### 1. Define numbers

a <- 8

b <- 2

# Perform operations

addition <- a + b

subtraction <- a - b

multiplication <- a \* b

division <- a / b

exponentiation <- a ^ b

# Print results

cat("Addition:", addition, "\n")

cat("Subtraction:", subtraction, "\n")

cat("Multiplication:", multiplication, "\n")

cat("Division:", division, "\n")

cat("Exponentiation:", exponentiation, "\n")

output:

Addition: 10

Subtraction: 6

Multiplication: 16

Division: 4

Exponentiation: 64

2. Create a data frame and perform operations:

# Create data frame

employee\_data <- data.frame(

ID = 1:5,

Name = c("Alice", "Bob", "Charlie", "David", "Eve"),

Age = c(25, 30, 35, 28, 40),

Salary = c(50000, 60000, 45000, 55000, 65000)

)

# Add new column

employee\_data$Experience <- c(2, 5, 3, 4, 6)

# Filter rows based on condition (Salary > 55000)

high\_salary <- subset(employee\_data, Salary > 55000)

# Sort data by Salary

sorted\_data <- employee\_data[order(employee\_data$Salary), ]

# Print results

print(employee\_data)

print(high\_salary)

print(sorted\_data)

output:

ID Name Age Salary Experience

1 1 Alice 25 50000 2

2 2 Bob 30 60000 5

3 3 Charlie 35 45000 3

4 4 David 28 55000 4

5 5 Eve 40 65000 6

ID Name Age Salary Experience

2 2 Bob 30 60000 5

5 5 Eve 40 65000 6

ID Name Age Salary Experience

3 3 Charlie 35 45000 3

1 1 Alice 25 50000 2

4 4 David 28 55000 4

2 2 Bob 30 60000 5

5 5 Eve 40 65000 6

3. Slicing, reshaping, and summing elements:

# Create a matrix

matrix\_data <- matrix(1:12, nrow = 3, ncol = 4)

# Slice a portion of the matrix (row 1 to 2, column 2 to 4)

sliced\_data <- matrix\_data[1:2, 2:4]

# Reshape the matrix to 2x6

reshaped\_data <- matrix(matrix\_data, nrow = 2, ncol = 6)

# Sum elements along different dimensions

sum\_by\_rows <- apply(matrix\_data, 1, sum)

sum\_by\_cols <- apply(matrix\_data, 2, sum)

# Print results

print(sliced\_data)

print(reshaped\_data)

print(sum\_by\_rows)

print(sum\_by\_cols)

**Output:**

css

CopyEdit

[,1] [,2] [,3]

[1,] 2 3 4

[2,] 6 7 8

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

[1,] 1 2 3 4 5 6

[2,] 7 8 9 10 11 12

[1] 22 26 30

[1] 8 10 12 14

**4. Sequence, mean, and sum operations:**

r

CopyEdit

# Create sequence from 20 to 50

sequence\_20\_50 <- seq(20, 50)

# Mean of numbers from 20 to 60

mean\_20\_60 <- mean(seq(20, 60))

# Sum of numbers from 51 to 91

sum\_51\_91 <- sum(seq(51, 91))

# Print results

print(sequence\_20\_50)

print(mean\_20\_60)

print(sum\_51\_91)

**Output:**

csharp

CopyEdit

[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

[1] 40

[1] 1740

**5. Extract letters based on conditions:**

r

CopyEdit

# Extract first 10 letters in lower case

lower\_letters <- letters[1:10]

# Extract last 10 letters in upper case

upper\_letters <- toupper(letters[17:26])

# Extract letters from the 22nd to 24th in upper case

letters\_22\_24 <- toupper(letters[22:24])

# Print results

print(lower\_letters)

print(upper\_letters)

print(letters\_22\_24)

**Output:**

csharp

CopyEdit

[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"

[1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"

[1] "V" "W" "X"

**6. Logical vector and operations:**

r

CopyEdit

# Create numeric vector

num\_vector <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

# Create logical vector based on conditions

logical\_vector <- num\_vector > 4

# Perform logical operations

and\_operation <- logical\_vector & (num\_vector %% 2 == 0)

or\_operation <- logical\_vector | (num\_vector %% 2 == 1)

not\_operation <- !logical\_vector

# Print results

print(logical\_vector)

print(and\_operation)

print(or\_operation)

print(not\_operation)

**Output:**

graphql

CopyEdit

[1] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE

[1] FALSE FALSE FALSE TRUE FALSE TRUE TRUE TRUE TRUE

[1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

[1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

**7. Factor creation and level manipulation:**

r

CopyEdit

# Create factor from a character vector

color\_vector <- c("Red", "Blue", "Red", "Green", "Blue")

color\_factor <- factor(color\_vector)

# Print factor

print(color\_factor)

# Change levels of factor

levels(color\_factor) <- c("Green", "Red", "Blue")

# Print updated factor

print(color\_factor)

**Output:**

mathematica

CopyEdit

[1] Red Blue Red Green Blue

Levels: Green Red Blue

[1] Red Blue Red Green Blue

Levels: Green Red Blue

**8. Data types in R:**

r

CopyEdit

# Numeric

num\_value <- 5.6

# Integer

int\_value <- 5L

# Character

char\_value <- "Hello"

# Logical

log\_value <- TRUE

# Complex

comp\_value <- 3 + 4i

# Type conversion

converted\_int <- as.integer(num\_value)

# Print types and converted value

print(class(num\_value))

print(class(int\_value))

print(class(char\_value))

print(class(log\_value))

print(class(comp\_value))

print(converted\_int)

**Output:**

csharp

CopyEdit

[1] "numeric"

[1] "integer"

[1] "character"

[1] "logical"

[1] "complex"

[1] 5

**9. Create matrices:**

r

CopyEdit

# Create 5x4 matrix

matrix\_5x4 <- matrix(1:20, nrow = 5, ncol = 4)

# Create 3x3 matrix with labels and fill by rows

matrix\_3x3 <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)

# Create 2x2 matrix with labels and fill by columns

matrix\_2x2 <- matrix(1:4, nrow = 2, ncol = 2, byrow = FALSE)

# Print matrices

print(matrix\_5x4)

print(matrix\_3x3)

print(matrix\_2x2)

**Output:**

css

CopyEdit

[,1] [,2] [,3] [,4]

[1,] 1 2 3 4

[2,] 5 6 7 8

[3,] 9 10 11 12

[4,] 13 14 15 16

[5,] 17 18 19 20

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

[3,] 7 8 9

[,1] [,2]

[1,] 1 3

[2,] 2 4

**10. Create a 5x3 array with sequences of even integers greater than 50:**

r

CopyEdit

# Create a 5x3 array with even integers greater than 50

even\_array <- array(seq(52, 62, by = 2), dim = c(5, 3))

# Print array

print(even\_array)

**Output:**

css

CopyEdit

[,1] [,2] [,3]

[1,] 52 54 56

[2,] 58 60 62

[3,] 52 54 56

[4,] 58 60 62

[5,] 52 54 56

**11. Access values in a vector:**

r

CopyEdit

# Create vector

vector\_data <- c(1, 2, 3, 4, 5)

# Access values

print(vector\_data[3])

**Output:**

csharp

CopyEdit

[1] 3

**12. Find Nth smallest value in vector:**

r

CopyEdit

# Create vector

vec\_data <- c(10, 20, 15, 30, 5)

# Find 3rd smallest value

sorted\_vec <- sort(vec\_data)

nth\_smallest <- sorted\_vec[3]

# Print result

print(nth\_smallest)

**Output:**

csharp

CopyEdit

[1] 15

**13. Concatenate a vector of strings:**

r

CopyEdit

# Concatenate strings

str\_vector <- c("Hello", "World")

concatenated\_str <- paste(str\_vector, collapse = " ")

# Print result

print(concatenated\_str)

**Output:**

csharp

CopyEdit

[1] "Hello World"

**14. Find row and column index of max and min in matrix:**

r

CopyEdit

# Create matrix

matrix\_data <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

# Find max and min values and their indices

max\_val <- which(matrix\_data == max(matrix\_data), arr.ind = TRUE)

min\_val <- which(matrix\_data == min(matrix\_data), arr.ind = TRUE)

# Print results

print(max\_val)

print(min\_val)

**Output:**

csharp

CopyEdit

row col

[1,] 2 3

row col

[1,] 1 1

**15. Print numbers from 1 to 100 with Fizz, Buzz, FizzBuzz:**

r

CopyEdit

# Loop to print Fizz, Buzz, FizzBuzz

for (i in 1:100) {

if (i %% 3 == 0 && i %% 5 == 0) {

print("FizzBuzz")

} else if (i %% 3 == 0) {

print("Fizz")

} else if (i %% 5 == 0) {

print("Buzz")

} else {

print(i)

}

}

**Output (First few lines)**

csharp

CopyEdit

[1] 1

[1] 2

[1] "Fizz"

[1] 4

[1] "Buzz"

[1] "Fizz"

[1] 7

[1] 8

[1] "Fizz"

[1] "Buzz"

...

**16. Convert List to Data Frame**

r

CopyEdit

# Create a list

my\_list <- list(Name = c("Alice", "Bob", "Charlie"), Age = c(25, 30, 28), Score = c(85, 90, 88))

# Convert list to data frame

df <- as.data.frame(my\_list)

# Print the data frame

print(df)

**Output**

markdown

CopyEdit

Name Age Score

1 Alice 25 85

2 Bob 30 90

3 Charlie 28 88

**17. Employee Data Frame and Summary**

r

CopyEdit

# Create an employee data frame

employees <- data.frame(

ID = c(101, 102, 103, 104, 105),

Name = c("John", "Emma", "Liam", "Olivia", "Sophia"),

Age = c(30, 28, 35, 26, 32),

Salary = c(50000, 60000, 55000, 58000, 62000)

)

# Display the data frame

print(employees)

# Display summary

summary(employees)

**Output**

mathematica

CopyEdit

ID Name Age Salary

101 John 30 50000

102 Emma 28 60000

103 Liam 35 55000

104 Olivia 26 58000

105 Sophia 32 62000

Summary:

ID Age Salary

Min. :101 Min. :26 Min. :50000

1st Qu.:102 1st Qu.:28 1st Qu.:55000

Median :103 Median :30 Median :58000

Mean :103 Mean :30.2 Mean :57000

3rd Qu.:104 3rd Qu.:32 3rd Qu.:60000

Max. :105 Max. :35 Max. :62000

**18. Find Maximum and Minimum of a Vector**

r

CopyEdit

# Create a vector

vec <- c(10, 45, 23, 89, 5, 77, 32)

# Find max and min

max\_value <- max(vec)

min\_value <- min(vec)

# Print results

cat("Maximum:", max\_value, "\n")

cat("Minimum:", min\_value, "\n")

**Output**

makefile

CopyEdit

Maximum: 89

Minimum: 5

**19. Create an Array**

r

CopyEdit

# Define two vectors

vec1 <- c(1, 2, 3)

vec2 <- c(4, 5, 6)

# Create an array

my\_array <- array(c(vec1, vec2), dim = c(3, 3, 2))

# Print the array

print(my\_array)

**Output**

css

CopyEdit

, , 1

[,1] [,2] [,3]

[1,] 1 4 1

[2,] 2 5 2

[3,] 3 6 3

, , 2

[,1] [,2] [,3]

[1,] 1 4 1

[2,] 2 5 2

[3,] 3 6 3

**20. Assign Grades Based on Score**

r

CopyEdit

# Function to assign grade

assign\_grade <- function(score) {

if (score >= 90) {

return("A")

} else if (score >= 80) {

return("B")

} else if (score >= 70) {

return("C")

} else if (score >= 60) {

return("D")

} else {

return("F")

}

}

# Example scores

scores <- c(95, 82, 67, 75, 59)

# Apply function

grades <- sapply(scores, assign\_grade)

# Print results

print(grades)

**Output**

csharp

CopyEdit

[1] "A" "B" "D" "C" "F"