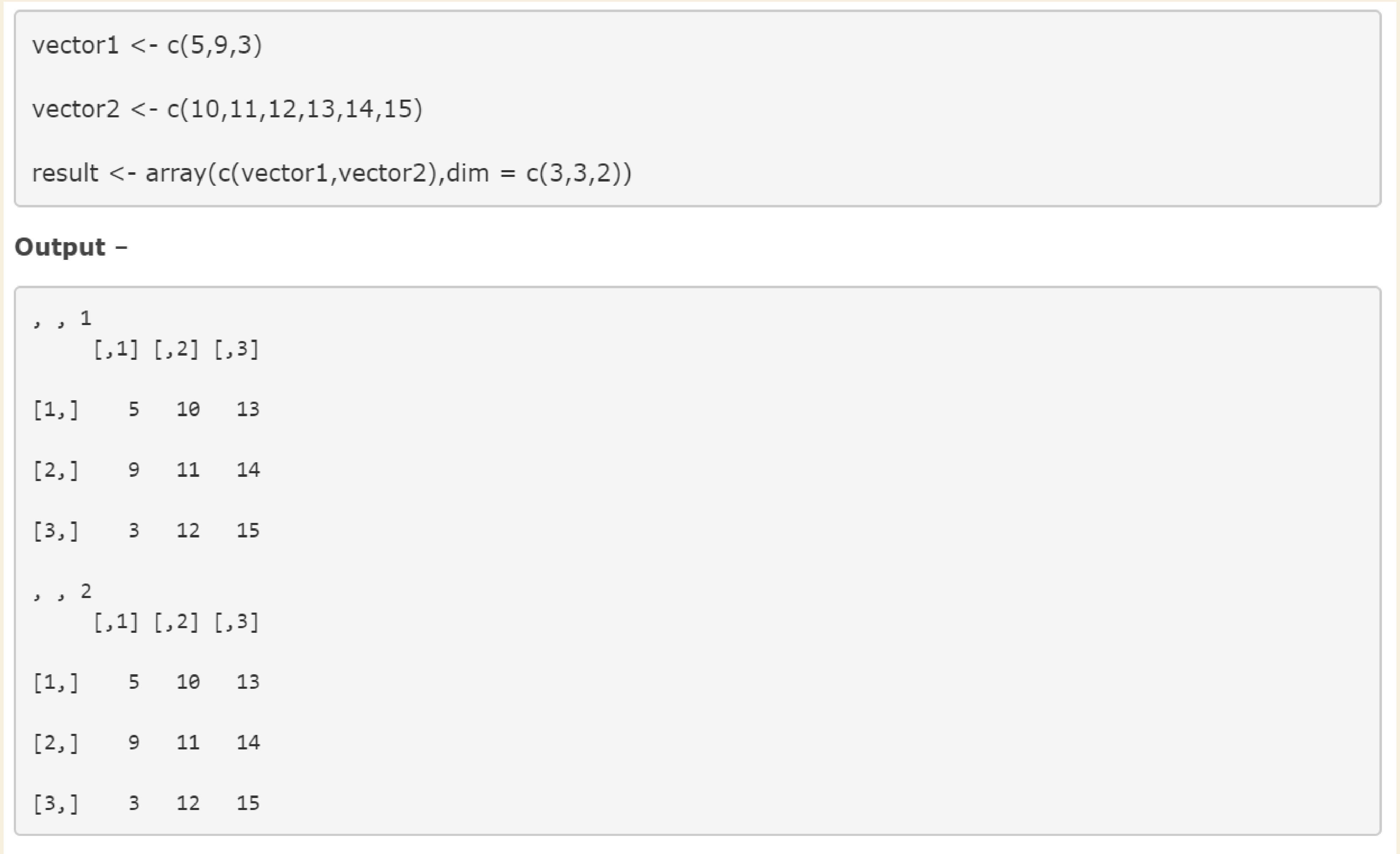
- *Arrays are the R data objects which can store data in* **more than two dimensions**.

- *It takes vectors as input and uses the values in the dim parameter to create an array.*

  Example – array(data, dim, dimnames)

  - data *is the input vector which becomes the data elements of the array.*
  - dim *is the dimension of the array, where you pass the* ***number of rows, column*** *and* ***the number of matrices to be created***
  - dimname *is the names assigned to the rows and columns.*

DATA TYPES - ARRAYS



```
vector1 <- c(5,9,3)

vector2 <- c(10,11,12,13,14,15)

result <- array(c(vector1,vector2),dim = c(3,3,2))
```

Output –

```
, , 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
```

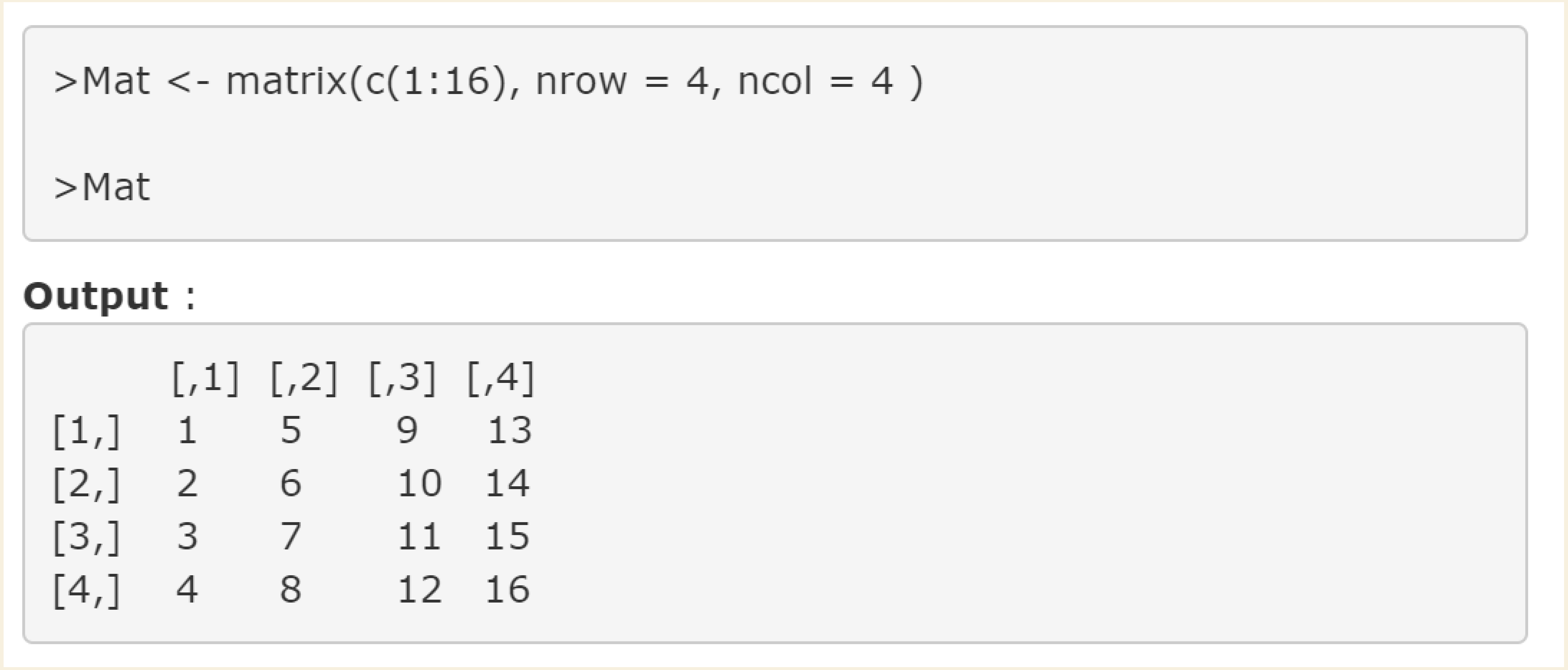
DATA TYPES - MATRICES

- *Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout.*
- *A **Matrix** is created using the **matrix()** function.*

  ***Example:** **matrix(data, nrow, ncol, byrow, dimnames)** where,*

  - ***data** is the input vector which becomes the data elements of the matrix.*
  - ***nrow** is the number of rows to be created.*
  - ***ncol** is the number of columns to be created.*
  - ***byrow** is a logical clue. If TRUE then the input vector elements are arranged by row.*
  - ***dimname** is the names assigned to the rows and columns.*

DATA TYPES - MATRICES

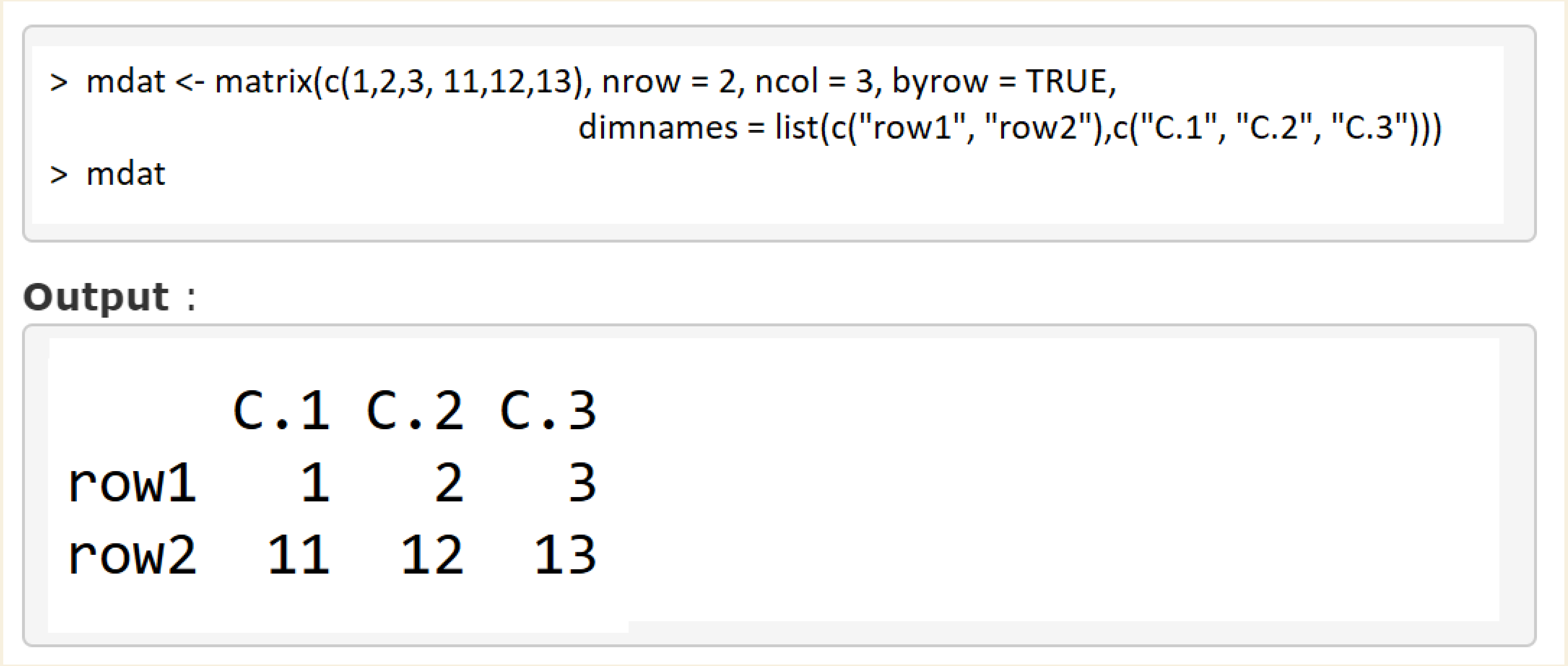


```
>Mat <- matrix(c(1:16), nrow = 4, ncol = 4 )

>Mat
```

Output :

```
      [,1] [,2] [,3] [,4]
[1,]   1    5     9   13
[2,]   2    6     10  14
[3,]   3    7     11  15
[4,]   4    8     12  16
```

DATA TYPES - MATRICES



```
> mdat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3, byrow = TRUE,
                                       dimnames = list(c("row1", "row2"),c("C.1", "C.2", "C.3")))
> mdat
```

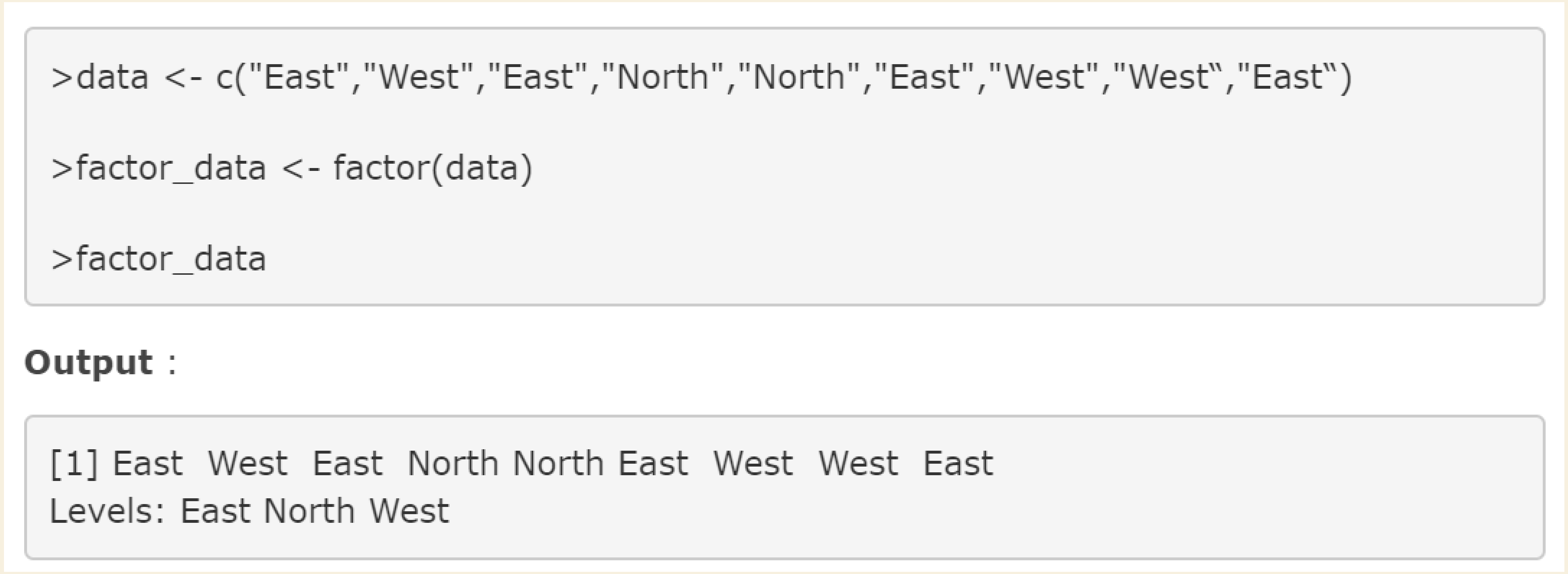
Output :

```
     C.1 C.2 C.3
row1   1   2   3
row2  11  12  13
```

DATA TYPES - FACTORS

- *Factors are the data objects*
- *Used to categorize the data and store it as levels*
- *They can store both strings and integers*
- *They are useful in data analysis for statistical modeling*

DATA TYPES - FACTORS

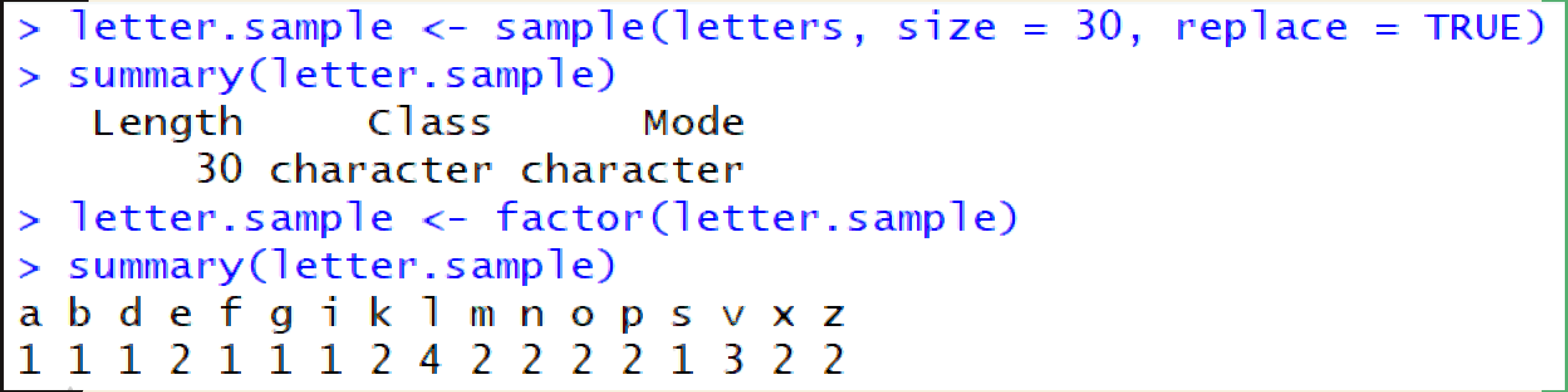


```
>data <- c("East","West","East","North","North","East","West","West“,"East“)

>factor_data <- factor(data)

>factor_data
```

Output :

```
[1] East  West  East  North North East  West  West  East
Levels: East North West
```

DATA TYPES - FACTORS

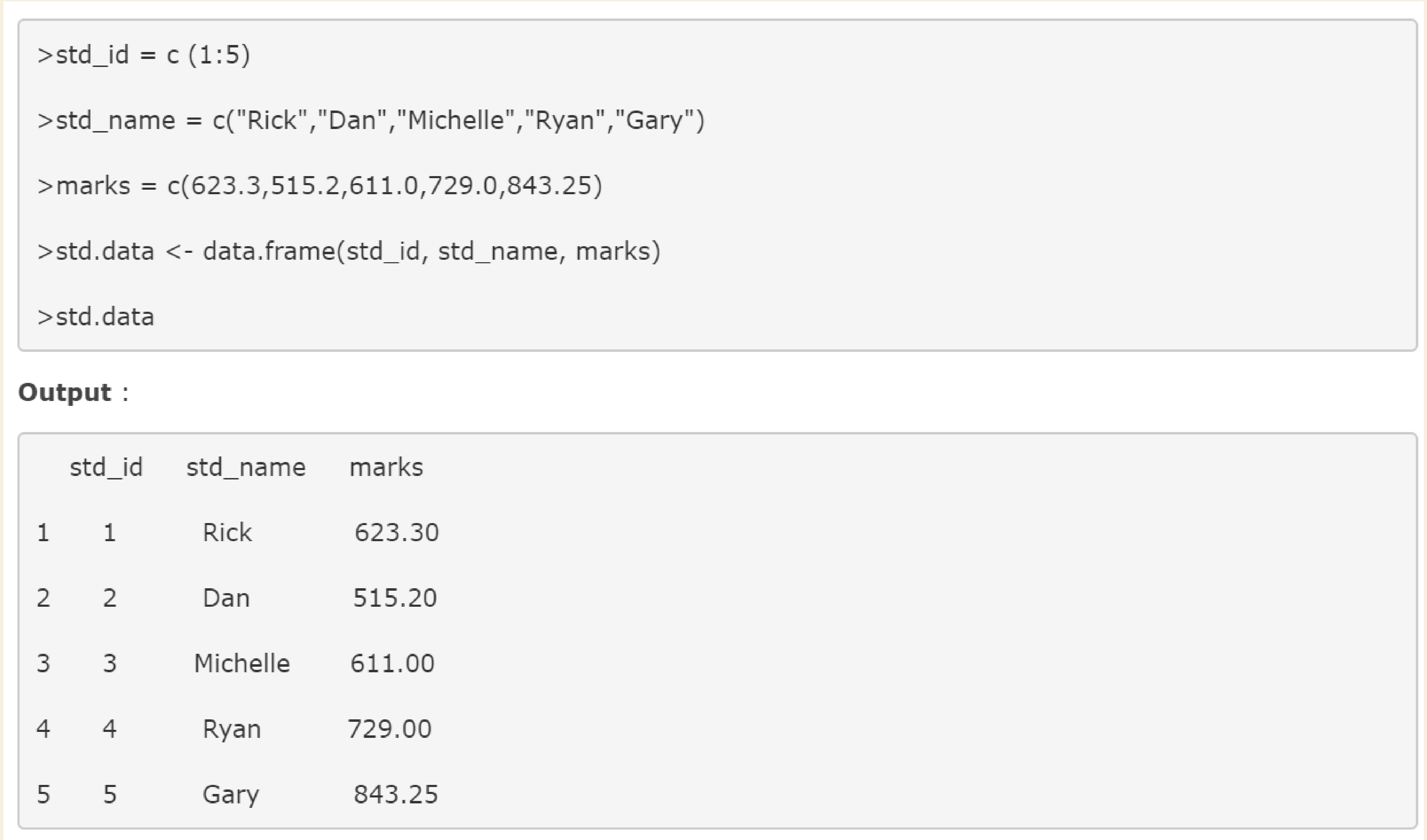


```
> letter.sample <- sample(letters, size = 30, replace = TRUE)
> summary(letter.sample)
   Length     Class      Mode 
       30 character character 
> letter.sample <- factor(letter.sample)
> summary(letter.sample)
a b d e f g i k l m n o p s v x z 
1 1 1 2 1 1 1 2 4 2 2 2 2 1 3 2 2 
```

DATA TYPES – DATA FRAMES

- *A Data Frame is a table or a two-dimensional array-like structure*
- *Each column contains values of one variable and each row contains one set of values for each column.*
- *The characteristics of a Data Frame that needs to be considered every time we work with them:*
    - *The column names should be non-empty.*
    - *Each column should contain the same amount of data items.*
    - *The data stored in a data frame can be of numeric, factor or character type.*
    - *The row names should be unique.*

DATA TYPES – DATA FRAMES



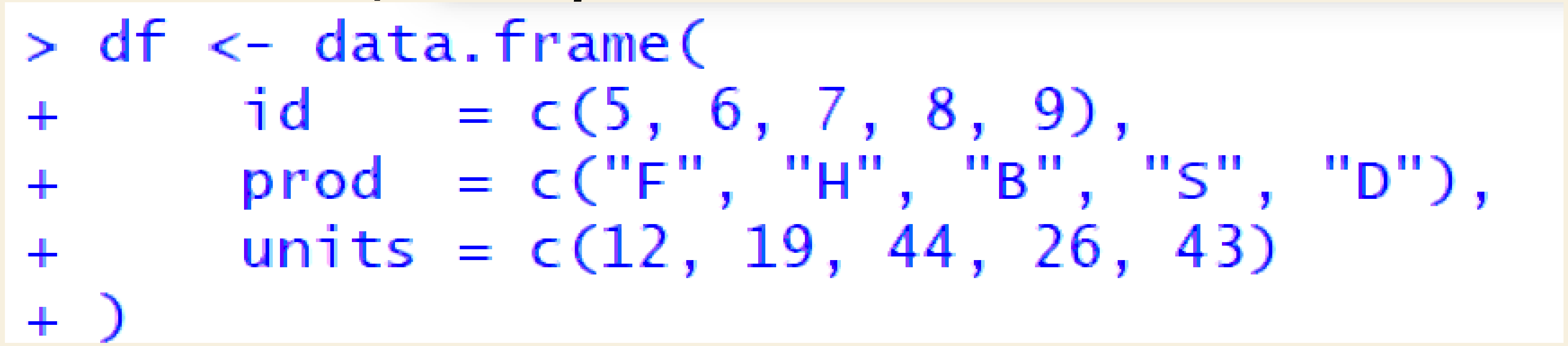
```
>std_id = c (1:5)

>std_name = c("Rick","Dan","Michelle","Ryan","Gary")

>marks = c(623.3,515.2,611.0,729.0,843.25)

>std.data <- data.frame(std_id, std_name, marks)

>std.data
```

Output :

```
    std_id    std_name    marks

1      1         Rick          623.30

2      2         Dan           515.20

3      3        Michelle      611.00

4      4         Ryan         729.00

5      5         Gary          843.25
```

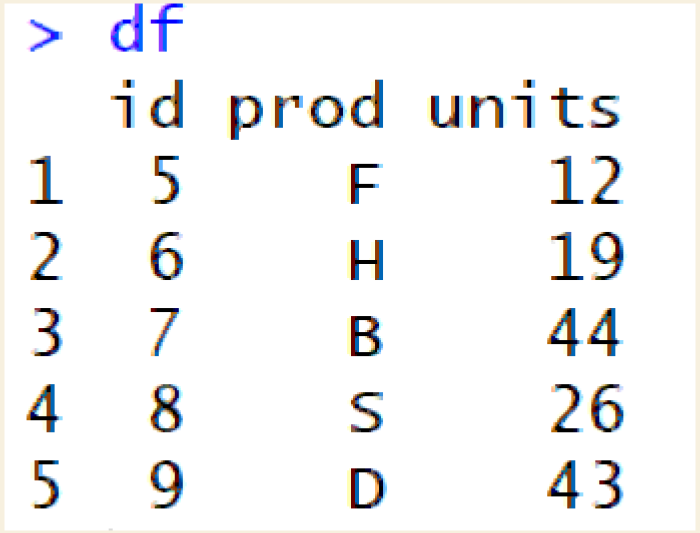
DATA TYPES – DATA FRAMES

- *Create a data frame* ***df***



```
> df <- data.frame(
+       id    = c(5, 6, 7, 8, 9),
+       prod  = c("F", "H", "B", "S", "D"),
+       units = c(12, 19, 44, 26, 43)
+ )
```

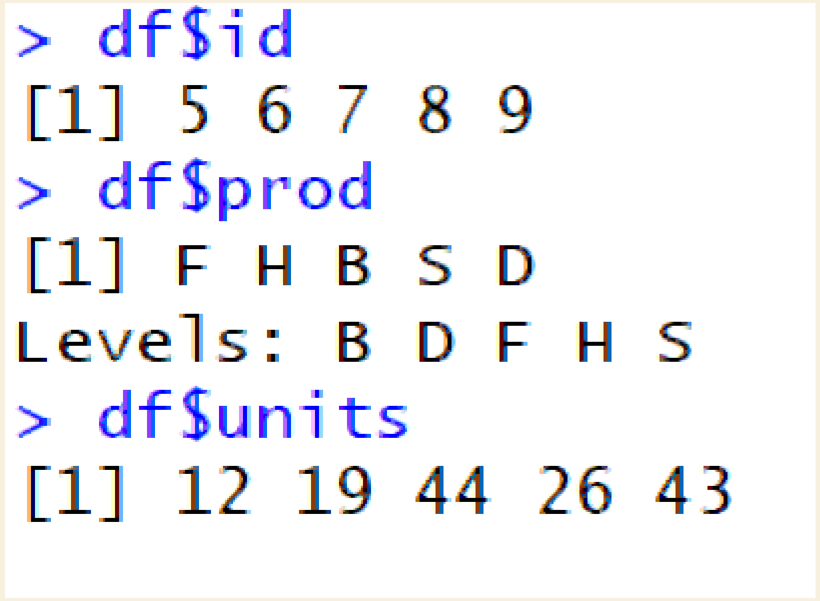
- *Print the data frame* ***df***



```
> df
  id prod units
1  5    F    12
2  6    H    19
3  7    B    44
4  8    S    26
5  9    D    43
```

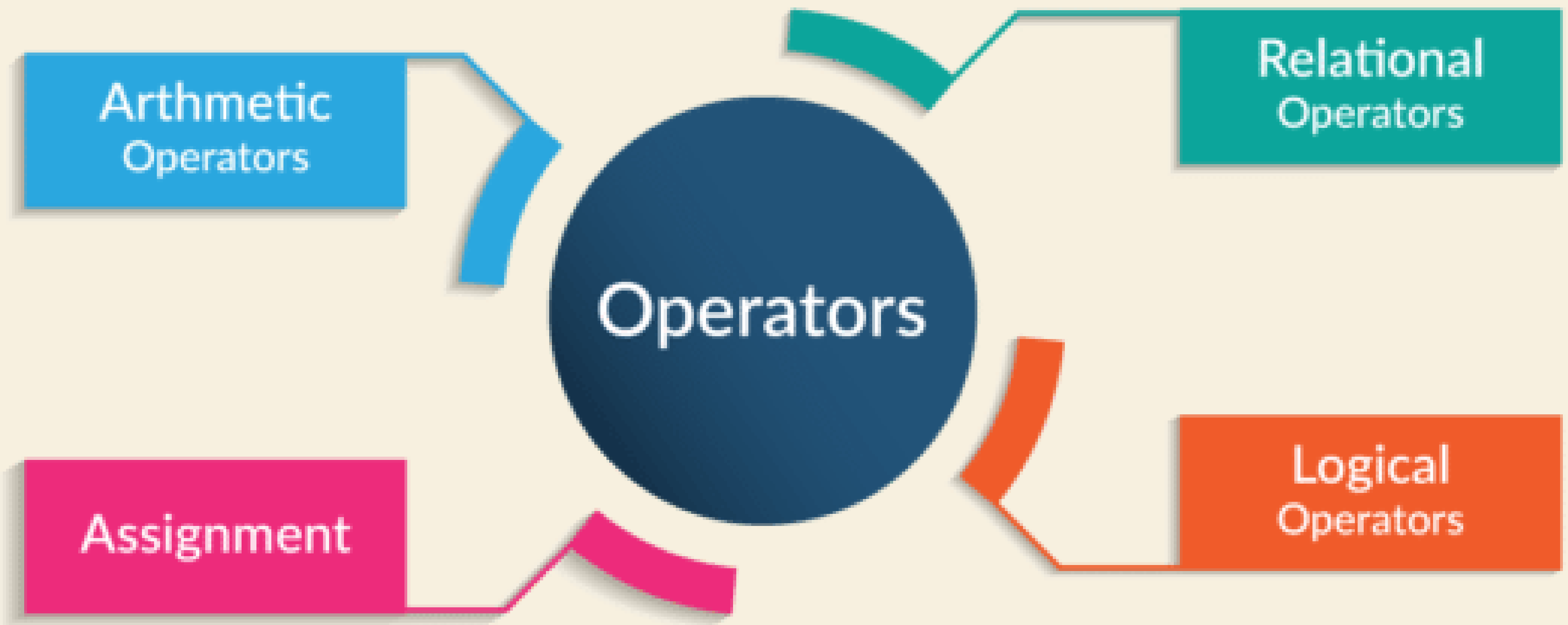
DATA TYPES – DATA FRAMES

- *Print a specific attribute (column) in* ***df***

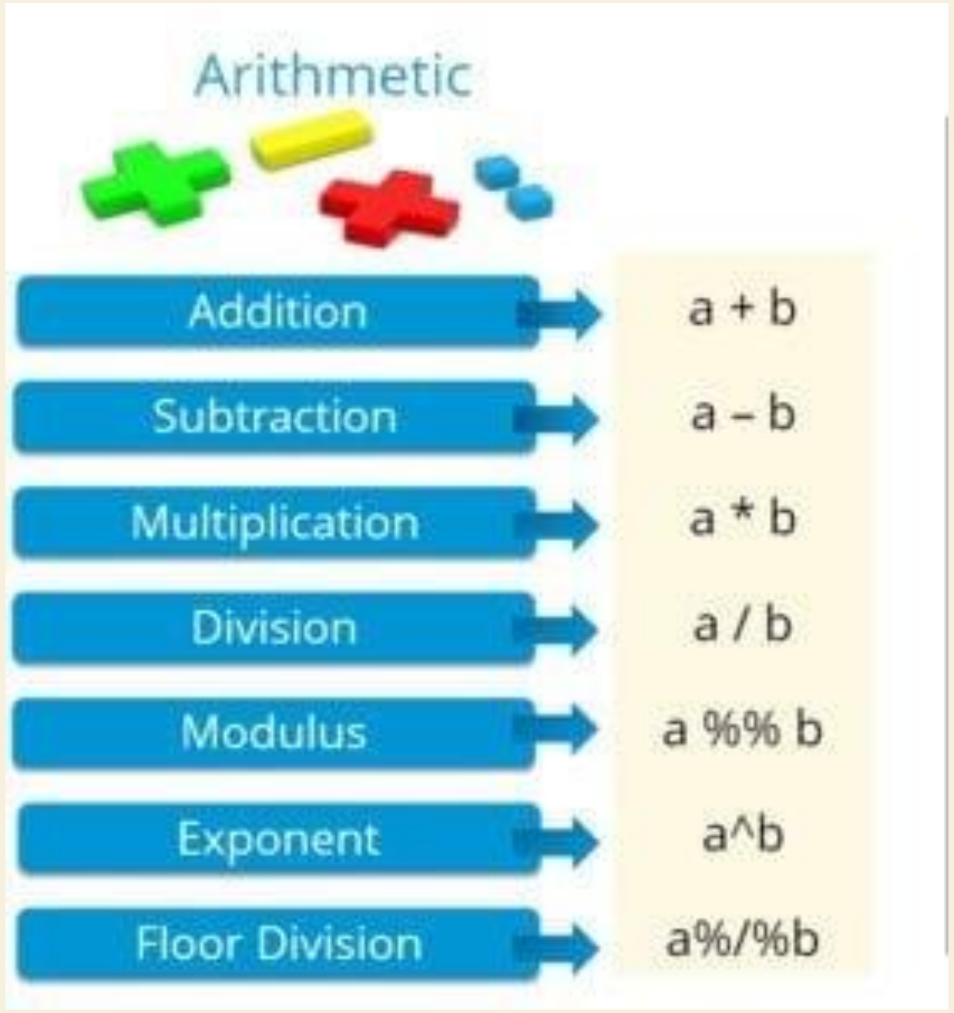


```
> df$id
[1] 5 6 7 8 9
> df$prod
[1] F H B S D
Levels: B D F H S
> df$units
[1] 12 19 44 26 43
```

R PROGRAMMING: DATA OPERATORS



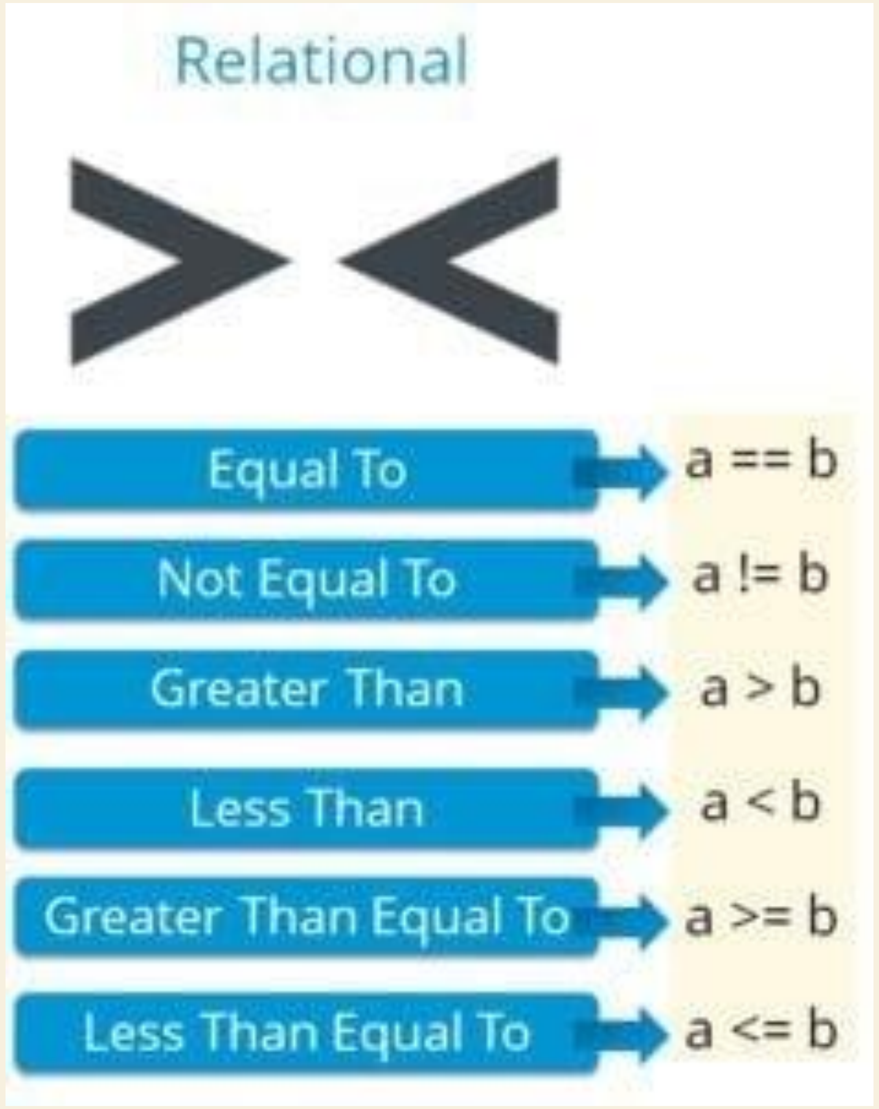
DATA OPERATORS - ARITHMETIC



Arithmetic

Addition	a + b
Subtraction	a – b
Multiplication	a * b
Division	a / b
Modulus	a %% b
Exponent	a^b
Floor Division	a%/%b

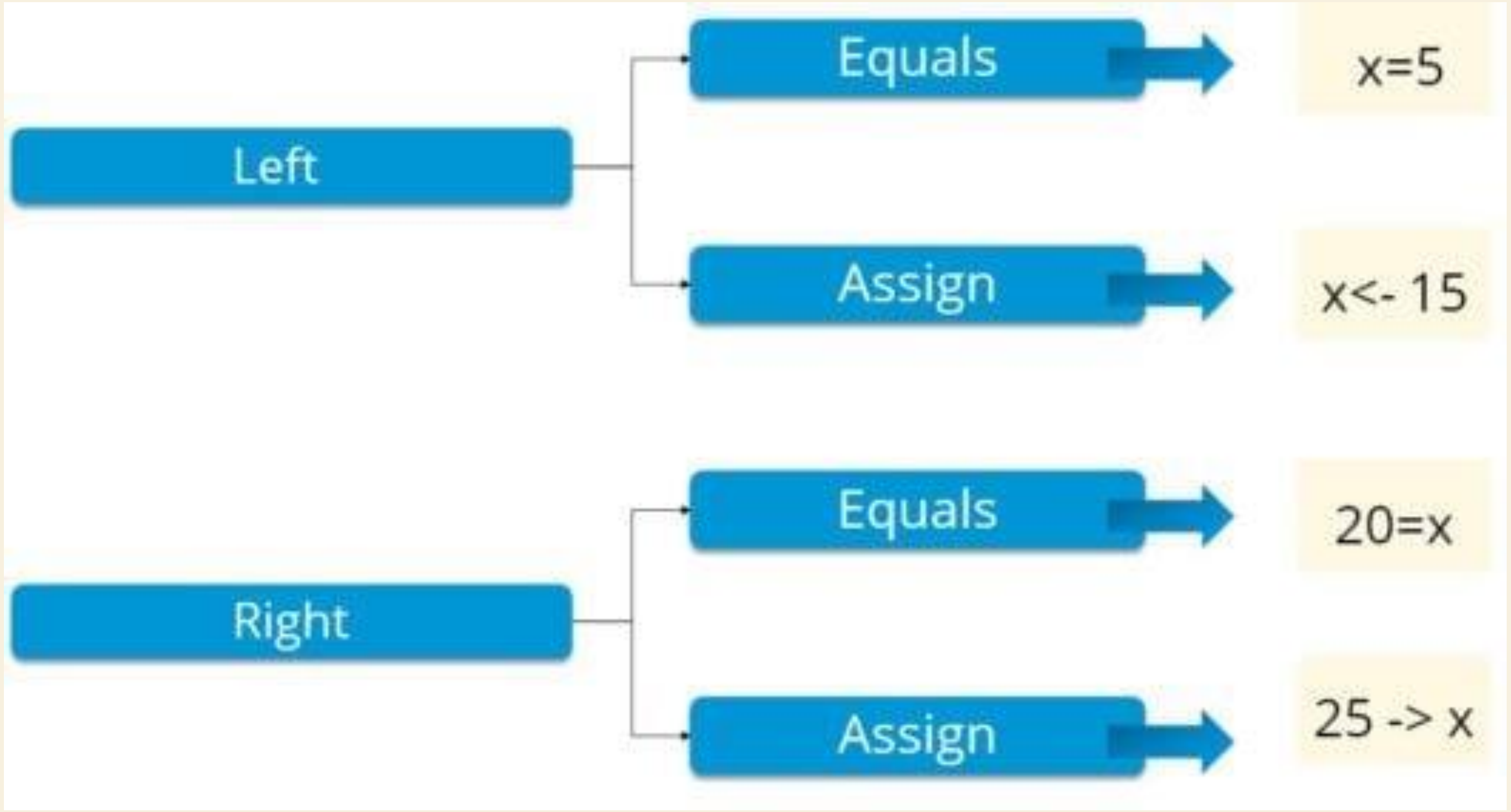
DATA OPERATORS - RELATIONAL



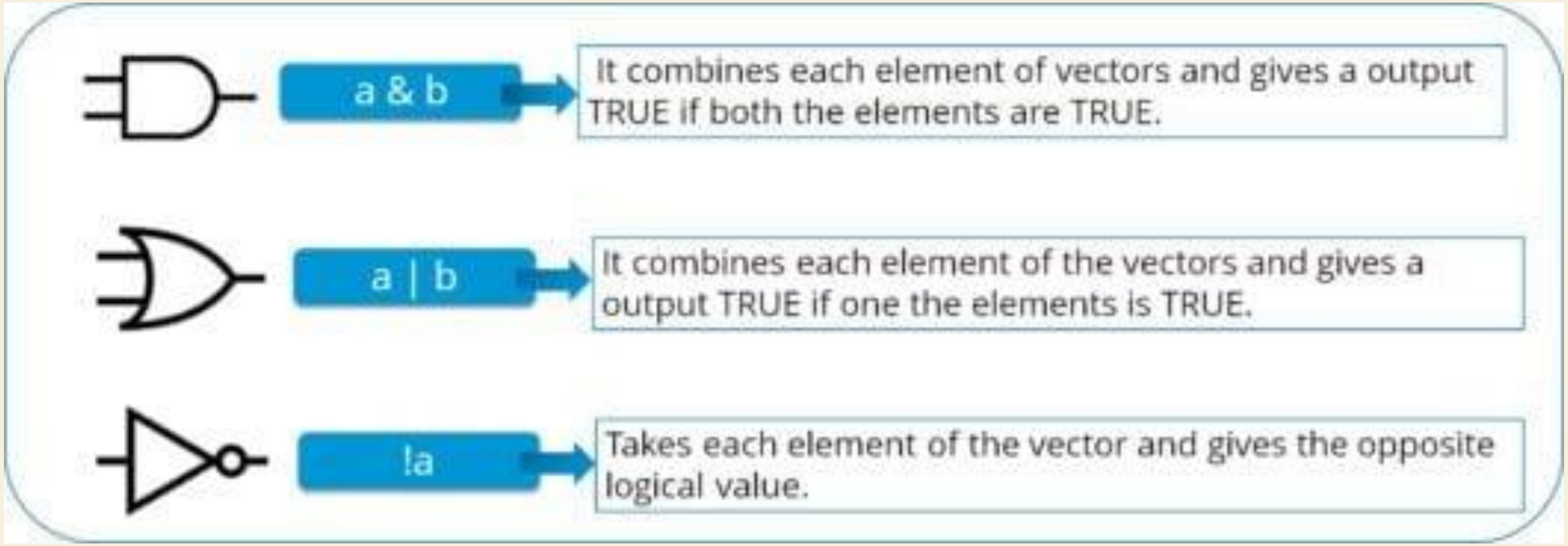
Relational

Equal To	a == b
Not Equal To	a != b
Greater Than	a > b
Less Than	a < b
Greater Than Equal To	a >= b
Less Than Equal To	a <= b

DATA OPERATORS - ASSIGNMENT



DATA OPERATORS - LOGICAL



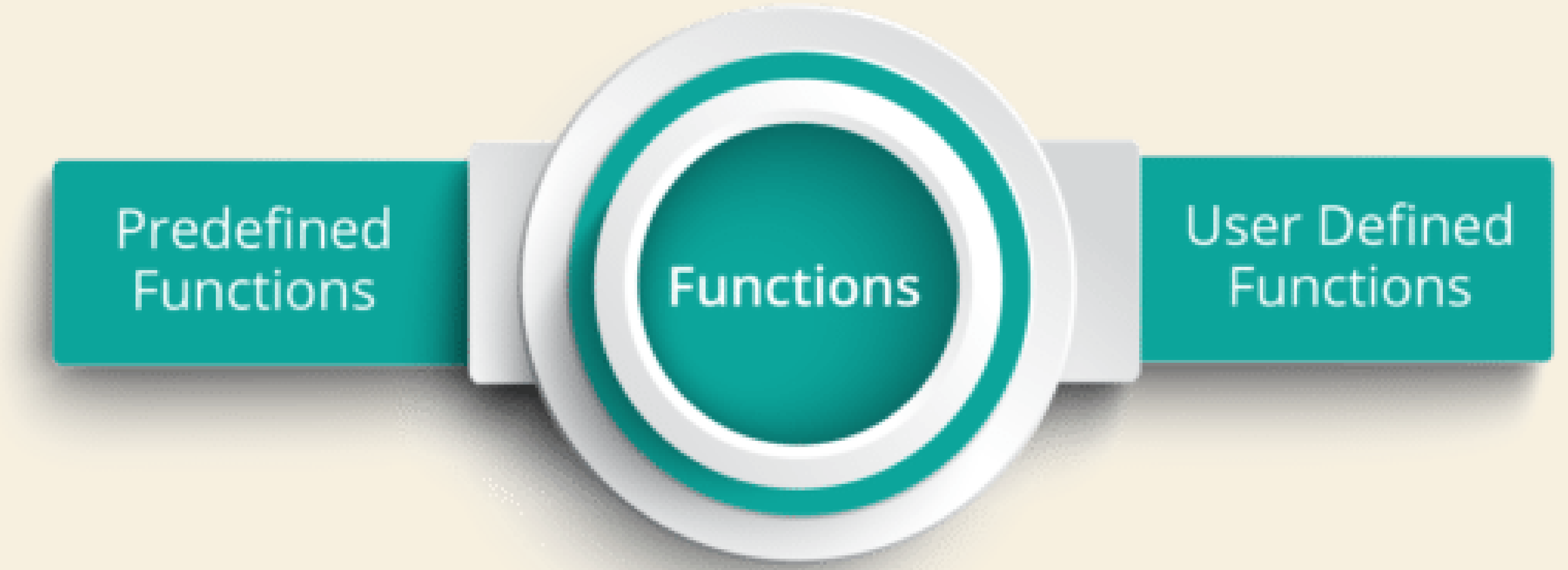
It combines each element of vectors and gives a output TRUE if both the elements are TRUE.

It combines each element of the vectors and gives a output TRUE if one the elements is TRUE.

Takes each element of the vector and gives the opposite logical value.

R PROGRAMMING: FUNCTIONS

- *A function is a* ***block of organized, reusable code***
- *A function is used to perform* ***a single, related action****.*
- *There are mainly two types of functions in R:*



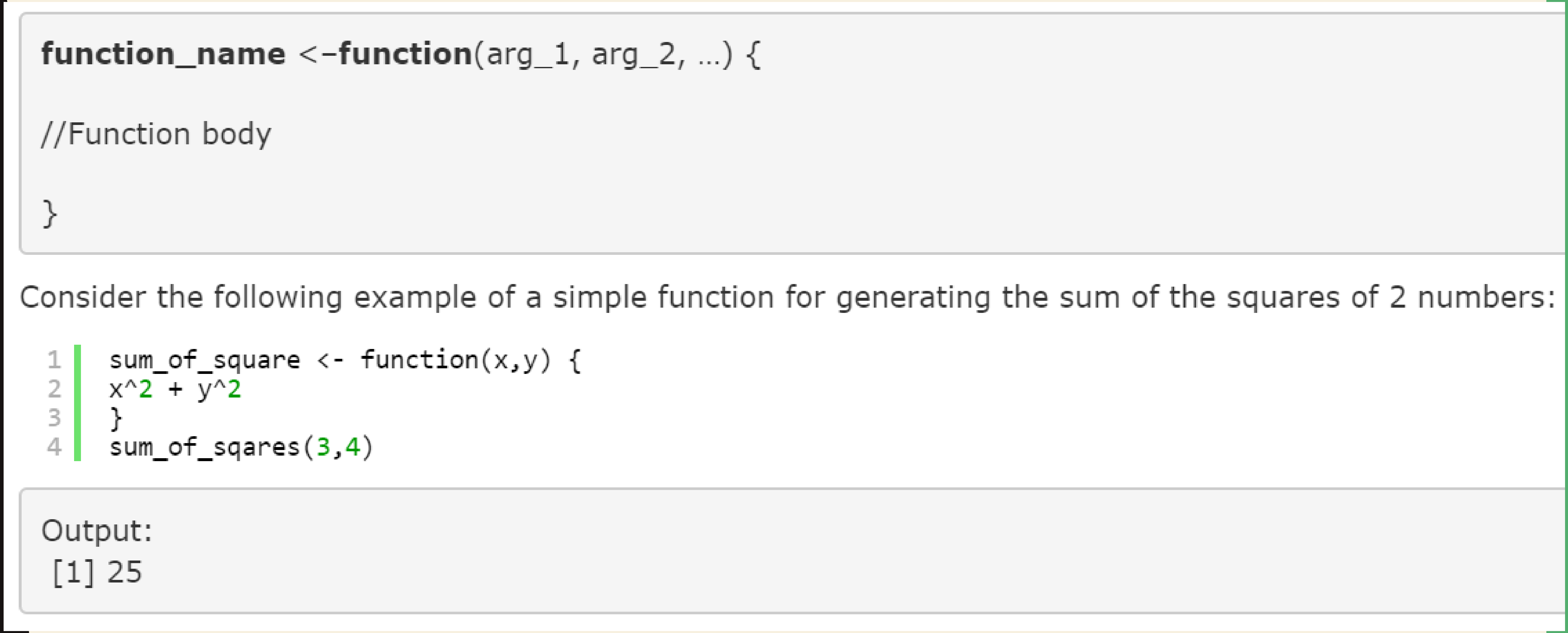
R PROGRAMMING: FUNCTIONS

Predefined Functions: These are built in functions that can be used by the user to make their work easier.

Eg: mean(x), sum(x) ,sqrt(x), toupper(x)

User Defined Functions: These functions are created by the user to meet a specific requirement of the user

R PROGRAMMING: FUNCTIONS



```
function_name <-function(arg_1, arg_2, ...) {

//Function body

}
```

Consider the following example of a simple function for generating the sum of the squares of 2 numbers:

```
sum_of_square <- function(x,y) {
x^2 + y^2
}
sum_of_sqares(3,4)
```

```
Output:
 [1] 25
```