LOVELY PROFESSIONAL UNIVERSITY

CAP-788
DATA SCIENCE TOOLBOX LAB (CA-1)

SUBMITTED TO:

Dr. APASH ROY

(23550)

SUBMITTED BY:

YASH AGARWAL

MCA-DE566

RDE566A03

12000323

LOVELY PROFESSIONAL UNIVERSITY

SET-A

Make a tutorial presentation/ documentation on R programming Basics-Introduction, Syntax, Comments, Variable, Data Type, Number, math, String, Logical/Boolean. Write programming example for each of them. (Use Screenshots of full screen for each step, so that it can be differentiated from other person) .

LOVELY PROFESSIONAL UNIVERSITY

INTRODUCTION:

- R is a language and environment for statistical computing and graphics.
- It is a GNU project which was developed at Bell by John Chambers and colleagues.
- R can be considered as a different implementation of S.
- R offers a huge type of statistical and graphical techniques, and is exceptionally extensible.

SYNTAX:

- To output text in R, use single or double quc

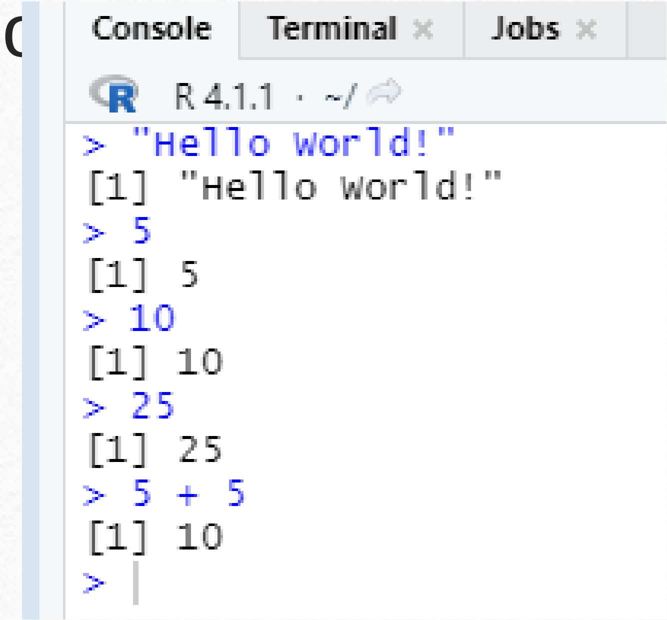
"Hello World!"

5

10

25

5 + 5

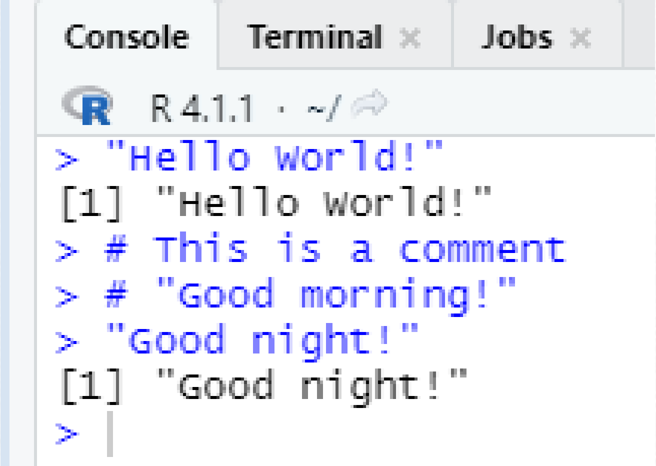


```
> "Hello World!"
[1] "Hello World!"
> 5
[1] 5
> 10
[1] 10
> 25
[1] 25
> 5 + 5
[1] 10
> 
```

COMMENTS:

- Comments can be used to explain R code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

```
"Hello World!" # This is a comment

# "Good morning!"
"Good night!"
```

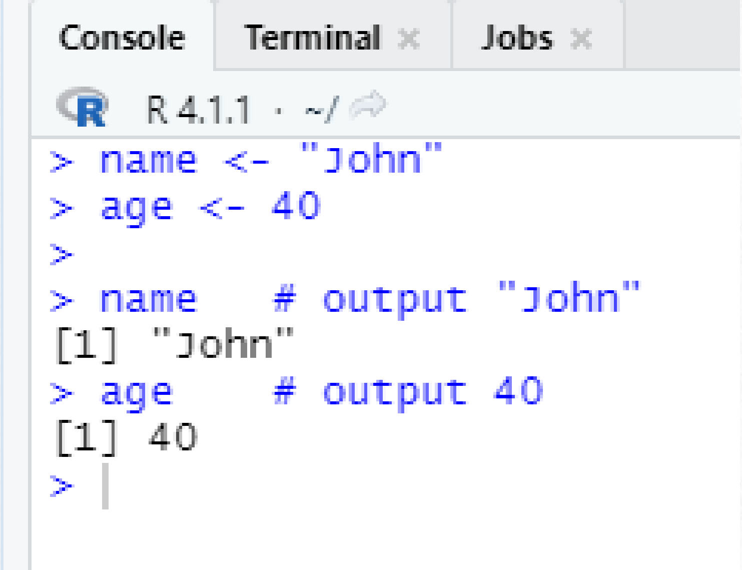


```
> "Hello World!"
[1] "Hello World!"
> # This is a comment
> # "Good morning!"
> "Good night!"
[1] "Good night!"
> 
```

VARIABLES:

- Variables are containers for storing data values.

```
name <- "John"
age <- 40

name   # output "John"
age    # output 40
```



```
> name <- "John"
> age <- 40
>
> name    # output "John"
[1] "John"
> age     # output 40
[1] 40
>
```

LOVELY
PROFESSIONAL
UNIVERSITY

DATA TYPE’s:

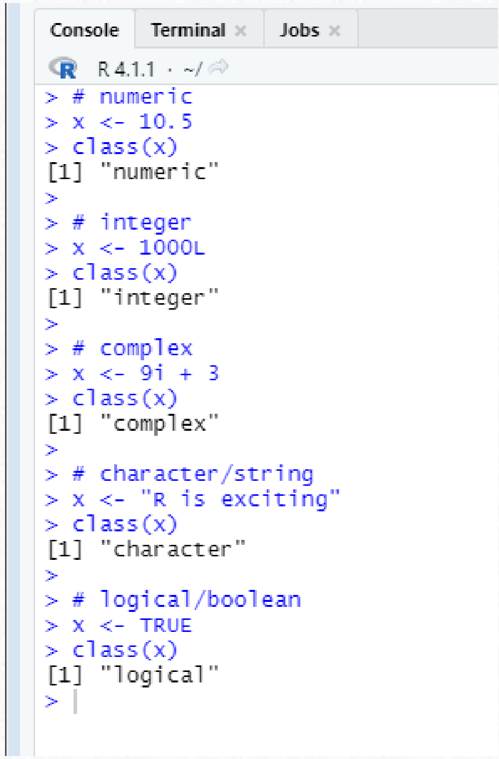
- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.

DATA TYPE	RANGE
Numeric	(10.5, 55, 787)
Integer	(1L, 55L, 100L, where the letter "L" declares this as an integer)
Complex (a.k.a. String)	(9 + 3i, where "i" is the imaginary part)
Character	("k", "R is exciting", "FALSE", "11.5")
Logical (a.k.a. Boolean)	(TRUE or FALSE)

DATA TYPE’s (Cont.):

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

# integer
x <- 1000L
class(x)

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

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

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



```
> # numeric
> x <- 10.5
> class(x)
[1] "numeric"
>
> # integer
> x <- 1000L
> class(x)
[1] "integer"
>
> # complex
> x <- 9i + 3
> class(x)
[1] "complex"
>
> # character/string
> x <- "R is exciting"
> class(x)
[1] "character"
>
> # logical/boolean
> x <- TRUE
> class(x)
[1] "logical"
>
```

NUMBER:

- There are three number types in R:

✓Numeric

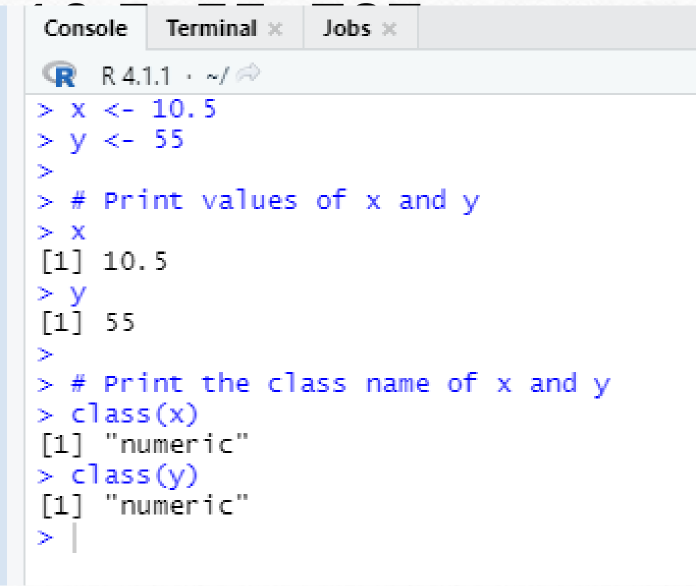
✓Integer

✓Complex

```
x <- 10.5   # numeric
 y <- 10L    # integer
 z <- 1i     # complex
```

NUMERIC:

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



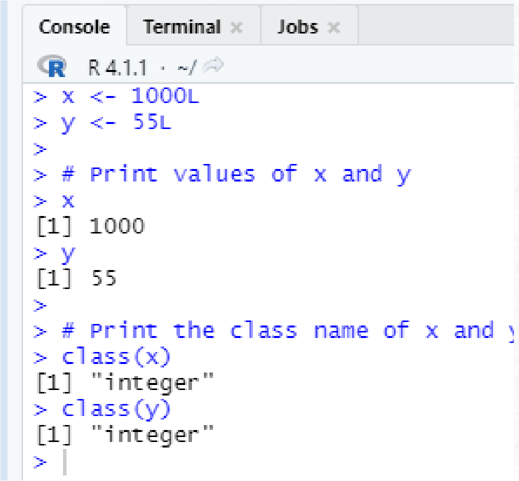
```
> x <- 10.5
> y <- 55
>
> # Print values of x and y
> x
[1] 10.5
> y
[1] 55
>
> # Print the class name of x and y
> class(x)
[1] "numeric"
> class(y)
[1] "numeric"
>
```

```
x <- 10.5
y <- 55

# Print values of x and y
x
y
# Print the class name of x and y
class(x)
class(y)
```

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.

```
x <- 1000L
y <- 55L
# Print values of x and y
x
y
# Print the class name of x and y
class(x)
class(y)
```

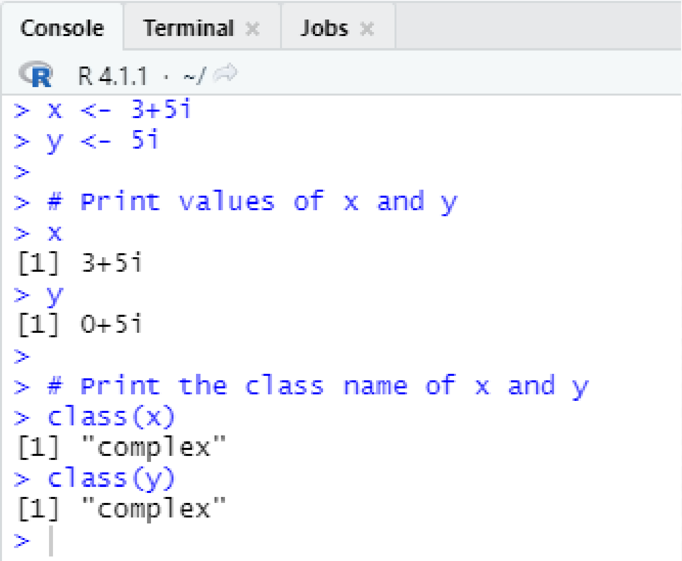


```
> x <- 1000L
> y <- 55L
>
> # Print values of x and y
> x
[1] 1000
> y
[1] 55
>
> # Print the class name of x and y
> class(x)
[1] "integer"
> class(y)
[1] "integer"
>
```

COMPLEX:

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

```
x <- 3+5i
y <- 5i
# Print values of x and y
x
y
# Print the class name of x and y
class(x)
class(y)
```



```
> x <- 3+5i
> y <- 5i
>
> # Print values of x and y
> x
[1] 3+5i
> y
[1] 0+5i
>
> # Print the class name of x and y
> class(x)
[1] "complex"
> class(y)
[1] "complex"
>
```

MATH:

- In R, you can use operators to perform common mathematical operations on numbers.

10 + 5   #addition

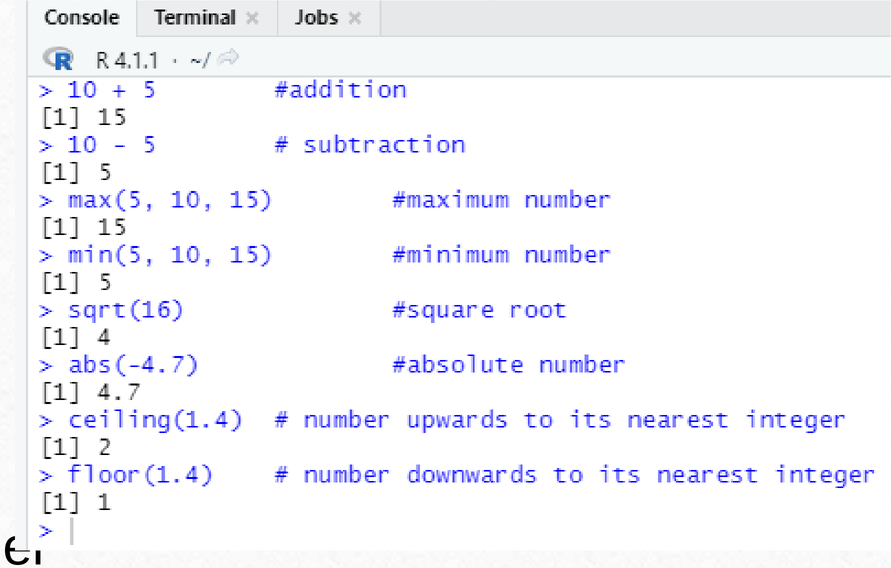
10 - 5   # subtraction

max(5, 10, 15)   #maximum number
min(5, 10, 15)   #minimum number

sqrt(16)   #square root

abs(-4.7)   #absolute number

ceiling(1.4) # number upwards to its nearest integer
floor(1.4)   # number downwards to its nearest integer

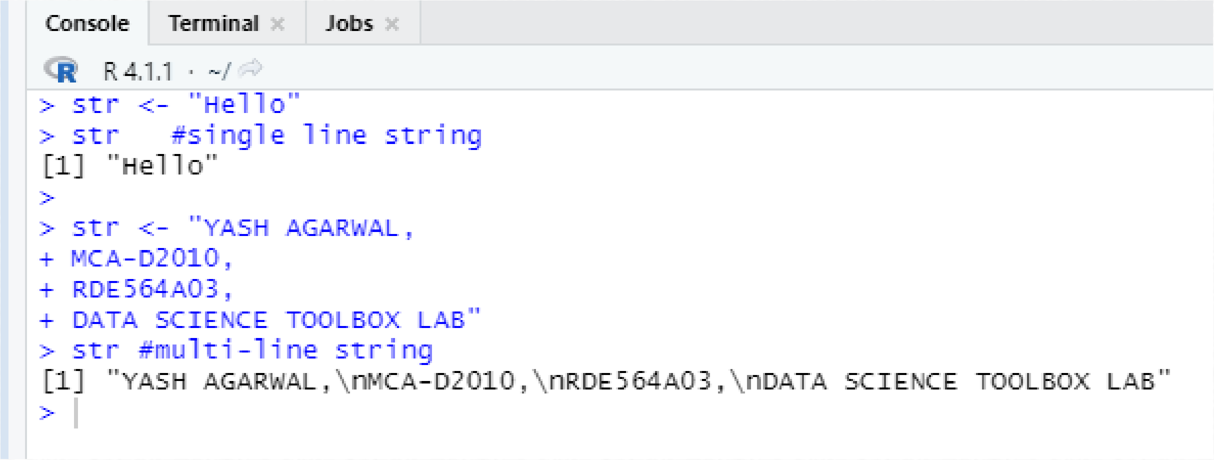


```
> 10 + 5        #addition
[1] 15
> 10 - 5        # subtraction
[1] 5
> max(5, 10, 15)        #maximum number
[1] 15
> min(5, 10, 15)        #minimum number
[1] 5
> sqrt(16)              #square root
[1] 4
> abs(-4.7)             #absolute number
[1] 4.7
> ceiling(1.4)  # number upwards to its nearest integer
[1] 2
> floor(1.4)    # number downwards to its nearest integer
[1] 1
>
```

STRING:

- A character, or strings, are used for storing text. A string is surrounded by either single quotation marks, or double quotation marks.

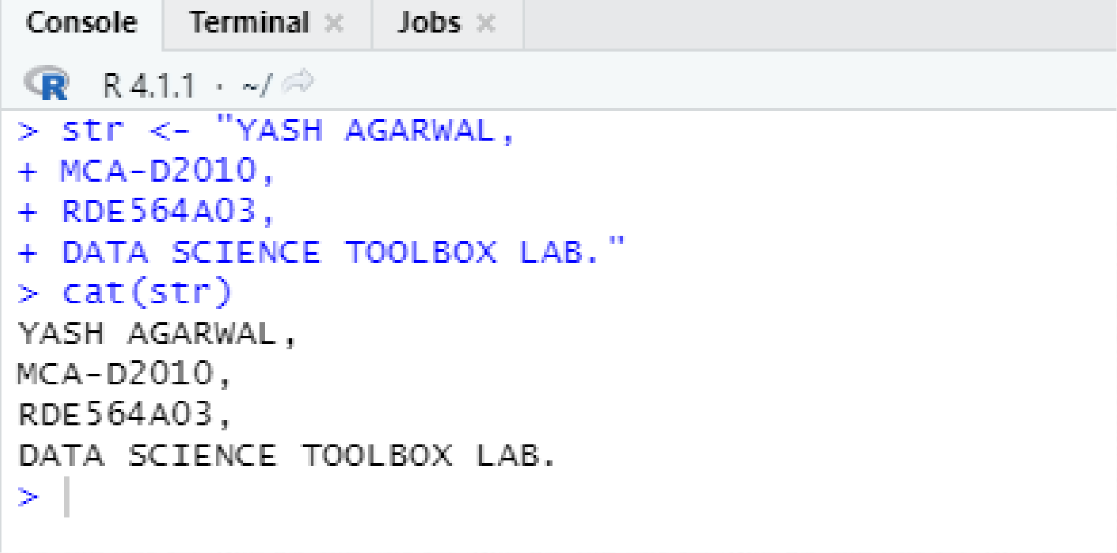
```
str <- "Hello"
str  #single line string

str <- "YASH AGARWAL,

MCA-D2010,

RDE564A03,

DATA SCIENCE TOOLBOX LAB"

str #multi-line string
```



```
> str <- "Hello"
> str   #single line string
[1] "Hello"
>
> str <- "YASH AGARWAL,
+ MCA-D2010,
+ RDE564A03,
+ DATA SCIENCE TOOLBOX LAB"
> str #multi-line string
[1] "YASH AGARWAL,\nMCA-D2010,\nRDE564A03,\nDATA SCIENCE TOOLBOX LAB"
> |
```

STRING (cont.):

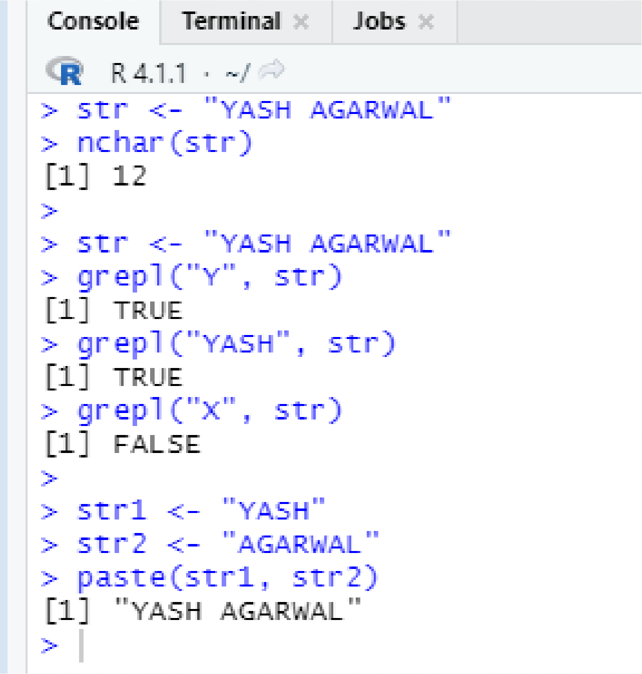
```
str <- "YASH AGARWAL,
MCA-D2010,
RDE564A03,
DATA SCIENCE TOOLBOX LAB."
cat(str)
```



```
> str <- "YASH AGARWAL,
+ MCA-D2010,
+ RDE564A03,
+ DATA SCIENCE TOOLBOX LAB."
> cat(str)
YASH AGARWAL,
MCA-D2010,
RDE564A03,
DATA SCIENCE TOOLBOX LAB.
> 
```

STRING (cont.):

```
str <- "YASH AGARWAL"

nchar(str)

str <- "YASH AGARWAL"
grepl("Y", str)
grepl("YASH", str)
grepl("X", str)


str1 <- "YASH"
str2 <- "AGARWAL"
paste(str1, str2)
```



```
> str <- "YASH AGARWAL"
> nchar(str)
[1] 12
>
> str <- "YASH AGARWAL"
> grepl("Y", str)
[1] TRUE
> grepl("YASH", str)
[1] TRUE
> grepl("X", str)
[1] FALSE
>
> str1 <- "YASH"
> str2 <- "AGARWAL"
> paste(str1, str2)
[1] "YASH AGARWAL"
>
```

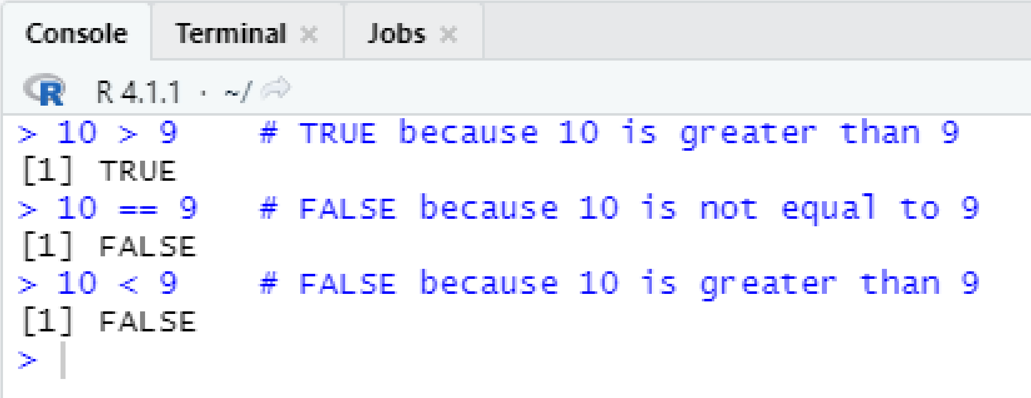
STRING (cont.):

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

LOGICAL/BOOLEAN:

- You can evaluate any expression in R, and get one of two answers, TRUE or FALSE

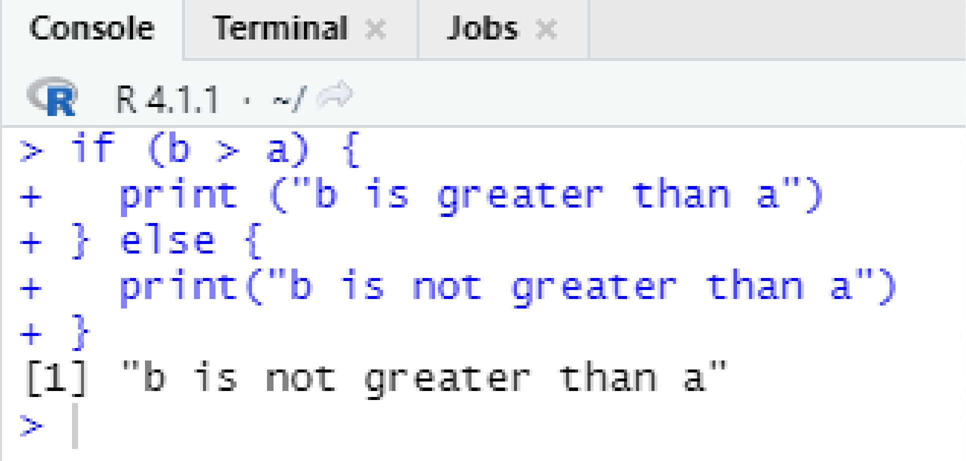
```
10 > 9    # TRUE because 10 is greater than
10 == 9   # FALSE because 10 is not equal
10 < 9    # FALSE because 10 is greater tha
```



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

LOGICAL/BOOLEAN:

```
a <- 200
b <- 33

if (b > a) {
  print ("b is greater than a")
} else {
  print("b is not greater than a")
}
```



```
> if (b > a) {
+   print ("b is greater than a")
+ } else {
+   print("b is not greater than a")
+ }
[1] "b is not greater than a"
> 
```

THANK YOU!