Basic of R Programming Language

R is a popular programming language used for statistical computing and graphical presentation. Its most common use is to analyze and visualize data.

Why Use R?

- It is a great resource for data analysis, data visualization, data science and machine learning
- It provides many statistical techniques (such as statistical tests, classification, clustering and data reduction)
- It is easy to draw graphs in R, like pie charts, histograms, box plot, scatter plot, etc...
- It works on different platforms (Windows, Mac, Linux)
- It is open-source and free
- It has a large community support
- It has many packages (libraries of functions) that can be used to solve different problems

R & RStudio Setup on Windows

Watch this Video By Click on it 👇

Get Started

```
# Start With "Hello World"
print("Hello World!")
```

```
[1] "Hello World!"
```

```
name <- "Janak Singh"
print(name)
```

```
[1] "Janak Singh"
```

```
add = "KTM"
```

```
cat("Hello Universe!😂")
```

```
Hello Universe!😂
```

```
cat(name)
```

```
Janak Singh
```

```
paste("Hello, Elon Musk!😊")
```

```
'Hello, Elon Musk!😊'
```

```
paste("Hello", "World", sep=",")
```

```
'Hello,World'
```

User Input

```
my_name <- readline(prompt = "What is Your Name?")
class(my_name)

age <- readline(prompt="What is Your age?")
my_age <- as.integer(age)
class(my_age)
```

```
What is Your Name?Janak SIngh DHami
'character'
What is Your age?20
'integer'
```

```
r <- 5
```

```
name <- readline()
```

```
Ram Shah
```

```
print(name)
```

```
[1] "Ram Shah"
```

```
n <- readline(prompt = "What is your Name?")
paste("Your name is", n)
```

```
What is your Name?Janak Singh dhami
'Your name is Janak Singh dhami'
```

Comments in R

Comments can be used to explain R code, and to make it more readable.

Comments starts with a #. When executing code, R will ignore anything that starts with #.

```
# This is Comment
print("How are you?") #This is also comment👌
```

```
[1] "How are you?"
```

```
# this is comment
print("Hello Guys")

#print(name) # My name
```

```
[1] "Hello Guys"
```

Variables in R

Variables in *R* are used to store data values. You can assign values to variables using the **<-** operator or the **=** operator.

Variable names must start with a letter and can include letters, numbers, and underscores.

Rules for Variable Naming:

- Must start with a letter.
- Cannot contain spaces or special characters (except _ and .).
- Case-sensitive (e.g., var and Var are different).

```
# name
# Name
# my_name
# my-name - X
# my addr = "KTM"


# case-sensitive
addr = "Nepal"
Addr = "India"

print(addr)
print(Addr)
```

```
[1] "Nepal"
[1] "India"
```

```
# Assigning values to variables

x <- 10         # Numeric value
y <- 20.5      # Decimal/float value
name <- "Alice" # Character value
isOk <- TRUE    # Logical value
com <- 3i+6   #Complex number

# Using variables
sum <- x + y

message <- paste("Name:", name, "| Sum:", sum, "| isOk:", isOk)

# Printing the result
print(message)
```

```
[1] "Name: Alice | Sum: 30.5 | isOk: TRUE"
```

- The ***paste()*** function concatenates strings.
- The ***print()*** function displays the output.

```
a <- 55L
b <- 55.5
hlo <- "Hii"
is <- TRUE
com <- 3i +7

class(a)
class(b)
class(hlo)
class(is)
class(com)
```

```
'integer'
'numeric'
'character'
'logical'
'complex'
```

```
# check variable type
class(x)
class(y)
class(com)
```

```
'numeric'
'numeric'
'complex'
```

```
x <- 1L # integer
y <- 2 # numeric

# convert from integer to numeric:
a <- as.numeric(x)

# convert from numeric to integer:
b <- as.integer(y)
```

```
p <- 5
class(p)
```

```
'numeric'
```

```
q = as.integer(p)
class(q)
```

```
'integer'
```

Multiple Variables

R allows you to assign the same value to multiple variables in one line:

```
addr1 <- aad2 <- "KTM"


print(addr1)
print(aad2)
```

```
[1] "KTM"
[1] "KTM"
```

```
# Assign the same value to multiple variables in one line
var1 <- var2 <- var3 <- "Orange"

# Print variable values
var1
var2
var3
```

```
'Orange'
'Orange'
'Orange'
```

Basic Data Types

Basic data types in R can be divided into the following types:

- **numeric** - (10.5, 55, 787)

> A numeric data type is the most common type in R, and contains any number with or without a decimal, like: 10.5, 55, 787

- **integer** - (1L, 55L, 100L, where the letter "L" declares this as an integer)

> > Integers are numeric data without decimals. This is used when you are certain that you will never create a variable that should contain decimals. To create an integer variable, you must use the letter L after the integer value

- **complex** - (9 + 3i, where "i" is the imaginary part)

> A complex number is written with an "i" as the imaginary part

- **character** (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
- **logical** (a.k.a. boolean) - (TRUE or FALSE)

> We can use the class() function to check the data type of a variable.

```
# numeric
x <- 10.5
class(x)

# integer
x <- 1000L
class(x)

# string/character
name <- "Janak"
class(name)

# complex
x <- 9i + 3
class(x)

# character/string
x <- "R is exciting"
class(x)

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

```
'numeric'
'integer'
'character'
'complex'
'character'
'logical'
```

```
Name <- "Ram"
class(Name)
```

```
'character'
```

```
y <- 5i +6
class(y)
```

```
'complex'
```

Math in R

In this section, we will discuss about how we use math by using arthmatic operators in R.

```
# add
5+6

# sub
55-6

# divide
58/4

# sq.
2**3

# mod
10%%3
```

```
11
49
14.5
8
1
```

```
3+5
```

```
8
```

```
55-5
```

```
50
```

```
10/5
```

```
2
```

```
2**3
```

```
8
```

```
3*4
```

```
12
```

```
x <- 55
y <- x**2

sum <- x + y

print(sum)
```

```
[1] 3080
```

Built-in Math Functions

```
# Max and Min
max(2, 5, 66, 4444)
min(2, 5, 66, 4444)

# Square root
sqrt(25)
```

```
4444
2
5
```

```
marks <- c(25,67, 78, 55)

first <- max(marks)
print(first)
```

```
[1] 78
```

```
fail <- min(marks)
print(fail)
```

```
[1] 25
```

```
a <- 44
sq <- sqrt(a)
```

```
print(sq)
```

```
[1] 6.63325
```

> **ceiling() and floor()**
>
> > The **ceiling()** function rounds a number upwards to its nearest integer, and the **floor()** function rounds a number downwards to its nearest integer, and returns the result:

```
ceiling(1.4)

floor(1.4)
```

```
2
1
```

```
# abs()
# The abs() function returns the absolute (positive) value of a number:
abs(-74.7)
```

```
74.7
```

```
t <- -56

abs(t)
```

```
56
```

String in R

```
fname <- "Janak" # First name of the user
lname <- "Dhami"  # last name of the user # Corrected the assignment to lnam
age <- 20 #age of the user
isHealthy = TRUE #health condition of user

msg = paste("My name is", fname, lname, "& i am", age, "Years old.")

print(msg)
```

```
[1] "My name is Janak Dhami & i am 20 Years old."
```

```
first_name <- "Janak"
second_name <- "Dhami"

full_name <- paste("My name is", first_name, second_name)

print(full_name)
```

```
[1] "My name is Janak Dhami"
```

```
nchar(full_name)
```

```
22
```

```
# String Length
nchar(msg)
```

```
43
```

```
grepl("Dhami", full_name)
```

```
TRUE
```

```
# Check a String
# Use the grepl() function to check if a character or a sequence of characte
grepl("Janak", msg)
```

```
TRUE
```

```
my_name <- "Janak S,ingh D,hami"

# Split the string into individual characters
ind <- strsplit(my_name, ",")
ind[1]
```

```
1. 'Janak S' · 'ingh D' · 'hami'
```

Escape Characters

To insert characters that are illegal in a string, you must use an escape character.

Code	Result
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace

```
intro <- "Nepal  \"isbeautiful\" country"
cat(intro)
```

```
Nepal  "isbeautiful" country
```

```
my_intro <- "My name is Janak. from DCL."
print(my_intro)
```

```
[1] "My name is Janak. from DCL."
```

```
my_intro <- "My \b name is \t Janak. \n from DCL."
cat(my_intro)
```

```
Myname is        Janak. 
 from DCL.
```

```
# Erro Example
intro <- "Nepal is beautiful country located in "Kathmandu""
```

```
Error in parse(text = input): <text>:2:50: unexpected symbol
1: # Erro Example
2: intro <- "Nepal is beautiful country located in "Kathmandu
                                                    ^
Traceback:
```

```
# Solution
intro <- "Nepal is beautiful country located in \"Kathmandu\". \n Mt. Everes
cat(intro)
```

```
Nepal is beautiful country located in "Kathmandu". 
 Mt. Everest is also located in Nepal.
```

R Booleans / Logical Values

When you compare two values, the expression is evaluated and R returns the logical answer

```
10 > 9    # TRUE because 10 is greater than 9
10 == 9   # FALSE because 10 is not equal to 9
10 < 9    # FALSE because 10 is greater than 9
```

```
TRUE
FALSE
FALSE
```

```
10 == 9
```

```
FALSE
```

```
"Janak" == "janak"
```

```
FALSE
```

```
# Example
a <- 10
b <- 9

a > b

# Note: We will use these conditions in if...else.. conditional statements a
```

```
TRUE
```

> ***Q&A: What is Statement?***

Operators in R

Operators are used to perform operations on variables and values.

R divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators

- Logical operators
- Miscellaneous operators

```
a <- 55
b <- 10
```

Arithmetic Operators:

Arithmetic operators are used with numeric values to perform common mathematical operations

```
# 1. addition
add <- a + b

# 2. Subtraction
sub <- a - b

# 3. Division
div <- a / b

# 4. Mode
mod <- b %% a

# 5. Exponent
ex <- 4^2

cat(add, sub, div, mod, ex)
```

```
65 45 5.5 10 16
```

```
a <- 4
b <- 6


sum <-  a + b
sub <- a - b
div <- a/b
```

Assignment Operators:

Assignment operators are used to assign values to variables

```
my_var <- 3

my_var <<- 3 # <<- is a global assigner.

3 -> my_var

3 ->> my_var
y = 55

my_var # print my_var
```

```
3
```

```
a <- 5

55 -> num


print(a)
print(num)
```

```
[1] 5
[1] 55
```

Comparison Operators:

Comparison operators are used to compare two values

```
# 1. Equal to
10 == 10

#  2. Not Equal to
5 != 6

# 3. Greater than
6 > 4

# 4. less than
4 < 56

# 5. Greater than or equal to
#  >=

# 6. Less than or equal to
#  <=
```

```
TRUE
TRUE
TRUE
TRUE
```

```
"Janak" != "janak"
```

```
TRUE
```

```
55>6
```

```
TRUE
```

Start coding or generate with AI.

Logical Operators:

Logical operators are used to combine conditional statements

```
# 1. & and && are the Logical AND Operator
p <- 5
q <- 6

p == 5 & q == 5

# 2. | and || are the locial OR Operator
```

```
p == 5 | q == 5

# 3. ! is the logical NOT Operator
!q == 5
!p == 5
```

```
FALSE
TRUE
TRUE
FALSE
```

```
p <- 8
a <- 5

p > 6 & a < 6
```

```
TRUE
```

```
 a == 5 | p > 10
```

```
TRUE
```

```
!a == 5
```

```
FALSE
```

Miscellaneous Operators:

1. Miscellaneous operators are used to manipulate data

```
# : Creates a series of numbers in a sequence   x <- 1:10
num <- 1:5
print(num)

# %in%  Find out if an element belongs to a vector  x %in% y
# wait for next topic

# %*%   Matrix Multiplication   x <- Matrix1 %*% Matrix2
# wait for next topic
```

```
[1] 1 2 3 4 5
```

Conditional Statements

Conditional statements let the program decide what to do based on whether something is true or false. They are used to make decisions and control what happens in your code.

if..else if ....else... Statement

Syntax

```
if (conditions){
   Positive Statements
   Positive Statements
}else if (conditions){
   Positiv statemests
}else{
   False Statements
}
```

```
a <- readline(prompt="Enter any number:")
```

```
Enter any number:67
```

```
if(a == 5){
  print("Ohh, It's 5.")
}


if(a == 6){
  print("Ohh, It's 6.")
}else{
  print("Ohh, it's not 6.")
}
```

```
[1] "Ohh, it's not 6."
```

```
a <- readline(prompt = "Enter your fav. Number:")
if(a == 5){
  print("Ohh, It's 5.")
}else if( a==6){
  print("It's 6.")
```

```r
}else if ( a == 0){
  print("It's Zero.")
}else{
  print("Sorry!")
}
```

```
Enter your fav. Number:45745754
[1] "Sorry!"
```

```r
height_of_janak <- 5.10
height_of_sita <- 5.2
height_of_ram <- 6


# if conditions
if (height_of_ram >= height_of_sita){
  cat("Height of Ram is greater than Sita.")
}
```

```
Height of Ram is greater than Sita.
```

```r
# if...else... condition
if (height_of_janak >= height_of_ram){
  cat("Height of Janak is greater than Ram.")
}else{
  cat("Height of Ram is greater than Sita.")
}
```

```
Height of Ram is greater than Sita.
```

```r
# if....else if.....else... condition

if(height_of_janak >= height_of_ram){
  cat("Height of Janak is greater than Ram.")
}else if(height_of_sita >= height_of_ram){
  cat("Height of Sita is Grater than Ram")
}else{
  cat("Height of Ram is Grater than Janak and Sita.")
}
```

```
Height of Ram is Grater than Janak and Sita.
```

```r
# another example
a <- 5
b <- 6
```