FDA Lab-1

KHAN MOHD OWAIS RAZA
20BCD7138

Arithmetic Operations

```
# KHAN MOHD OWAIS RAZA
# 20BCD7138

# Addition
a <- 10
b <- 5
sum <- a + b
print(sum)

# Subtraction
diff <- a - b
print(diff)

# Division
div <- a / b
print(div)

# Multiplication
mult <- a * b
print(mult)

# Modulo (remainder)
mod <- a %% b
print(mod)

# Exponentiation
exp <- a ^ b
print(exp)
```

```
> #KHAN MOHD OWAIS RAZA
> # 20BCD7138
>
> # Addition
> a <- 10
> b <- 5
> sum <- a + b
> print(sum)
[1] 15
>
> # Subtraction
> diff <- a - b
> print(diff)
[1] 5
>
> # Division
> div <- a / b
> print(div)
[1] 2
>
> # Multiplication
> mult <- a * b
> print(mult)
[1] 50
>
> # Modulo (remainder)
> mod <- a %% b
> print(mod)
[1] 0
>
> # Exponentiation
> exp <- a ^ b
> print(exp)
[1] 1e+05
>
```

Assignment Operations

```
# KHAN MOHD OWAIS RAZA
# 20BCD7138
x <- 5
y <- x + 3
# Assignment Operators
x <- 10  # x is assigned the value 10
print(x)
x <- x + 3  # x is incremented by 3 and assigned the result
print(x)
x <- x - 2  # x is decremented by 2 and assigned the result
print(x)
x <- x * 2  # x is multiplied by 2 and assigned the result
print(x)
x <- x / 3  # x is divided by 3 and assigned the result
print(x)
x <- x %% 4  # x is divided by 4 and the remainder is assigned
print(x)
x <- x ^ 2  # x is raised to the power of 2 and assigned the result
print(x)
```

```
> # KHAN MOHD OWAIS RAZA
> # 20BCD7138
>
> x <- 5
> y <- x + 3
>
> # Assignment Operators
> x <- 10  # x is assigned the value 10
> print(x)
[1] 10
>
> x <- x + 3  # x is incremented by 3 and assigned the result
> print(x)
[1] 13
>
> x <- x - 2  # x is decremented by 2 and assigned the result
> print(x)
[1] 11
>
> x <- x * 2  # x is multiplied by 2 and assigned the result
> print(x)
[1] 22
>
> x <- x / 3  # x is divided by 3 and assigned the result
> print(x)
[1] 7.333333
>
> x <- x %% 4  # x is divided by 4 and the remainder is assigned
> print(x)
[1] 3.333333
>
> x <- x ^ 2  # x is raised to the power of 2 and assigned the result
> print(x)
[1] 11.11111
```

Checking varables

```
# KHAN MOHD OWAIS RAZA
# 20BCD7138
# Create variables
x <- 5
y <- "Hello, world!"
z <- c(1, 2, 3, 4, 5)
# Check variable types
print(class(x))  # Check type of variable x
print(class(y))  # Check type of variable y
print(class(z))  # Check type of variable z
# Check variable values
print(x)  # Print value of variable x
print(y)  # Print value of variable y
print(z)  # Print value of variable z
# Check variable attributes
print(attributes(x))  # Check attributes of variable x
print(attributes(y))  # Check attributes of variable y
print(attributes(z))  # Check attributes of variable z
```

```
> # KHAN MOHD OWAIS RAZA
> # 20BCD7138
>
> # Create variables
> x <- 5
> y <- "Hello, world!"
> z <- c(1, 2, 3, 4, 5)
>
> # Check variable types
> print(class(x))  # Check type of variable x
[1] "numeric"
> print(class(y))  # Check type of variable y
[1] "character"
> print(class(z))  # Check type of variable z
[1] "numeric"
>
> # Check variable values
> print(x)  # Print value of variable x
[1] 5
> print(y)  # Print value of variable y
[1] "Hello, world!"
> print(z)  # Print value of variable z
[1] 1 2 3 4 5
>
> # Check variable attributes
> print(attributes(x))  # Check attributes of variable x
NULL
> print(attributes(y))  # Check attributes of variable y
NULL
> print(attributes(z))  # Check attributes of variable z
NULL
>
```

Variable manipulation

```
# KHAN MOHD OWAIS RAZA
# 20BCD7138

# Variable assignment
x <- 5
y <- "Hello"
z <- TRUE
# Display variable values
x
y
z
# Arithmetic operations
a <- x + 3
b <- sqrt(a)
# Concatenation
greeting <- paste(y, "World!")
# Logical operations
is_positive <- a > 0
is_equal <- a == b
# Data transformation
upper_greeting <- toupper(greeting)
# Vector creation
numbers <- c(1, 2, 3, 4, 5)
# Accessing vector elements
second_number <- numbers[2]
# Data frame creation
df <- data.frame(Name = c("John", "Emily", "Michael"), Age = c(25, 30, 35))
# Accessing data frame columns
names <- df$Name
ages <- df$Age
```

```
> # KHAN MOHD OWAIS RAZA
> # 20BCD7138
>
> # Variable assignment
> x <- 5
> y <- "Hello"
> z <- TRUE
> # Display variable values
> x
[1] 5
> y
[1] "Hello"
> z
[1] TRUE
> # Arithmetic operations
> a <- x + 3
> b <- sqrt(a)
> # Concatenation
> greeting <- paste(y, "World!")
```

```
> # Logical operations
> is_positive <- a > 0
> is_equal <- a == b
> # Data transformation
> upper_greeting <- toupper(greeting)
> # Vector creation
> numbers <- c(1, 2, 3, 4, 5)
> # Accessing vector elements
> second_number <- numbers[2]
> # Data frame creation
> df <- data.frame(Name = c("John", "Emily", "Michael"), Age = c(25, 30, 35))
> # Accessing data frame columns
> names <- df$Name
> ages <- df$Age
```

Comparison operators

```
#KHAN MOHD OWAIS RAZA
#20BCD7138

x <- 5
y <- 3

# Equal to
isEqual <- x == y
print(isEqual)

# Not equal to
isNotEqual <- x != y
print(isNotEqual)

# Greater than
isGreaterThan <- x > y
print(isGreaterThan)

# Less than
isLessThan <- x < y
print(isLessThan)

# Greater than or equal to
isGreaterThanOrEqual <- x >= y
print(isGreaterThanOrEqual)

# Less than or equal to
isLessThanOrEqual <- x <= y
print(isLessThanOrEqual)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> x <- 5
> y <- 3
>
> # Equal to
> isEqual <- x == y
> print(isEqual)
[1] FALSE
>
> # Not equal to
> isNotEqual <- x != y
> print(isNotEqual)
[1] TRUE
>
> # Greater than
> isGreaterThan <- x > y
> print(isGreaterThan)
[1] TRUE
>
> # Less than
> isLessThan <- x < y
> print(isLessThan)
[1] FALSE
>
> # Greater than or equal to
> isGreaterThanOrEqual <- x >= y
> print(isGreaterThanOrEqual)
[1] TRUE
>
> # Less than or equal to
> isLessThanOrEqual <- x <= y
> print(isLessThanOrEqual)
[1] FALSE
```

Logical operators

```
# KHAN MOHD OWAIS RAZA
#20BCD7138

x <- TRUE
y <- FALSE

# Logical AND
andResult <- x & y
print(andResult)

# Logical OR
orResult <- x | y
print(orResult)

# Logical NOT
notResultX <- !x
notResultY <- !y
print(notResultX)
print(notResultY)

# Logical XOR
xorResult <- xor(x, y)
print(xorResult)
```

```
> # KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> x <- TRUE
> y <- FALSE
>
> # Logical AND
> andResult <- x & y
> print(andResult)
[1] FALSE
>
> # Logical OR
> orResult <- x | y
> print(orResult)
[1] TRUE
>
> # Logical NOT
> notResultX <- !x
> notResultY <- !y
> print(notResultX)
[1] FALSE
> print(notResultY)
[1] TRUE
>
> # Logical XOR
> xorResult <- xor(x, y)
> print(xorResult)
[1] TRUE
>
```

Membership operators

```
# KHAN MOHD OWAIS RAZA
# 20BCD7138

x <- c(1, 2, 3)
y <- c(3, 4, 5)
# %in% membership operator
isInX <- 2 %in% x
isInY <- 2 %in% y
print(isInX)
print(isInY)
```

```
> # KHAN MOHD OWAIS RAZA
> # 20BCD7138
>
> x <- c(1, 2, 3)
> y <- c(3, 4, 5)
> # %in% membership operator
> isInX <- 2 %in% x
> isInY <- 2 %in% y
> print(isInX)
[1] TRUE
> print(isInY)
[1] FALSE
>
```

Concatenation operators

```
#KHAN MOHD OWAIS RAZA
#20BCD7138

x <- c(1, 2, 3)
y <- c(3, 4, 5)
# Concatenation operator
concatenated <- c(x, y)
print(concatenated)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> x <- c(1, 2, 3)
> y <- c(3, 4, 5)
> # Concatenation operator
> concatenated <- c(x, y)
> print(concatenated)
[1] 1 2 3 3 4 5
>
```

Matrix

```
#KHAN MOHD OWAIS RAZA
#20BCD7138

vector <- c(1, 2, 3, 4, 5, 6)
# Create matrix
matrix <- matrix(vector, nrow = 2, ncol = 3, byrow = TRUE)
print(matrix)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> vector <- c(1, 2, 3, 4, 5, 6)
> # Create matrix
> matrix <- matrix(vector, nrow = 2, ncol = 3, byrow = TRUE)
> print(matrix)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
>
```

Arrays

```
#KHAN MOHD OWAIS RAZA
#20BCD7138

vector <- c(1, 2, 3, 4, 5, 6)
# Create array
array <- array(vector, dim = c(2, 3, 2))
print(array)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> vector <- c(1, 2, 3, 4, 5, 6)
> # Create array
> array <- array(vector, dim = c(2, 3, 2))
> print(array)
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

>
```

Lists

```
#KHAN MOHD OWAIS RAZA
#20BCD7138

name <- "John"
age <- 25
scores <- c(80, 90, 95)
# Create list
myList <- list(name = name, age = age, scores = scores)
print(myList)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
>
> name <- "John"
> age <- 25
> scores <- c(80, 90, 95)
> # Create list
> myList <- list(name = name, age = age, scores = scores)
> print(myList)
$name
[1] "John"

$age
[1] 25

$scores
[1] 80 90 95
```

Dataframe

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
names <- c("John", "Emma", "David")
age <- c(25, 30, 27)
scores <- c(80, 90, 85)
# Create data frame
df <- data.frame(names = names, age = age, scores = scores)
print(df)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
> names <- c("John", "Emma", "David")
> age <- c(25, 30, 27)
> scores <- c(80, 90, 85)
> # Create data frame
> df <- data.frame(names = names, age = age, scores = scores)
> print(df)
  names age scores
1  John  25     80
2  Emma  30     90
3 David  27     85
>
```

Vectors

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
numeric_vector <- c(1, 2, 3, 4, 5)
character_vector <- c("red", "green", "blue")
logical_vector <- c(TRUE, FALSE, TRUE)
print(numeric_vector)
print(character_vector)
print(logical_vector)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
> numeric_vector <- c(1, 2, 3, 4, 5)
> character_vector <- c("red", "green", "blue")
> logical_vector <- c(TRUE, FALSE, TRUE)
> print(numeric_vector)
[1] 1 2 3 4 5
> print(character_vector)
[1] "red"   "green" "blue"
> print(logical_vector)
[1]  TRUE FALSE  TRUE
>
```

Factors

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
color_vector <- c("red", "green", "blue", "red", "green", "blue")
color_factor <- factor(color_vector)
print(color_factor)
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
> color_vector <- c("red", "green", "blue", "red", "green", "blue")
> color_factor <- factor(color_vector)
> print(color_factor)
[1] red   green blue  red   green blue
Levels: blue green red
>
```

Conditional statement (if-else)

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
x <- 10
if (x > 0) {
  print("Positive")
} else {
  print("Negative or zero")
}
```

```
> #KHAN MOHD OWAIS RAZA
> #20BCD7138
> x <- 10
> if (x > 0) {
+   print("Positive")
+ } else {
+   print("Negative or zero")
+ }
[1] "Positive"
>
```

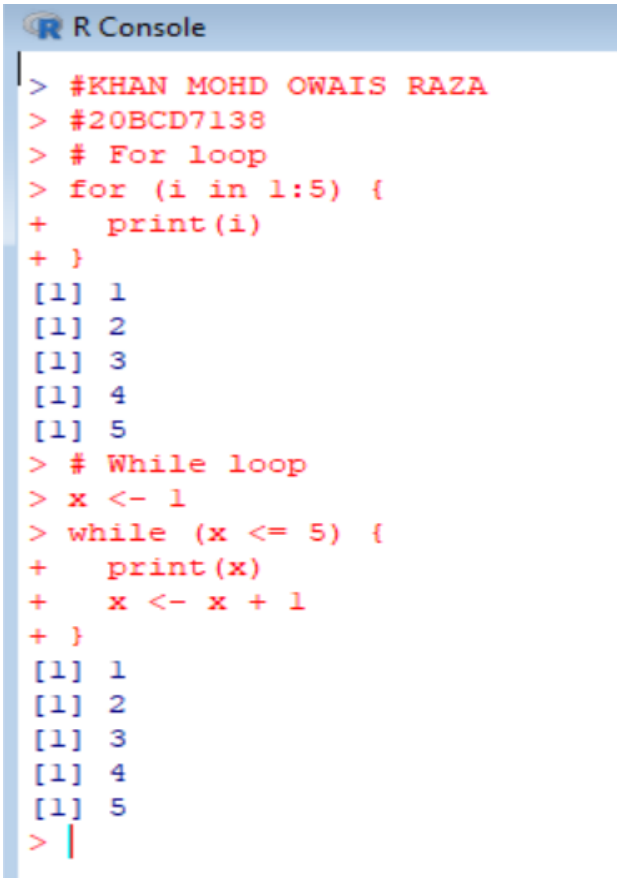
Nested Conditional Statements (if-else if-else)

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
x <- 0
if (x > 0) {
  print("Positive")
} else if (x < 0) {
  print("Negative")
} else {
  print("Zero")
}
```

```
> x <- 0
> if (x > 0) {
+   print("Positive")
+ } else if (x < 0) {
+   print("Negative")
+ } else {
+   print("Zero")
+ }
[1] "Zero"
>
```

Loops

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
# For loop
for (i in 1:5) {
  print(i)
}
# While loop
x <- 1
while (x <= 5) {
  print(x)
  x <- x + 1
}
```



Control statement

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
# Break statement
for (i in 1:10) {
  if (i == 5) {
    break
  }
  print(i)
}
# Next statement
for (i in 1:10) {
  if (i %% 2 == 0) {
    next
  }
  print(i)
}
```