R- Basic WORKSHOP DAY 2

Chandan Kumar Pandey

06-10-2022

Today we will start learning the computer language of R. Now it is very important to understand that **R** is not a tools or application but a computer language. Therefore, it is required to have a basic computation logic to start coding in **R**. This document will help you to understand some of the basic theory and logic of coding. Any coding or computation language is set of instruction we use to communicate with the computer. In order to understand different type of computer language you can visit wikipedia URL . So without any further due let us start with our first program

Our first program to print "Hello World"

```
print("Hello world")
```

```
## [1] "Hello world"
```

As you can see above the we wrote the command called *print()* and instruction/argument "Hello world" inside the parenthesis. What R does when we run the command , it print the string/sentence inside the parenthesis.

Concept of variable.

In R, variable is a place holder of value in the program. For example below.

```
x<-5
y<-8
z<-5
```

x, y and z are the variable whose value are 5,8, and 5 respectively. In this case if we have to print the value of z by just applying the command *print(z)*.

```
print(z)
```

```
## [1] 5
```

The output in this case is 5 not "z".

Concept of operator

Operator are the symbol which allow R to operate the relationship between different variable or data. There are various kinds of operator in R but most common types are

1. Arithmetic operators

```
#ADDITION
2+4
```

```
## [1] 6
```

```
# Subtraction
4-2
```

```
## [1] 2
```

```
#Multiply
2*3
```

```
## [1] 6
```

```
#Divide
3/2
```

```
## [1] 1.5
```

```
#exponential
3^2
```

```
## [1] 9
```

```
# Reminder
100%%23
```

```
## [1] 8
```

```
## Integer division
100%/%23
```

```
## [1] 4
```

2. Comparison operators: this operator is require compare two values or variable. The output of this operator is either true or false.

```
#Equal to
2==3
```

```
## [1] FALSE
```

```r
# Not equal to
2!=3
```

```
## [1] TRUE
```

```r
#Greater than
2>3
```

```
## [1] FALSE
```

```r
# less than
2<3
```

```
## [1] TRUE
```

```r
#greater than equal to
2>=3
```

```
## [1] FALSE
```

```r
#less than equal to
2<=3
```

```
## [1] TRUE
```

3. Logical Operators : AND , OR and NOT. Classic logic gate concept. The output is either true or false.

```r
# AND : Compare two condition and return True only if both condition is true . The operator is &
z<-2==2 & 3==3
print(z) ## both condition it true
```

```
## [1] TRUE
```

```r
y<-2>3 & 3==3
print(y) # first condition is true and second condition is false. To y is false.
```

```
## [1] FALSE
```

```r
# OR operator : Compare two condition and return True  if any one condition is true. the operator is |
z<-2==2 | 3==3
print(z) ## both condition it true
```

```
## [1] TRUE
```

```r
y<-2>3 | 3==3
print(y) # first condition is true and second condition is false. To y is true
```

```
## [1] TRUE
```

```
# NOT : this return true if value is false and vice-versa. The operator symbol is !
z<-2==2 | 3==3 ## both condition it true
y<-!z
print(y)
```

```
## [1] FALSE
```

3. Other Miscellaneous Operators; some examples are

- : Creates a series of numbers in a sequence
- %in% Find out if an element belongs to a vector

Concept of data type/class in R

Even though are any different type or class of data in R. However, the 5 common type of data are

1. numeric : They are number such as 10.5, 4.2,4.0
2. integer : As name suggest, they contain integers only. Example, 1L, 2L; the letter L suggest it is integer.
3. Character (also called as string): These are letters which are written under quotes,example "HEllO","50","FLASE",etc.
4. Logical (Boolean): TRUE or FALSE
5. Complex numbers. in the form of a+bi

We can use command/function called *class()* to check the data type.

```
# numeric
x <- 10
class(x)
```

```
## [1] "numeric"
```

```
# integer
y <- 20L
class(y)
```

```
## [1] "integer"
```

```
# character/string
x <- "Hello world"
class(x)
```

```
## [1] "character"
```

```
# logical/boolean
x <- TRUE
class(x)
```

```
## [1] "logical"
```

```
# complex
x <- 7i + 88
class(x)
```

```
## [1] "complex"
```

You have to note few minute difference here.

- FALSE is logical datatype while "FALSE" is character.
- 10 is numeric data type, 10L is integer and "10" is character.

```
Var1<-FALSE
class(Var1)
```

```
## [1] "logical"
```

```
Var2<-"FALSE"
class(Var2)
```

```
## [1] "character"
```

```
num10<-10
class(num10)
```

```
## [1] "numeric"
```

```
int10<-10L
class(int10)
```

```
## [1] "integer"
```

```
car10<-"10"
class(car10)
```

```
## [1] "character"
```

Concept of Objects.

There are 5 type or broad classification of variable types in R. They are

1. Vector : They are the one dimensional list of items that are of same type. We need to use function *c()* and separate items using "," to create a vector.

```
#this is the example for vector of string
names<-c("chandan","Pandey","Kumar")
print(names)
```

```
## [1] "chandan" "Pandey"  "Kumar"
```

```r
##vector of numbers
numbers<-c(1,2.3,6,3)
print(numbers)
```

```
## [1] 1.0 2.3 6.0 3.0
```

To access the vector items we can refer to index or position of the items in square bracket "[]". For example by running the code below, we can access the item at position 1 of vector saved as numbers.

```r
print(numbers[1])
```

```
## [1] 1
```

2. Matrices: They are two dimensional data set , generally consist of number. A matrix can be created with command *matrix()* and to specify number of row and columns use arguments "nrow" and "ncol". **for example**

```r
my_matrix<-matrix(c(1,2,3,4,5,6,7,8),nrow = 4,ncol = 2)
print(my_matrix)
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

```r
my_matrix2<-matrix(c(1,2,3,4,5,6,7,8),nrow = 2,ncol = 4)
print(my_matrix2)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

In the above example note the difference between *my_matrix* and *my_matrix2*.

One can also create a string matrix

3. Arrays
4. Factors

```r
color_names <- factor(c("red", "blue", "green", "red", "red", "blue", "red", "green"))

levels(color_names)
```

```
## [1] "blue"  "green" "red"
```

One of the advantage of keeping the data type as factor is ability to order the factor.

```
col_order<-ordered(color_names,levels=c("green","red","blue"))
print(col_order)
```

```
## [1] red   blue  green red   red   blue  red   green
## Levels: green < red < blue
```

You can see that in output green < red < blue. This will help in ploting graphs as now green will be ploted first and blue last.

5. Data Frames

```
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

print(Data_Frame)
```

```
##   Training Pulse Duration
## 1 Strength   100       60
## 2  Stamina   150       30
## 3    Other   120       45
```

```
## check the 1st Row of the data frame Data_Frame[row,col]
Data_Frame[1,]
```

```
##   Training Pulse Duration
## 1 Strength   100       60
```

```
## to see the first Col of data frame
Data_Frame[,1]
```

```
## [1] "Strength" "Stamina"  "Other"
```

6. List.

```
list1<-list(Data_Frame,color_names,my_matrix2)
list2<-list(car10,names)
print(list1)
```

```
## [[1]]
##   Training Pulse Duration
## 1 Strength   100       60
## 2  Stamina   150       30
## 3    Other   120       45
##
## [[2]]
## [1] red   blue  green red   red   blue  red   green
```

```
## Levels: blue green red
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

```
print(list2)
```

```
## [[1]]
## [1] "10"
##
## [[2]]
## [1] "chandan" "Pandey"  "Kumar"
```

```
list3<-c(list1,list2)
print(list3)
```

```
## [[1]]
##   Training Pulse Duration
## 1 Strength   100       60
## 2  Stamina   150       30
## 3    Other   120       45
##
## [[2]]
## [1] red   blue  green red   red   blue  red   green
## Levels: blue green red
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
##
## [[4]]
## [1] "10"
##
## [[5]]
## [1] "chandan" "Pandey"  "Kumar"
```