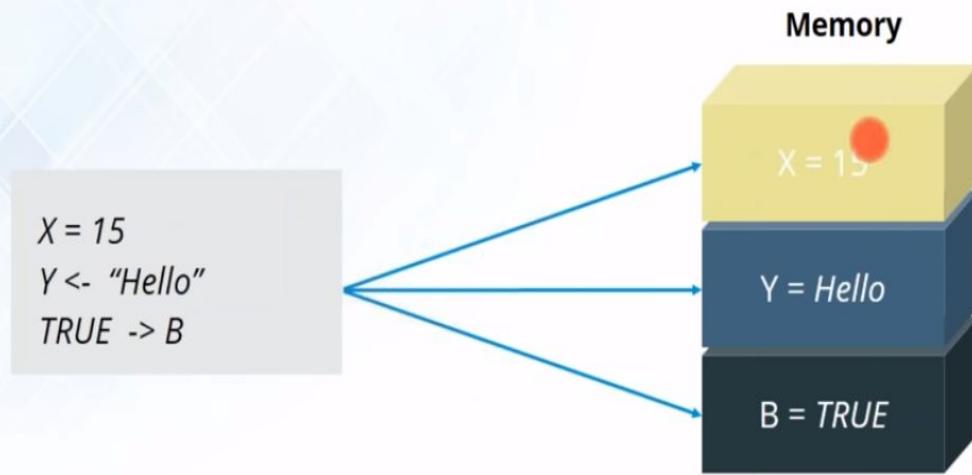


Variable :

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.



Variable Name	Validity	Reason
var_name2.	valid	Has letters, numbers, dot and underscore
var_name%	Invalid	Has the character '%'. Only dot(.) and underscore allowed.
2var_name	invalid	Starts with a number
.var_name, var.name	valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
.2var_name	invalid	The starting dot is followed by a number making it invalid.
_var_name	invalid	Starts with _ which is not valid

Deleting Variables

Variables can be deleted by using the `rm()` function. Below we delete the variable `var.3`. On printing the value of the variable error is thrown.

```
rm(var.3)
```

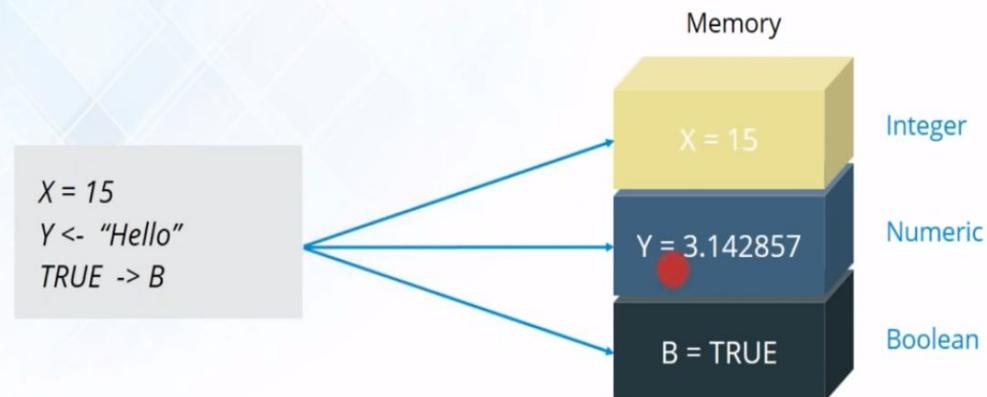
```
print(var.3)
```

When we execute the above code, it produces the following result –

```
[1] "var.3"  
Error in print(var.3) : object 'var.3' not found
```

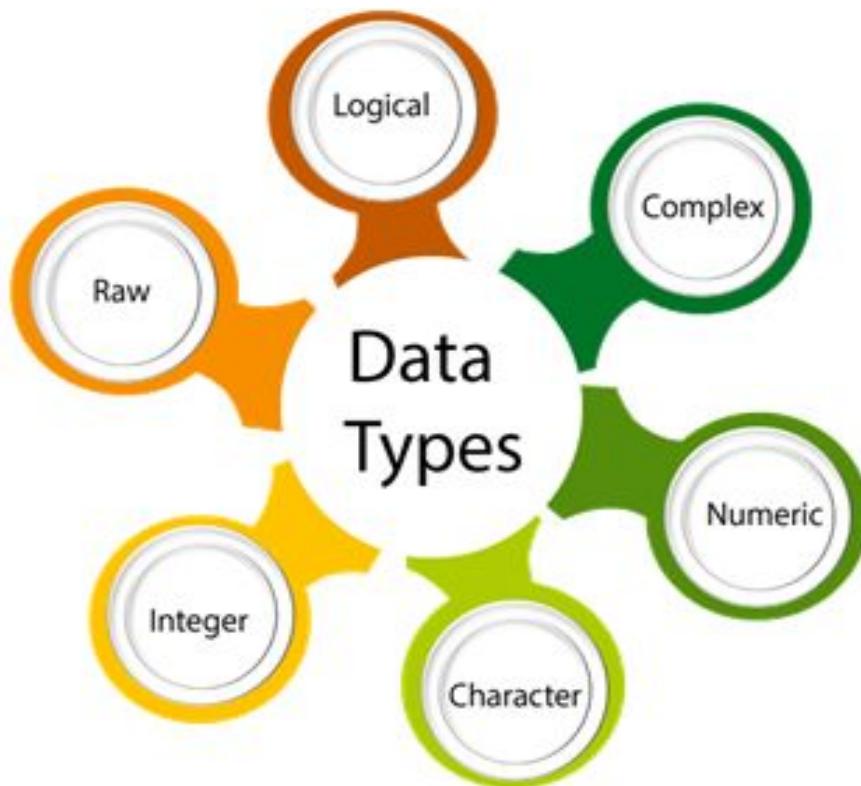
DataTypes in R-prog.

A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.



Data Types in R - Programming :

- R's basic data types are character, numeric, integer, complex, and logical, Raw.
- R's basic data structures include the vector, list, matrix, data frame, and factors, array.



1. **#Logical Data type**
2. **variable_logical<- TRUE**
3. **cat(variable_logical,"\\n")**
4. **cat("The data type of variable_logical is
,class(variable_logical),"\\n\\n")**
- 5.
6. **#Numeric Data type**
7. **variable_numeric<- 3532**
8. **cat(variable_numeric,"\\n")**

```
9. cat("The data type of variable_numeric is  
",class(variable_numeric),"\n\n")  
10.  
11. #Integer Data type  
12. variable_integer<- 133L  
13. cat(variable_integer,"\n")  
14. cat("The data type of variable_integer is  
",class(variable_integer),"\n\n")  
15.  
16. #Complex Data type  
17. variable_complex<- 3+2i  
18. cat(variable_complex,"\n")  
19. cat("The data type of variable_complex is  
",class(variable_complex),"\n\n")  
20.  
21. #Character Data type  
22. variable_char<- "Learning r programming"  
23. cat(variable_char,"\n")  
24. cat("The data type of variable_char is  
",class(variable_char),"\n\n")  
25.  
26. #Raw Data type  
27. variable_raw<- charToRaw("Learning r programming")  
28. cat(variable_raw,"\n")
```

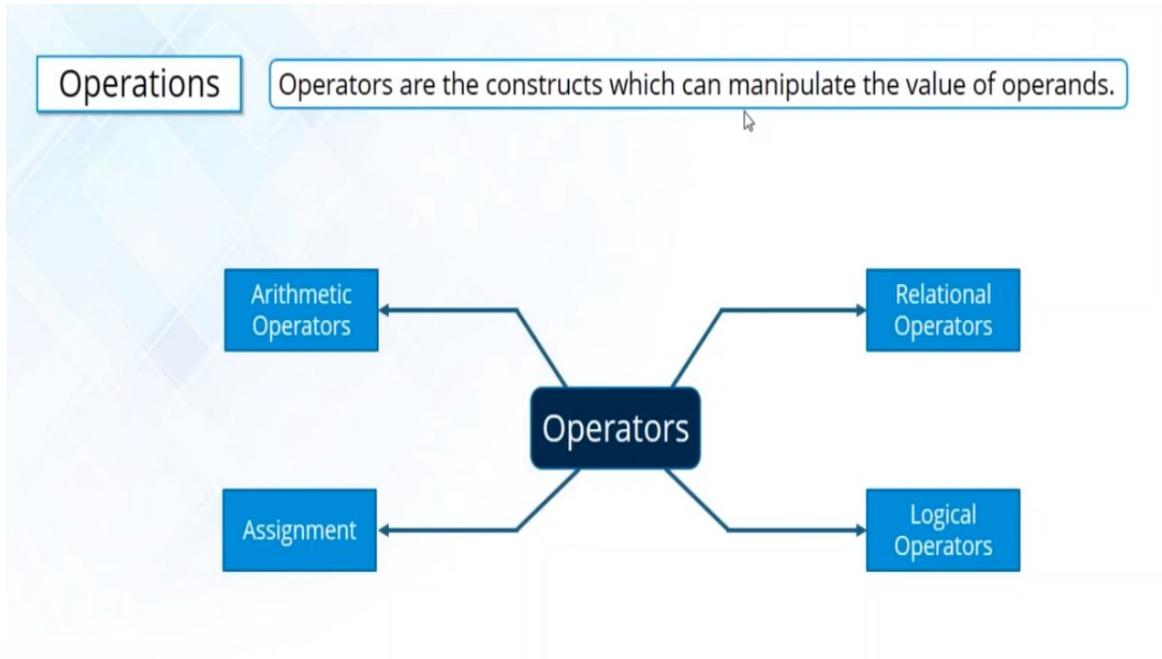
29. `cat("The data type of variable_char is
,class(variable_raw),"\n\n")`

When we execute the following program, it will give us the following output:

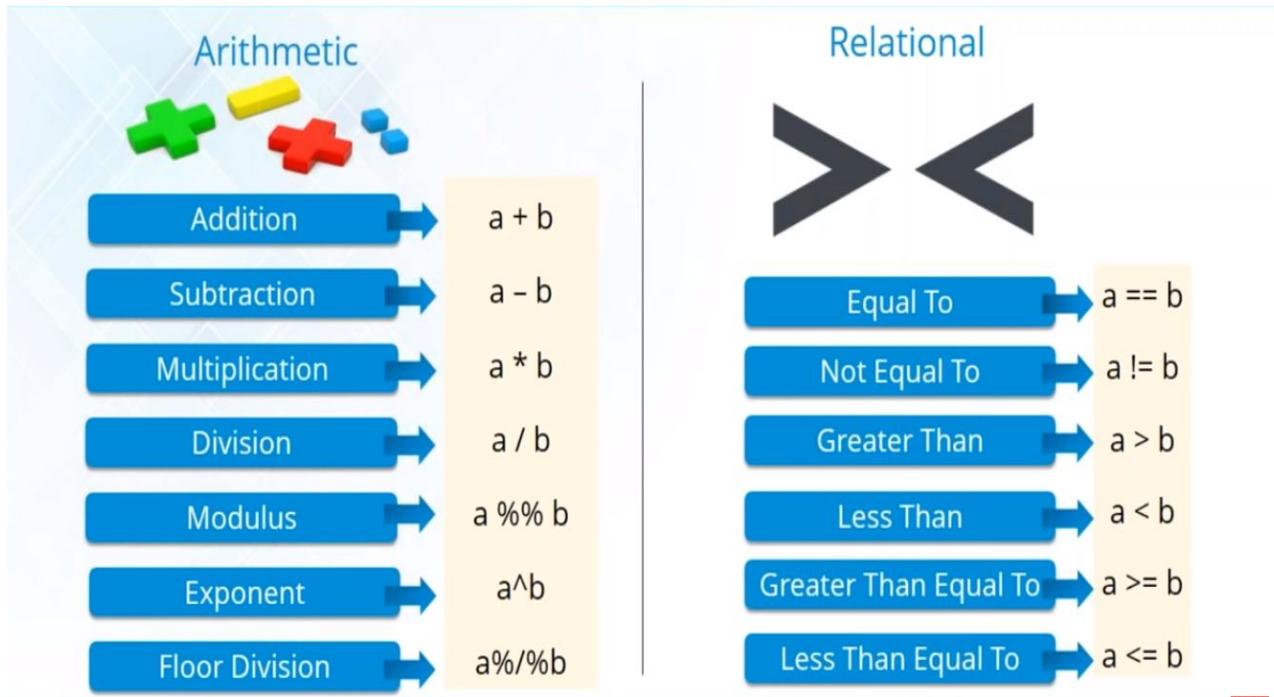
```
C:\Users\ajeet\R>Rscript datatype.R
TRUE
The data type of variable_logical is logical
3532
The data type of variable_numeric is numeric
133
The data type of variable_integer is integer
3+2i
The data type of variable_complex is complex
Learning r programming
The data type of variable_char is character
4c 65 61 72 6e 69 6e 67 20 72 20 70 72 6f 67 72 61 6d 6d 69 6e 67
The data type of variable_char is raw
```



Data Operators In R



Arithmetic and Relational Operators



```

1 ~ - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1* R data sets Source on Save Go to file/function Addins
1 var = 25
2 var1 = 60
3
4 print(var>var1)

```

Values

```

var      25
var1     60
vtr1    int [1:5] 1 2 3 4 5
vtr2    chr [1:5] "Neel" ...
vtr3    num [1:5] 15 25 65...

```

Console

```

[1] 1
> print(2^7)
[1] 128
> print(22%%7)
[1] 3
> print(3.9%%2)
[1] 1
> var = 25
> var1 = 60
> print(var>var1)
[1] FALSE

```

Assignment Operator

There are two types of Assignment operators: **Left and Right**



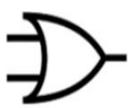
Logical Operator

There are three types of logical operators: **AND, NOT, OR**



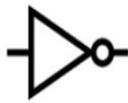
$a \& b$

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



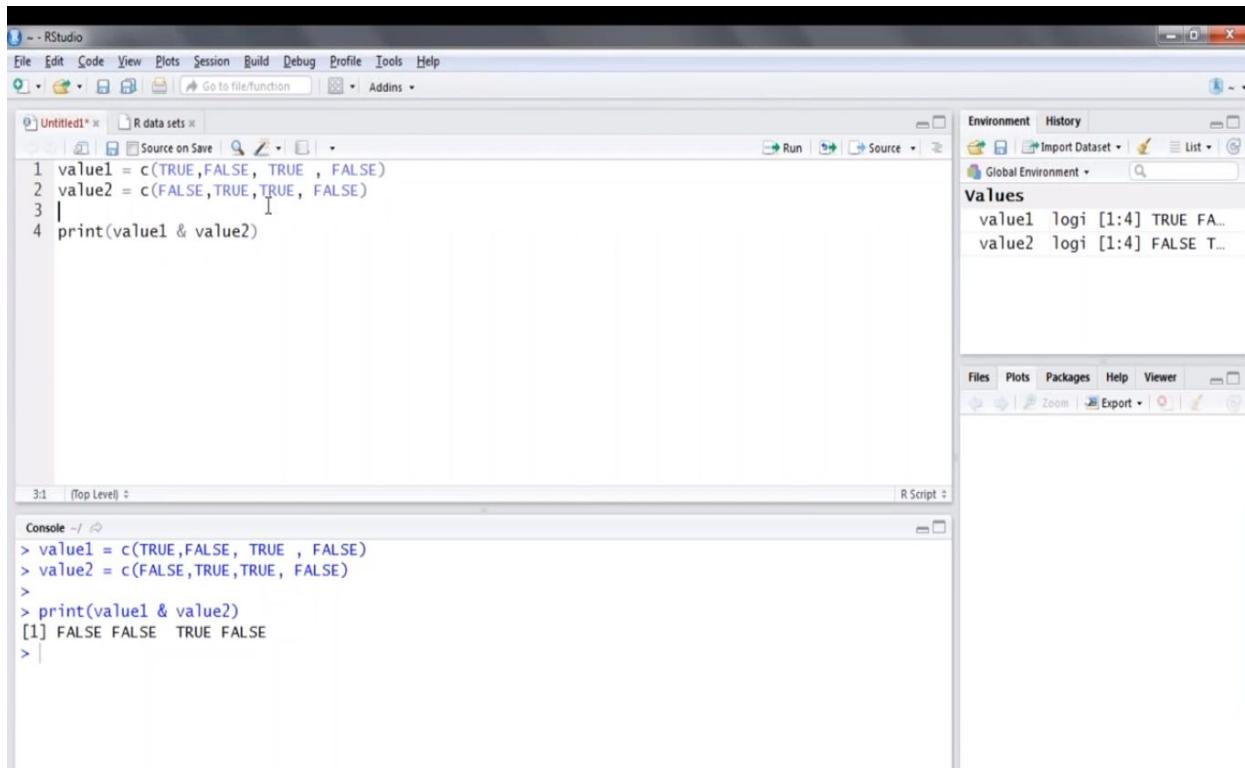
$a | b$

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



$!a$

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



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The left sidebar shows an 'Untitled1.R' script with the following code:

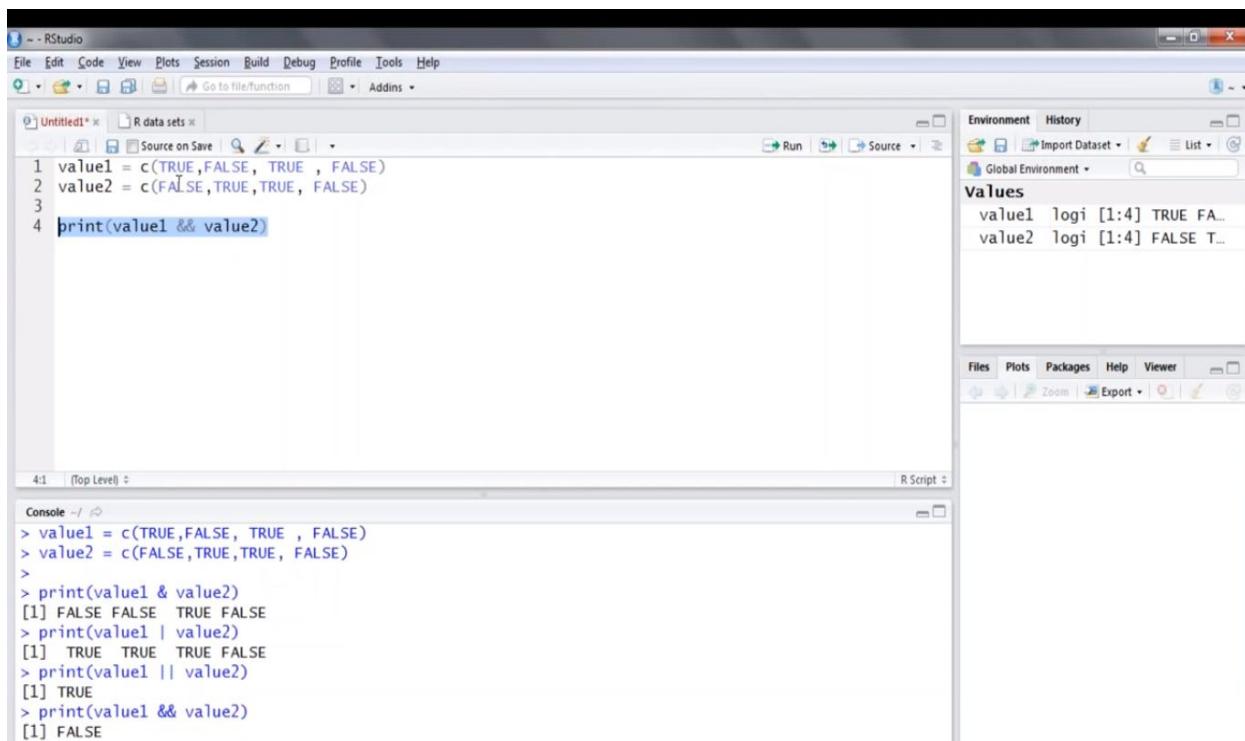
```
1 value1 = c(TRUE,FALSE, TRUE , FALSE)
2 value2 = c(FALSE,TRUE,TRUE , FALSE)
3 |
4 print(value1 & value2)
```

The right sidebar displays the 'Environment' and 'Values' panes. The 'Values' pane shows:

```
value1 logi [1:4] TRUE FA...
value2 logi [1:4] FALSE T...
```

The bottom panel is the 'Console' window, which also contains the same code and output:

```
> value1 = c(TRUE,FALSE, TRUE , FALSE)
> value2 = c(FALSE,TRUE,TRUE , FALSE)
>
> print(value1 & value2)
[1] FALSE FALSE TRUE FALSE
>
```



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The left sidebar shows an 'Untitled1.R' script with the following code:

```
1 value1 = c(TRUE,FALSE, TRUE , FALSE)
2 value2 = c(FALSE,TRUE,TRUE , FALSE)
3 |
4 print(value1 && value2)
```

The right sidebar displays the 'Environment' and 'Values' panes. The 'Values' pane shows:

```
value1 logi [1:4] TRUE FA...
value2 logi [1:4] FALSE T...
```

The bottom panel is the 'Console' window, which contains the following code and output:

```
> value1 = c(TRUE,FALSE, TRUE , FALSE)
> value2 = c(FALSE,TRUE,TRUE , FALSE)
>
> print(value1 & value2)
[1] FALSE FALSE TRUE FALSE
> print(value1 | value2)
[1] TRUE TRUE TRUE FALSE
> print(value1 || value2)
[1] TRUE
> print(value1 && value2)
[1] FALSE
```

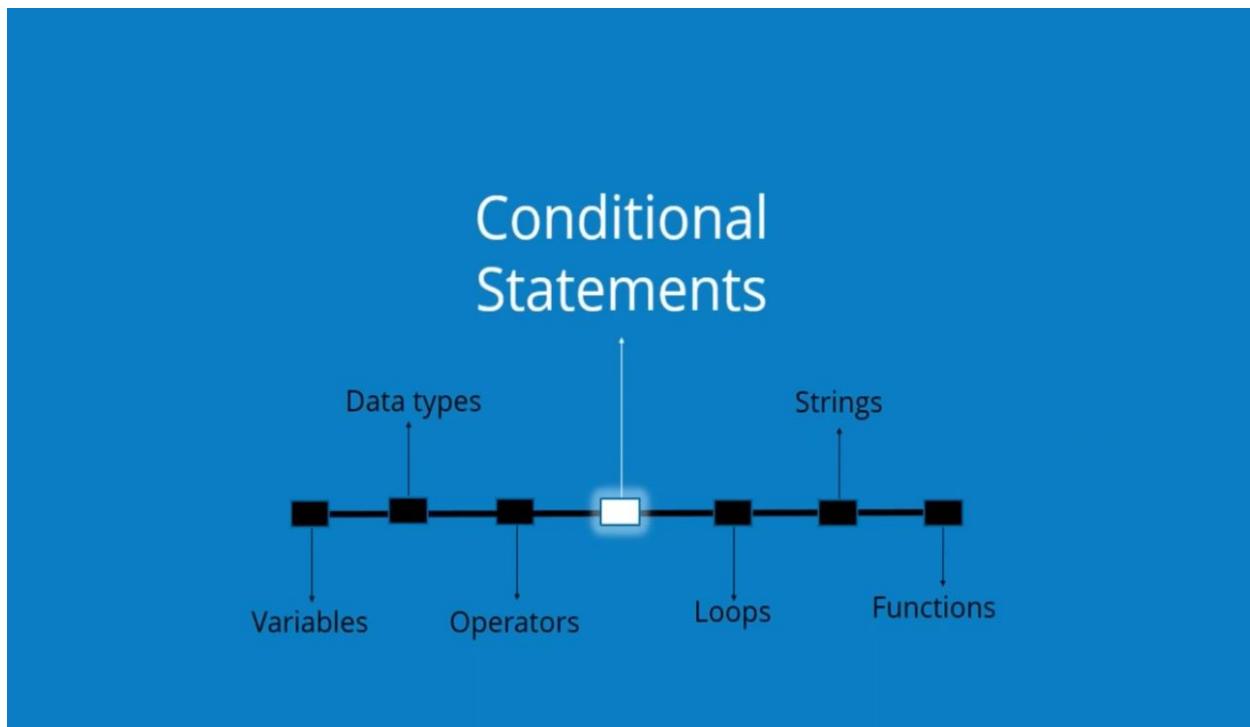
The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1.R' script with the following code:

```
1 value1 = c(TRUE, FALSE, TRUE, FALSE)
2 value2 = c(FALSE, TRUE, TRUE, FALSE)
3
4 print(!value1)
```

In the bottom-left pane, the 'Console' window shows the following session:

```
>
> print(value1 & value2)
[1] FALSE FALSE TRUE FALSE
> print(value1 | value2)
[1] TRUE TRUE TRUE FALSE
> print(value1 || value2)
[1] TRUE
> print(value1 && value2)
[1] FALSE
> print(!value1)
[1] FALSE TRUE FALSE TRUE
```

The right-hand side of the interface includes the Environment browser, which displays the values of 'value1' and 'value2' as logical vectors.

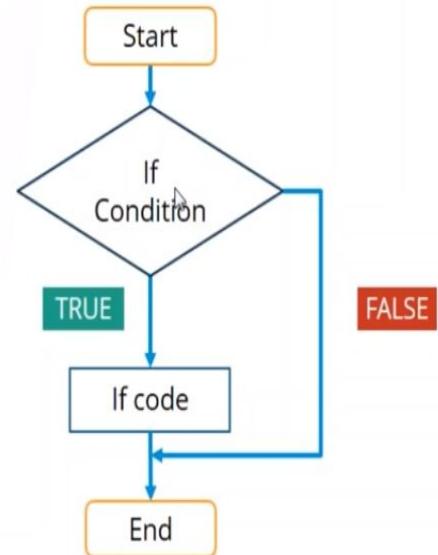


Conditional Statements

If Statement

Syntax:

```
1 if (expression)
2 {
3     //statements
4 }
```



The screenshot shows an RStudio interface with a script editor, console, and environment viewer.

Script Editor:

```
1 var1 = 25
2 var2 = 35
3
4 if ((var1+var2)>100){
5   print("Value is greater than 100")
6 }
```

Console:

```
> var1 = 25
> var2 = 35
>
> if ((var1+var2)>50){
+   print("Value is greater than 50")
+ }
[1] "Value is greater than 50"
> if ((var1+var2)>100){
+   print("Value is greater than 100")
+ }
> |
```

Environment Viewer:

Values	var1	25
Values	var2	35

Conditional Statements

if...else statement

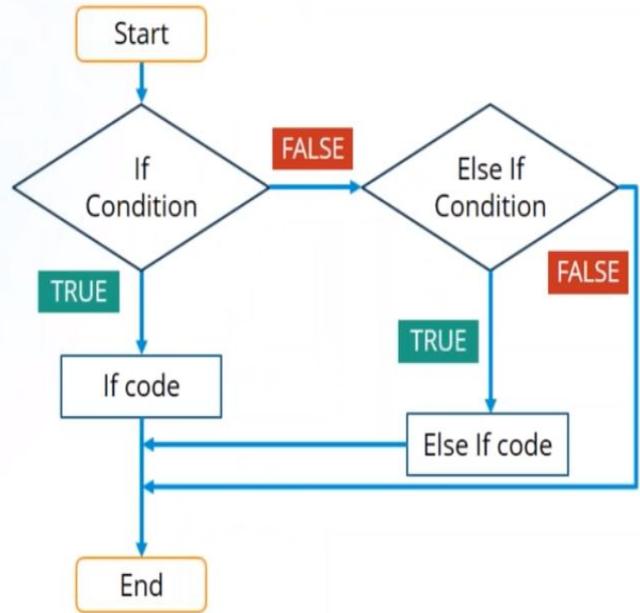
The syntax of if...else statement is:

```
if (test_expression) {  
statement1  
} else {  
statement2  
}
```

Else If Statement

Syntax:

```
1  if (expression 1)  
2  {  
3      //Statement 1  
4  }  
5  else if (expression 2)  
6  {  
7      //Statement 2  
8  }
```



The screenshot shows the RStudio interface with the following details:

- Code Editor:** An R script titled "Untitled1" containing the following code:

```
1 var1 = 25
2 var2 = 35
3
4- if ((var1+var2)>100){
5  print("Value is greater than 100")
6- } else if ((var1 + var2)>75){
7  print("Value is greater than 75")
8- }else if ((var1 + var2)>65){
9  print("Value is greater than 65")
10 }
```
- Console:** Displays the output of the script:

```
[1] "Value is greater than 50"
> if ((var1+var2)>100){
+   print("Value is greater than 100")
+
> if ((var1+var2)>100){
+   print("Value is greater than 100")
+ } else if ((var1 + var2)>75){
+   print("Value is greater than 75")
+ }else if ((var1 + var2)>65){
+   print("Value is greater than 65")
+ }
```
- Environment:** Shows variable values: var1 = 25 and var2 = 35.

Conditional Statements

if...else Ladder

The **if...else ladder (if...else...if)** statement allows you execute a block of code among more than 2 alternatives

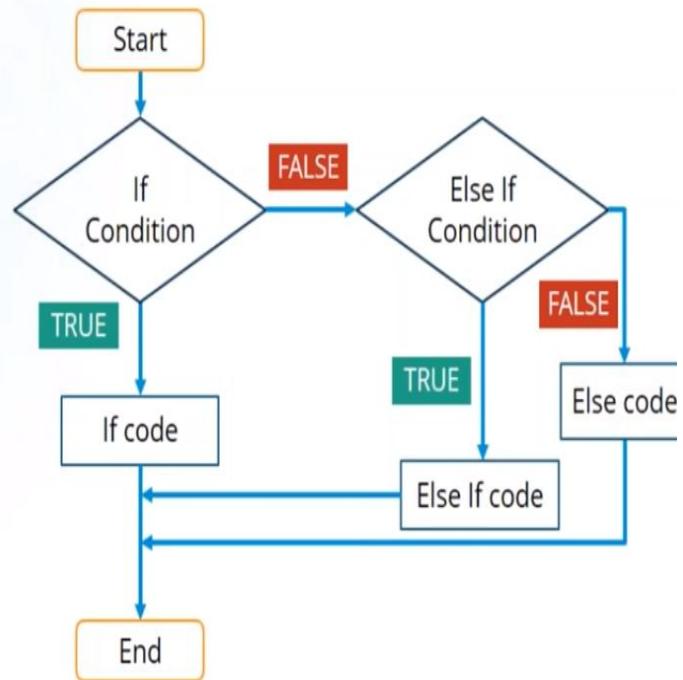
The syntax of if...else statement is:

```
if ( test_expression1) {
statement1
} else if ( test_expression2) {
statement2
} else if ( test_expression3) {
statement3
} else {
statement4
}
```

Else Statement

Syntax:

```
1 if (expression 1)
2 { //Statement 1
3 }
4 else if (expression 2)
5 { //Statement 2
6 }
7 else
8 { //Statement 3
9 }
```



The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1' script with the following R code:

```

1 var1 = 25
2 var2 = 35
3
4 if ((var1+var2)>100){
5   print("Value is greater than 100")
6 } else if ((var1 + var2)>75){
7   print("Value is greater than 75")
8 }else if ((var1 + var2)>65){
9   print("Value is greater than 65")
10 } else
11   print("Number is less than 65")

```

In the bottom-right pane, the 'Console' window shows the execution of the script. It starts with the message 'Value is greater than 75' and ends with '[1] "Number is less than 65"'. There is also an error message 'Error: unexpected '}' in ""'.

Switch Statement

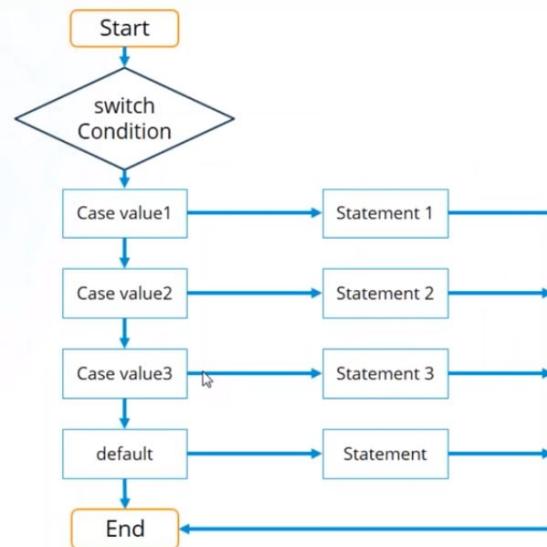
Switch case Statement

Syntax:

```

1 switch(Expression
2
3   case 1 = #Statement1
4   case 2 = #Statement2
5   case 3 = #Statement3
6
7   default Statement
8 )

```



The screenshot shows the RStudio interface with the following details:

- Code Editor:** An R script named "Untitled1.R" containing the following code:

```
1 switch(4,
2     '1'= print("Monday"),
3     '2'= print("Tuesday"),
4     '3'= print("Wednesday"),
5     '4'=print("Thursday"),
6     '5'=print("Friday"),
7     '6'=print("Saturday"),
8     '7'= print("Sunday")
9 )
10 
```
- Console:** The output of the script is shown in the console:

```
> switch(4,
+     '1'= print("Monday"),
+     '2'= print("Tuesday"),
+     '3'= print("Wednesday"),
+     '4'=print("Thursday"),
+     '5'=print("Friday"),
+     '6'=print("Saturday"),
+     '7'= print("Sunday")
+ )
[1] "Thursday"
> 
```
- Environment View:** Shows variables defined in the global environment:

Values	day	3
var1	25	
var2	35	

The screenshot shows the RStudio interface with the following details:

- Code Editor:** An R script named "Untitled1.R" containing the following code:

```
1 switch("%",
2     '1'= print("Monday"),
3     '2'= print("Tuesday"),
4     '3'= print("Wednesday"),
5     '4'=print("Thursday"),
6     '5'=print("Friday"),
7     '6'=print("Saturday"),
8     '7'= print("Sunday"),
9     print("Invalid input")
10 )
11 
```
- Console:** The output of the script is shown in the console:

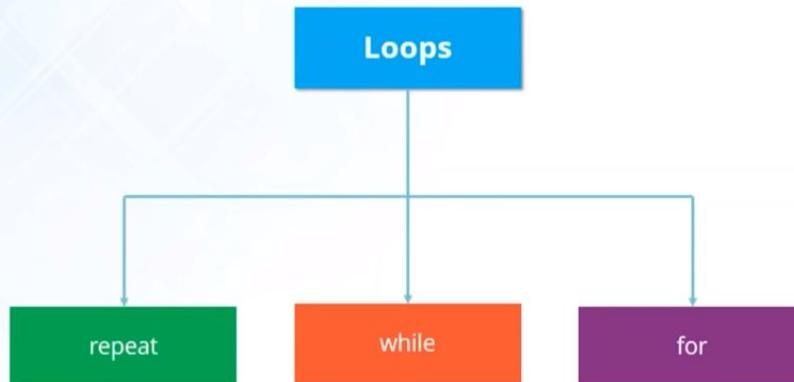
```
> switch("%",
+     '1'= print("Monday"),
+     '2'= print("Tuesday"),
+     '3'= print("Wednesday"),
+     '4'=print("Thursday"),
+     '5'=print("Friday"),
+     '6'=print("Saturday"),
+     '7'= print("Sunday"),
+     print("Invalid input")
+ )
[1] "Invalid input"
> 
```
- Environment View:** Shows variables defined in the global environment:

Values	day	3
var1	25	
var2	35	

Loops In R

Loops

A loop statement allows us to execute a statement or group of statements multiple times.



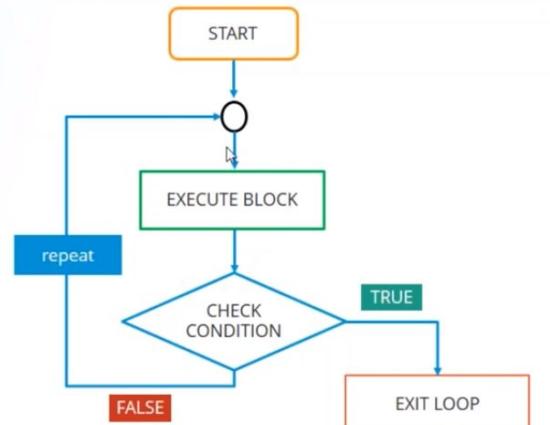
Repeat Loop

repeat

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition after executing the loop body.

Syntax:

```
1 repeat {  
2   commands  
3   if(condition) {  
4     break  
5   }  
6 }
```



The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays the R script code:

```

1 var1 = 5
2 repeat{
3   print(var1)
4   var1 = var1 +2
5   if ( var1 > 21){
6     break
7   }
8 }
```
- Environment View:** Shows the variable `var1` with the value `23`.
- Console View:** Displays the output of the script, which is a sequence of odd numbers from 5 to 21.

While loop

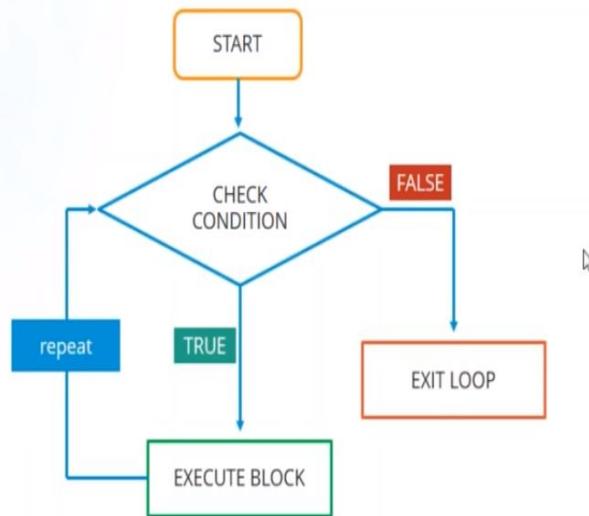
while Loop

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

Syntax:

```

1 while (condition)
2 {
3   //Statements
4 }
5 
```



The screenshot shows an RStudio interface with the following components:

- Code Editor:** Displays the following R code:

```
1 var1 = 5
2 repeat{
3   print(var1)
4   var1 = var1 +2
5   if ( var1 > 21){
6     break
7   }
8 }
9
10 var1 = 5
11 while ( var1 < 21){
12   print(var1)
13   var1 = var1 +2
14 }
```

- Console:** Shows the output of the code execution:

```
[1] 5
[1] 7
[1] 9
[1] 11
[1] 13
[1] 15
[1] 17
[1] 19
```

- Environment:** Shows the value of `var1`:

Values
var1 21

The screenshot shows an RStudio interface with the following components:

- Code Editor:** Displays the same R code as the first screenshot, but with a modification in the second loop:

```
1 var1 = 22
2 repeat{
3   print(var1)
4   var1 = var1 +2
5   if ( var1 > 21){
6     break
7   }
8 }
9
10 var1 = 5
11 while ( var1 < 21){
12   print(var1)
13   var1 = var1 +2
14 }
```

- Console:** Shows the output of the code execution:

```
[1] 17
[1] 19
> var1 = 22
> repeat{
+   print(var1)
+   var1 = var1 +2
+   if ( var1 > 21){
+     break
+   }
+ }
[1] 22
```

- Environment:** Shows the value of `var1`:

Values
var1 24

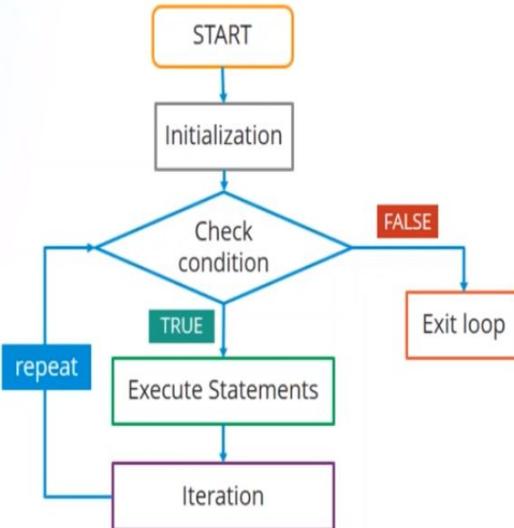
For Loop

for Loop

Repeats a statement or group of statements a fixed number of times. It tests the condition before executing the loop body.

Syntax:

```
1 for (value in vector) {  
2     statements  
3 }  
4 }
```



A screenshot of the RStudio interface. The code editor shows the following R script:

```
3 print(var1)
4 var1 = var1 +2
5 if ( var1 > 21){
6   break
7 }
8 }
10 var1 = 5
11 while ( var1 < 21){
12   print(var1)
13   var1 = var1 +2
14 }
15
16 for(x in 1:25){
17   print(x)
18 }
```

The console window below displays the output of the script:

```
[1] 15
[1] 16
[1] 17
[1] 18
[1] 19
[1] 20
[1] 21
[1] 22
[1] 23
[1] 24
[1] 25
```

The Environment tab shows the global environment with variables:

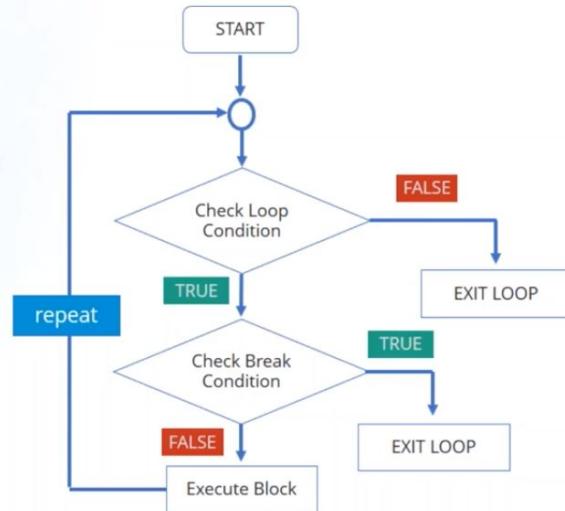
Values	var1	24
x	25L	

Flow Control

edureka!

7. break

Syntax:
break;

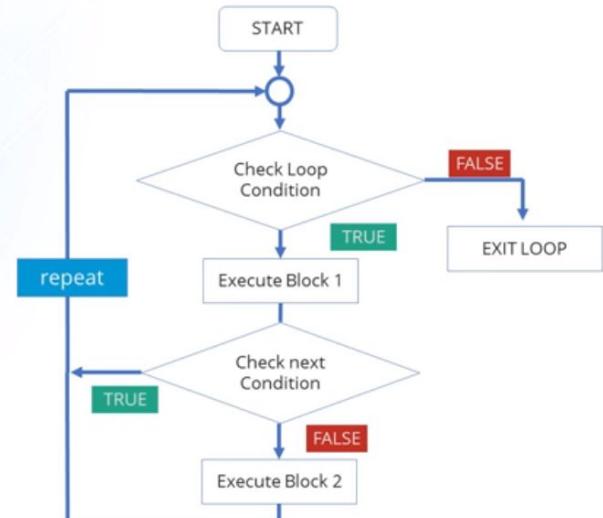


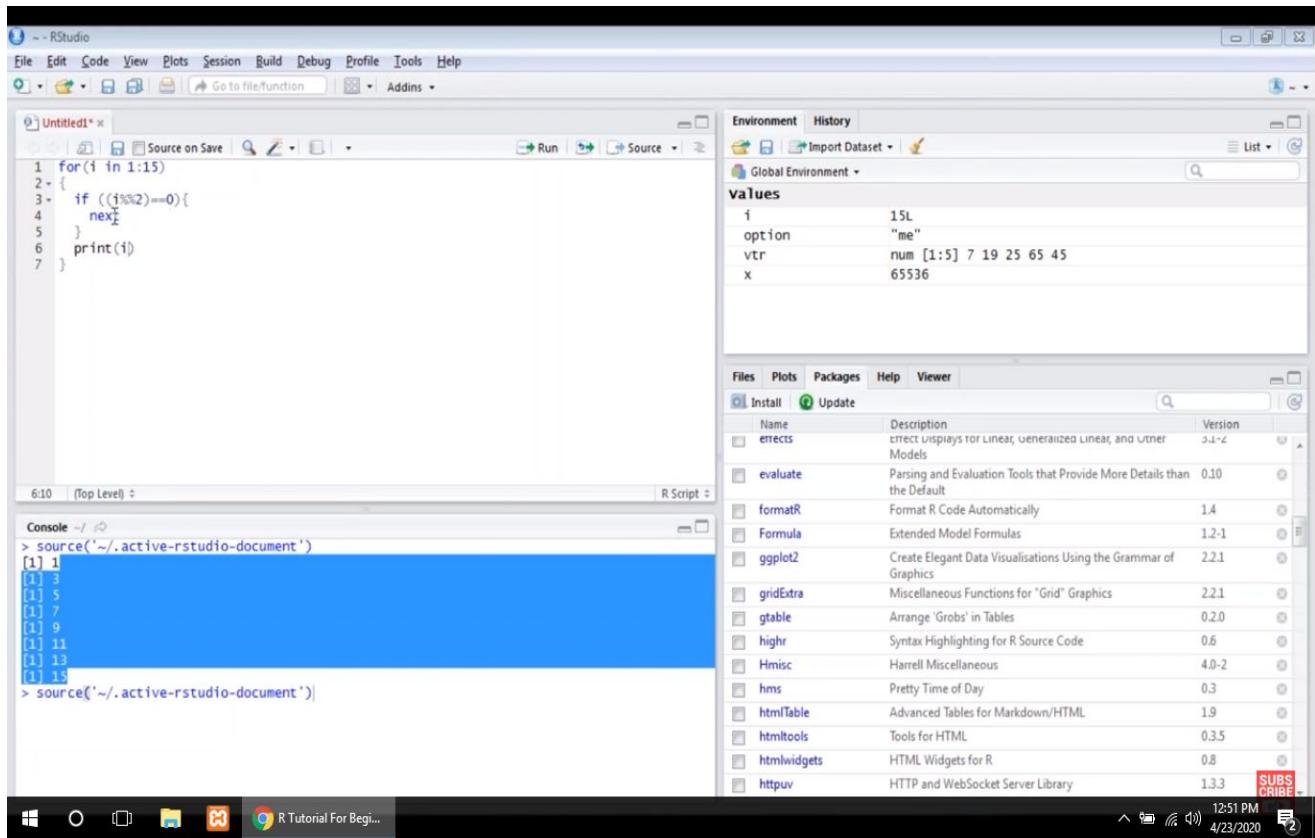
Flow Control

edureka!

8. next

Syntax:
next;





String in R

edureka!

Any value written within a pair of single quote or double quotes in R is treated as a string.

Syntax:

```

1 a <- 'Start and end with single quote'
2 print(a)
3
4 or
5
6 b <- "Start and end with double quotes"
7 print(b)

```

```

> print(a)
[1] "Start and end with single quote"
> print(b)
[1] "Start and end with double quotes"
>

```

EDUREKA DATA ANALYTICS WITH R CERTIFICATION TRAINING

https://www.edureka.co/ 12:54 PM 4/23/2020

The screenshot shows the RStudio interface. In the code editor (Top Level), the following R code is written:

```

1 str1 <- 'Hey, R is fun'
2 print(str1)
3 str2 <- "Hey, R is fun"
4 print(str2)
5

```

In the Console, the output of the code is displayed:

```

> str1 <- 'Hey, R is fun'
> print(str1)
[1] "Hey, R is fun"
> str2 <- "Hey, R is fun"
> print(str2)
[1] "Hey, R is fun"
>

```

The Environment pane shows the following variables:

Values	str1	"Hey, R is fun"
	str2	"Hey, R is fun"
	var1	24
	x	25L

The status bar at the bottom indicates the time as 12:54 PM and the date as 4/23/2020.

String in R

edureka!

Sequence Operations:

- Any value written within a pair of single quote or double quotes in R is treated as a string.

Concatenation:

```

str1<- 'Hi'
str2<- "How are you?"
paste(str1,str2)
"Hi How are you?"

```

Syntax:

Character Count:

```

1 a <- 'StartSand-endwith"single quote'
2 print(a)
3
4 or
5 b <- "Start and end with double quotes"
6 print(b)
7 str<"Edureka"

```

Case Change

```

> nchar(str)
[1] 10
> print(a)
[1] "Start and end with single quote"
> print(b)
[1] "Start and end with double quotes"

```

```

> toupper(str)
[1] "EDUREKA"
> tolower(str)
[1] "e lureka"

```

Substring

```

str<"Edureka"
substring(str, 3, 6)
"urek"

```

edureka! DATA ANALYTICS WITH R CERTIFICATION TRAINING

https://www.edureka.co/ 12:55 PM 4/23/2020

String in R

edureka!

Sequence Operations:

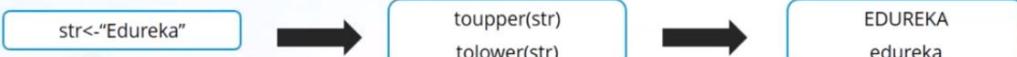
➤ Concatenation:



➤ Character Count:



➤ Case Change



➤ Substring



A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and Addins. The left pane shows an R script named "Untitled1.R" with the following code:

```
1 str1 <- 'Hey, R is fun'
2 print(str1)
3 str2 <- "Hey, R is fun"
4 print(str2)
5
6 str3 <- paste(str1,str2)
7
8 print(str3)
```

The right pane shows the "Environment" tab with variables defined:

Values	str1	"Hey, R is fun"
str2	"Hey, R is fun"	
str3	"Hey, R is fun Hey, R is fun"	
var1	24	
x	25L	

The bottom pane is the "Console" tab, showing the execution of the R script:

```
> str1 <- 'Hey, R is fun'
> print(str1)
[1] "Hey, R is fun"
> str2 <- "Hey, R is fun"
> print(str2)
[1] "Hey, R is fun"
> str3 <- paste(str1,str2)
>
> print(str3)
[1] "Hey, R is fun Hey, R is fun"
```

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main window shows an R script named 'Untitled1' with the following code:

```
1 str1 <- 'Hey, R is fun'
2 print(str1)
3 str2 <- "Hey, R is fun"
4 print(str2)
5
6 str3 <- paste(str1,str2)
7
8 print(str3)
9
10 nchar(str3)
```

The console window below shows the execution of the script:

```
> print(str1)
[1] "Hey, R is fun"
> str2 <- "Hey, R is fun"
> print(str2)
[1] "Hey, R is fun"
> str3 <- paste(str1,str2)
>
> print(str3)
[1] "Hey, R is fun Hey, R is fun"
> nchar(str3)
[1] 27
```

The right pane displays the 'Values' environment, showing:

Value	Content
str1	"Hey, R is fun"
str2	"Hey, R is fun"
str3	"Hey, R is fun Hey, R is fun"
var1	24
x	25L

The status bar at the bottom indicates 12:57 PM on 4/23/2020.

A screenshot of the RStudio interface, similar to the first one but with a 'SUBSCRIBE' button in the bottom right corner. The top menu bar and script content are identical to the first screenshot. The console output is also identical:

```
> print(str1)
[1] "Hey, R is fun"
> str2 <- "Hey, R is fun"
> print(str2)
[1] "Hey, R is fun"
>
> str3 <- paste(str1,str2)
>
> print(str3)
[1] "Hey, R is fun Hey, R is fun"
> nchar(str3)
[1] 27
```

The right pane shows the 'Values' environment with the same entries as the first screenshot.

The status bar at the bottom indicates 12:58 PM on 4/23/2020.

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1.R' script with the following R code:

```

4 print(str2)
5
6 str3 <- paste(str1,str2)
7
8 print(str3)
9
10 nchar(str3)
11
12 str4 = toupper(str1)
13 str5 = tolower(str2)
14
15 print(str4)
16 print(str5)
17
18 str6 = substr(str3,5,16)
19 print(str6)

```

In the bottom-left pane, the 'Console' window shows the results of running this code:

```

> str4 = toupper(str1)
> str5 = tolower(str2)
>
> print(str4)
[1] "HEY, R IS FUN"
> print(str5)
[1] "hey, r is fun"
> str6 = substr(str3,5,16)
> print(str6)
[1] "R is fun He"
> |

```

To the right of the console is the 'Environment' tab of the global environment pane, displaying variables and their values:

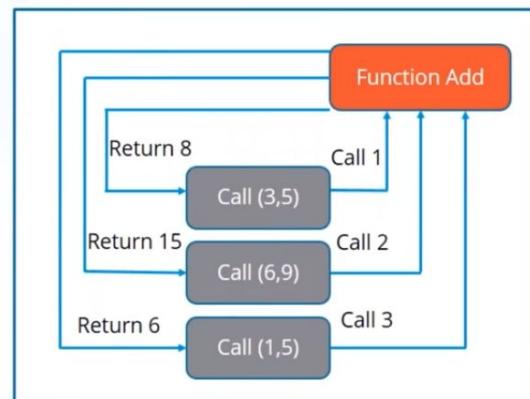
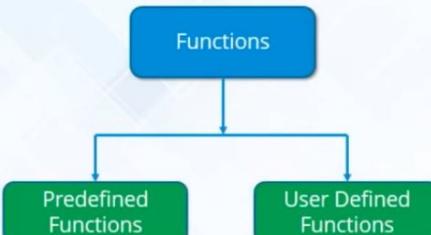
Values	str1	"Hey, R is fun"
	str2	"Hey, R is fun"
	str3	"Hey, R is fun Hey..."
	str4	"HEY, R IS FUN"
	str5	"hey, r is fun"
	str6	" R is fun He"

Functions In R

edureka!

Functions

A **function** is a block of organized, reusable code that is used to perform a single, related action.



```

1 fibo <- function(a){
2   var1 = 0
3   var2 = 1
4   print(var1)
5   print(var2)
6   for(x in 1:a){
7     var3 = var1 + var2
8     print(var3)
9     var1 = var2
10    var2 = var3
11  }
12}
13
14 fibo(5)
15 fibo(10)
16 fibo(13)

```

(Top Level) :: R Script

```

> fibo(5)
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
>

```

Console -/ →

1:02 PM 4/23/2020

Vector

[edureka!](#)

01 Vector

- 02 Matrix
- 03 Array
- 04 List
- 05 Data Frame

- ❑ A **Vector** is a sequence of data elements of the same basic type
- ❑ There are **5 Atomic vectors**, also termed as five classes of vectors.

Logical → True or False

Integer → 15L, 30L, 1699L

Numeric → 5, 3.14285, 945678

Complex → 4+3i, 8+7i

Character → 'A', "Hey", 'True'



R Programming For Beginners | R Language Tutorial | R Tutorial For Beginners | Edureka

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1* R data sets Source on Save Run Source

```
1 #Vectors
2
3 #Logical
4
5 vtr1 = c(TRUE, FALSE)
6 vtr2 = c(15, 85.674954,999999)
7 class(vtr1)
8
9 vt
```

Environment History Import Dataset List Global Environment Values

vtr1 logi [1:2] TRUE FA...
vtr2 num [1:3] 1.50e+01...

Files Plots Packages Help Viewer

Console / Help.start() for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

```
> vtr1 = c(TRUE, FALSE)
> class(vtr1)
[1] "logical"
> vtr2 = c(15, 85.674954,999999)
> vtr2
[1] 1.500000e+01 8.567495e+01 9.999999e+06
```

11:11 / 1:10:55

R ~ - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1* R data sets Source on Save Run Source

```
1 #Vectors
2
3 #Logical
4
5 vtr1 = c(TRUE, FALSE)
6 vtr2 = c(15, 85.674954,999999)
7 class(vtr1)
8
9 vt
```

Environment History Import Dataset List Global Environment Values

vtr1 logi [1:2] TRUE FA...
vtr2 num [1:3] 15 85.7 ...

Files Plots Packages Help Viewer

Console / Help.start() for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

```
> vtr1 = c(TRUE, FALSE)
> class(vtr1)
[1] "logical"
> vtr2 = c(15, 85.674954,999999)
> vtr2
[1] 1.500000e+01 8.567495e+01 9.999999e+06
> vtr2 = c(15, 85.674954,999999)
> vtr2
[1] 15.00000 85.67495 99999.00000
> |
```

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1' script file with the following R code:

```
1 #Vectors
2
3 #Logical
4
5 vtr1 = c(TRUE, FALSE)
6
7 #Numeric
8 vtr2 = c(15, 85.674954, 99999)
9
10 #Integer
11 vtr3 = c(35L, 58L, 146L)
12 vtr4 = c(58.465L)
13
```

In the bottom-left pane, the 'Console' window shows the execution of the code:

```
> vtr2 = c(15, 85.674954, 99999)
> vtr2
[1] 15.00000 85.67495 99999.00000
> vtr3 = c(35L, 58L, 146L)
> vtr3
[1] 35 58 146
> class(vtr3)
[1] "integer"
> vtr4 = c(58.465L)
Warning message:
integer literal 58.465L contains decimal; using numeric value
>
```

The right side of the interface includes the 'Environment' and 'History' panes, and the 'Values' pane which lists variables and their types and values:

Variables	Type	Value
vtr1	logi	[1:2] TRUE FALSE
vtr2	num	[1:3] 15 85.7 ...
vtr3	int	[1:3] 35 58 146
vtr4	num	58.465

This screenshot is identical to the one above, showing the same RStudio interface, script content, and console output. The only difference is the timestamp in the console window, which has changed from '12:1' to '15:30'.

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1' script with the following R code:

```

1 #Vectors
2
3 #Logical
4
5 vtr1 = c(TRUE, FALSE)
6
7 #Numeric
8 vtr2 = c(15, 85.674954, 99999)
9
10 #Integer
11 vtr3 = c(35L, 58L, 146L)
12 vtr4 = c(58.465L)
13
14 vtr5 = c(TRUE, 35L, 3.14)
15 vtr6 = c("Hello", FALSE, 65L)

```

In the bottom-left pane, the 'Console' window shows the results of running the code:

```

integer literal 58.465L contains decimal; using numeric value
> vtr5 = c(TRUE, 35L, 3.14)
> vtr5
[1] 1.00 35.00 3.14
> vtr6 = c("Hello", FALSE, 65L)
> vtr6
[1] "Hello" "FALSE" "65"
> class(vtr5)
[1] "numeric"
> class(vtr6)
[1] "character"
>

```

The top-right pane shows the 'Environment' tab with variables defined:

Values	vtr1	logi [1:2] TRUE FA...
	vtr2	num [1:3] 15 85.7 ...
	vtr3	int [1:3] 35 58 146
	vtr4	58.465
	vtr5	num [1:3] 1 35 3.14
	vtr6	chr [1:3] "Hello" ...

The bottom status bar shows the time as 12:12 PM and the date as 4/23/2020.

Matrix

edureka!

01 Vector

02 Matrix

Syntax

`matrix(data, nrow, ncol, byrow, dimnames)`

03 Array

04 List

05 Data Frame

Matrix are the R objects in which the elements are arranged in a two-dimensional rectangular layout.

- ❖ **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.

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1' script with the following code:

```

1 mtr = matrix(c(5:29),5,5)

```

In the bottom-left pane, the 'Console' window displays the following output:

```

Warning message:
In matrix(c(5:30), 5, 5) :
  data length [26] is not a sub-multiple or multiple of the number of rows [5]
> mtr = matrix(c(5:29),5,5)
> mtr
[,1] [,2] [,3] [,4] [,5]
[1,]    5   10   15   20   25
[2,]    6   11   16   21   26
[3,]    7   12   17   22   27
[4,]    8   13   18   23   28
[5,]    9   14   19   24   29
>

```

The right side of the interface shows the 'Environment' tab with variables defined:

- Data**: mtr int [1:5, 1:5] 5...
- Values**: vtr1 logi [1:2] TRUE FA...
vtr2 num [1:3] 15 85.7 ...
vtr3 int [1:3] 35 58 146
vtr4 58.465

The status bar at the bottom indicates the time as 12:15 PM and the date as 4/23/2020.

The slide is titled "R Programming For Beginners | R Language Tutorial | R Tutorial For Beginners | Edureka" and features the "edureka!" logo.

A vertical list on the left side shows the following items:

- 01 Vector
- 02 Matrix
- 03 Array
- 04 List
- 05 Data Frame

In the center, there is a box containing the text: "Arrays are the R data objects which can store data in more than two dimensions."

Below this, a green box labeled "Syntax" has an arrow pointing to the text: `array(data, dim, dimnames)`.

Further down, another box contains the code: `array(c(0:15), dim=c(4,4,2,2))`, with an arrow pointing to a visual representation of a 4x4x2x2 array.

The visual representation shows four 4x4 matrices, each labeled with indices 0, 1, 2, 3 along the top and left axes. The values in the matrices are as follows:

	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

The status bar at the bottom indicates the time as 12:15 PM and the date as 4/23/2020.