

Name : K.Nichal haas
RegNo: 22BCE9651
Slot : L22+L23
Prof :Posham Uppamma

Vellore Institute of Technology
SCOPE
FDA (CSE1006) - Slot - L22+L23

DA - V

- 1. Create a 8×8 NumPy array with random integers between 50 and 300. Perform the following operations**
- i. Find the standard deviation, mean and variance of the entire array.**

```
1 # 1. Create 8x8 matrix with random integers between 50 and 300
2 print("K.Nichal haas RegNo:22BCE9651")
3 set.seed(123) # for reproducibility
4 A <- matrix(sample(50:300, 64, replace = TRUE), nrow = 8, ncol = 8)
5 A
6
7 # i. Standard deviation, mean, variance
8 std_dev <- sd(A)
9 mean_val <- mean(A)
10 var_val <- var(as.vector(A))
11
12 cat("Standard Deviation:", std_dev, "\n")
13 cat("Mean:", mean_val, "\n")
14 cat("Variance:", var_val, "\n\n")
```

Program input
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [1,] 208 92 246 148 127 283 237 83 [2,] 256 278 292 121 130 158 102 270 [3,] 228 293 140 75 92 56 184 118 [4,] 63 63 234 56 152 186 102 121 [5,] 243 167 141 219 166 218 204 125 [6,] 219 202 298 186 125 123 273 112 [7,] 99 139 186 260 192 72 215 190 [8,] 167 148 270 213 81 204 266 259 Standard Deviation: 69.94419 Mean: 174.5156 Variance: 4892.19 [Execution complete with exit code 0]

A <- matrix(sample(50:300, 64, replace = TRUE), nrow = 8, ncol = 8)

A

```
# i. Standard deviation, mean, variance
std_dev <- sd(A)
mean_val <- mean(A)
var_val <- var(as.vector(A))

cat("Standard Deviation:", std_dev, "\n")
cat("Mean:", mean_val, "\n")
cat("Variance:", var_val, "\n\n")
```

ii. Compute the column sum, row sum and diagonal sum and anti-diagonal sum.

```

1 # 1. Create 8x8 matrix with random integers between 50 and 300
2 print("K.Nichal has RegNo:22BCE9651")
3 set.seed(123) # for reproducibility
4 A <- matrix(sample(50:300, 64, replace = TRUE), nrow = 8, ncol = 8)
5 A
6
7 # i. Standard deviation, mean, variance
8 std_dev <- sd(A)
9 mean_val <- mean(A)
10 var_val <- var(as.vector(A))
11
12 cat("Standard Deviation:", std_dev, "\n")
13 cat("Mean:", mean_val, "\n")
14 cat("Variance:", var_val, "\n\n")
15
16 # ii. Column sum, row sum, diagonal sum, anti-diagonal sum
17 col_sum <- colSums(A)
18 row_sum <- rowSums(A)
19 diag_sum <- sum(diag(A))
20 anti_diag_sum <- sum(diag(apply(A, 2, rev))) # reverse columns for anti-diagonal
21
22 cat("Column Sum:", col_sum, "\n")
23 cat("Row Sum:", row_sum, "\n")
24 cat("Diagonal Sum:", diag_sum, "\n")
25 cat("Anti-Diagonal Sum:", anti_diag_sum, "\n\n")

```

Program input	Output
<pre>[1,] 208 92 246 148 127 283 237 83 [2,] 256 278 292 121 130 158 102 270 [3,] 228 293 140 75 92 56 184 118 [4,] 63 63 234 56 152 186 102 121 [5,] 244 167 141 219 166 218 204 125 [6,] 219 202 298 186 125 123 273 112 [7,] 99 139 186 260 192 72 215 190 [8,] 167 140 270 213 81 204 266 259 Standard Deviation: 69.94419 Mean: 174.5156 Variance: 4892.19</pre>	<pre>Column Sum: 1484 1374 1807 1278 1065 1300 1583 1278 Row Sum: 1424 1607 1186 977 1484 1538 1353 1600 Diagonal Sum: 1445 Anti-Diagonal Sum: 1216</pre>
	[Execution complete with exit code 0]

```

# ii. Column sum, row sum, diagonal sum, anti-diagonal sum
col_sum <- colSums(A)
row_sum <- rowSums(A)
diag_sum <- sum(diag(A))
anti_diag_sum <- sum(diag(apply(A, 2, rev))) # reverse columns for anti-diagonal

cat("Column Sum:", col_sum, "\n")
cat("Row Sum:", row_sum, "\n")
cat("Diagonal Sum:", diag_sum, "\n")
cat("Anti-Diagonal Sum:", anti_diag_sum, "\n\n")

```

iii. Extract a sub-matrix of size 3×3 from the center of the array.

```

1 print("K.Nichal has RegNo:22BCE9651")
2 set.seed(123) # for reproducibility
3 A <- matrix(sample(50:300, 64, replace = TRUE), nrow = 8, ncol = 8)
4 A
5
6 # i. Standard deviation, mean, variance
7 std_dev <- sd(A)
8 mean_val <- mean(A)
9 var_val <- var(as.vector(A))
10
11 cat("Standard Deviation:", std_dev, "\n")
12 cat("Mean:", mean_val, "\n")
13 cat("Variance:", var_val, "\n\n")
14
15 # ii. Column sum, row sum, diagonal sum, anti-diagonal sum
16 col_sum <- colSums(A)
17 row_sum <- rowSums(A)
18 diag_sum <- sum(diag(A))
19 anti_diag_sum <- sum(diag(apply(A, 2, rev))) # reverse columns for anti-diagonal
20
21 cat("Column Sum:", col_sum, "\n")
22 cat("Row Sum:", row_sum, "\n")
23 cat("Diagonal Sum:", diag_sum, "\n")
24 cat("Anti-Diagonal Sum:", anti_diag_sum, "\n\n")
25
26 # iii. Extract 3x3 submatrix from center
27 sub_matrix <- A[3:5, 3:5]
28 cat("3x3 Sub-matrix from center:\n")
29 print(sub_matrix)

```

Program input	Output
<pre>[1,] 219 202 298 186 125 123 273 112 [2,] 99 139 186 260 192 72 215 190 [3,] 167 140 270 213 81 204 266 259 Standard Deviation: 69.94419 Mean: 174.5156 Variance: 4892.19</pre>	<pre>Column Sum: 1484 1374 1807 1278 1065 1300 1583 1278 Row Sum: 1424 1607 1186 977 1484 1538 1353 1600 Diagonal Sum: 1445 Anti-Diagonal Sum: 1216 3x3 Sub-matrix from center: [1,] 140 75 92 [2,] 234 56 152 [3,] 141 219 166</pre>
	[Execution complete with exit code 0]

```

# iii. Extract 3x3 submatrix from center
sub_matrix <- A[3:5, 3:5]
cat("3x3 Sub-matrix from center:\n")
print(sub_matrix)

```

iv. Replace all elements in the first row with -1.

```

R R v
Run Save

4 A <- matrix(sample(50:300, 64, replace = TRUE), nrow = 8, ncol = 8)
5 A
6
7 # i. Standard deviation, mean, variance
8 std_dev <- sd(A)
9 mean_val <- mean(A)
10 var_val <- var(as.vector(A))
11
12 cat("Standard Deviation:", std_dev, "\n")
13 cat("Mean:", mean_val, "\n")
14 cat("Variance:", var_val, "\n\n")
15
16 # ii. Column sum, row sum, diagonal sum, anti-diagonal sum
17 col_sum <- colSums(A)
18 row_sum <- rowSums(A)
19 diag_sum <- sum(diag(A))
20 anti_diag_sum <- sum(diag(apply(A, 2, rev))) # reverse columns for anti-diagonal
21
22 cat("Column Sum:", col_sum, "\n")
23 cat("Row Sum:", row_sum, "\n")
24 cat("Diagonal Sum:", diag_sum, "\n")
25 cat("Anti-Diagonal Sum:", anti_diag_sum, "\n\n")
26
27 # iii. Extract 3x3 submatrix from center
28 sub_matrix <- A[3:5, 3:5]
29 cat("3x3 Sub-matrix from center:\n")
30 print(sub_matrix)
31
32 # iv. Replace first row with -1
33 A[1, ] <- -1
34 cat("\nMatrix after replacing first row with -1:\n")
35 print(A)

```

Output

```

Program input
Output
3x3 Sub-matrix from center:
[,1] [,2] [,3]
[1,] 140   75   92
[2,] 234   56   152
[3,] 141   219  166

Matrix after replacing first row with -1:
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] -1   -1   -1   -1   -1   -1   -1   -1
[2,] 256  278  292  121 130 158 102 270
[3,] 228  293  140   75   92   56 184 118
[4,] 63   63 234   56 152 186 102 121
[5,] 244  167 141  219 166 218 204 125
[6,] 219  202 298 186 125 123 273 112
[7,] 99   139 186 260 192 72 215 190
[8,] 167  148 270 213  81 204 266 259

[Execution complete with exit code 0]

```

iv. Replace first row with -1

```

A[1, ] <- -1
cat("\nMatrix after replacing first row with -1:\n")
print(A)

```

v. Find the index position of the minimum value and maximum value in the array.

```

11
12 cat("Standard Deviation:", std_dev, "\n")
13 cat("Mean:", mean_val, "\n")
14 cat("Variance:", var_val, "\n\n")
15
16 # ii. Column sum, row sum, diagonal sum, anti-diagonal sum
17 col_sum <- colSums(A)
18 row_sum <- rowSums(A)
19 diag_sum <- sum(diag(A))
20 anti_diag_sum <- sum(diag(apply(A, 2, rev))) # reverse columns for anti-diagonal
21
22 cat("Column Sum:", col_sum, "\n")
23 cat("Row Sum:", row_sum, "\n")
24 cat("Diagonal Sum:", diag_sum, "\n")
25 cat("Anti-Diagonal Sum:", anti_diag_sum, "\n\n")
26
27 # iii. Extract 3x3 submatrix from center
28 sub_matrix <- A[3:5, 3:5]
29 cat("3x3 Sub-matrix from center:\n")
30 print(sub_matrix)
31
32 # iv. Replace first row with -1
33 A[1, ] <- -1
34 cat("\nMatrix after replacing first row with -1:\n")
35 print(A)
36
37 # v. Index of min and max value
38 min_index <- which(A == min(A), arr.ind = TRUE)
39 max_index <- which(A == max(A), arr.ind = TRUE)
40
41 cat("\nMin Value:", min(A), "at position", min_index, "\n")
42 cat("Max Value:", max(A), "at position", max_index, "\n")

```

Output

```

Program input
Output
[1,] 140   75   92
[2,] 234   56   152
[3,] 141   219  166

Matrix after replacing first row with -1:
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] -1   -1   -1   -1   -1   -1   -1   -1
[2,] 256  278  292  121 130 158 102 270
[3,] 228  293  140   75   92   56 184 118
[4,] 63   63 234   56 152 186 102 121
[5,] 244  167 141  219 166 218 204 125
[6,] 219  202 298 186 125 123 273 112
[7,] 99   139 186 260 192 72 215 190
[8,] 167  148 270 213  81 204 266 259

Min Value: -1 at position 1 1 1 1 1 1 1 1 2 3 4 5 6 7 8
Max Value: 296 at position 6 3

[Execution complete with exit code 0]

```

v. Index of min and max value

```

min_index <- which(A == min(A), arr.ind = TRUE)
max_index <- which(A == max(A), arr.ind = TRUE)

```

```

cat("\nMin Value:", min(A), "at position", min_index, "\n")
cat("Max Value:", max(A), "at position", max_index, "\n")

```

2. Generate two 4x4 NumPy arrays with random even values between 20 and 100. Perform the following

```
1 # Generate even numbers between 20 and 100
2 print("K.Nichal haas RegNo:22BCE9651")
3 even_nums <- seq(20, 100, by = 2)
4
5 # Create two 4x4 matrices
6 B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
7 C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
8
9 cat("Matrix B:\n"); print(B)
10 cat("Matrix C:\n"); print(C)
```

Program input

Output

```
[1] "K.Nichal haas RegNo:22BCE9651"
Matrix B:
 [,1] [,2] [,3] [,4]
 [1,] 20 90 98 86
 [2,] 100 68 86 80
 [3,] 22 38 34 32
 [4,] 42 62 32 28

Matrix C:
 [,1] [,2] [,3] [,4]
 [1,] 46 30 52 74
 [2,] 74 64 86 92
 [3,] 48 92 36 84
 [4,] 44 60 30 80
```

[Execution complete with exit code 0]

```
# Generate even numbers between 20 and 100
print("K.Nichal haas RegNo:22BCE9651")
even_nums <- seq(20, 100, by = 2)
```

```
# Create two 4x4 matrices
B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)

cat("Matrix B:\n"); print(B)
cat("Matrix C:\n"); print(C)
```

i. Perform element-wise multiplication of both arrays

```
1 # Generate even numbers between 20 and 100
2 print("K.Nichal haas RegNo:22BCE9651")
3 even_nums <- seq(20, 100, by = 2)
4
5 # Create two 4x4 matrices
6 B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
7 C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
8
9 cat("Matrix B:\n"); print(B)
10 cat("Matrix C:\n"); print(C)
11
12 # i. Element-wise multiplication
13 elem_mult <- B * C
14 cat("\nElement-wise multiplication:\n"); print(elem_mult)
```

Program input

Output

```
[1,] 56 68 28 64
[2,] 58 72 78 86
[3,] 26 52 76 24
[4,] 90 42 40 24

Matrix C:
 [,1] [,2] [,3] [,4]
 [1,] 30 82 26 36
 [2,] 58 34 44 30
 [3,] 64 64 78 48
 [4,] 28 20 66 34

Element-wise multiplication:
 [,1] [,2] [,3] [,4]
 [1,] 1680 5576 728 2304
 [2,] 3364 2448 3432 2580
 [3,] 1664 3328 5928 1152
 [4,] 2520 840 2640 816
```

[Execution complete with exit code 0]

```
# i. Element-wise multiplication
elem_mult <- B * C
cat("\nElement-wise multiplication:\n"); print(elem_mult)
```

ii. Find the dot product of the two matrices.

```

1 # Generate even numbers between 20 and 100
2 print("K.Nichal has RegNo:22BCE9651")
3 even_nums <- seq(20, 100, by = 2)
4
5 # Create two 4x4 matrices
6 B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
7 C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
8
9 cat("Matrix B:\n"); print(B)
10 cat("Matrix C:\n"); print(C)
11
12 # i. Element-wise multiplication
13 elem_mult <- B * C
14 cat("\nElement-wise multiplication:\n"); print(elem_mult)
15
16 # ii. Dot product (matrix multiplication)
17 dot_product <- B %*% C
18 cat("\nDot product (B %*% C):\n"); print(dot_product)

```

Program input

Output

```

[1,] 46 36 50 76
[2,] 54 84 72 24
[3,] 78 36 48 92
[4,] 2576 3168 1708 7448
[5,] 4212 8664 4464 2916
[6,] 4524 2232 3936 9200

```

Element-wise multiplication:

```

[1,] [2,] [3,] [4,]
[1,] 1344 2088 3520 2376
[2,] 2576 3168 1708 7448
[3,] 4212 8664 4464 2916
[4,] 4524 2232 3936 9200

```

Dot product (B %*% C):

```

[1,] [2,] [3,] [4,]
[1,] 11964 12824 11688 10872
[2,] 16664 12800 13792 18984
[3,] 18684 16212 16416 19944
[4,] 18328 16684 16124 18432

```

[Execution complete with exit code 0]

ii. Dot product (matrix multiplication)

```

dot_product <- B %*% C
cat("\nDot product (B %*% C):\n"); print(dot_product)

```

iii. Replace all values greater than 50 in the second array with 99.

```

1 # Generate even numbers between 20 and 100
2 print("K.Nichal has RegNo:22BCE9651")
3 even_nums <- seq(20, 100, by = 2)
4
5 # Create two 4x4 matrices
6 B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
7 C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
8
9 cat("Matrix B:\n"); print(B)
0 cat("Matrix C:\n"); print(C)
1
2 # i. Element-wise multiplication
3 elem_mult <- B * C
4 cat("\nElement-wise multiplication:\n"); print(elem_mult)
5
6 # ii. Dot product (matrix multiplication)
7 dot_product <- B %*% C
8 cat("\nDot product (B %*% C):\n"); print(dot_product)
9
0 #iii. Replace all values greater than 50 in the second array with 99
1 C[C > 50] <- 99
2 cat("\nMatrix C after replacing values > 50 with 99:\n"); print(C)

```

Program input

Output

```

[1,] 7896 1596 1920 2652
[2,] 5676 1048 2760 7748
[3,] 3000 5624 7600 2408

```

Dot product (B %*% C):

```

[1,] [2,] [3,] [4,]
[1,] 18360 13580 19840 21352
[2,] 9964 11652 10528 14252
[3,] 11176 13276 14760 16488
[4,] 16240 12504 17072 19136

```

Matrix C after replacing values > 50 with 99:

```

[1,] [2,] [3,] [4,]
[1,] 32 99 20 99
[2,] 99 42 99 99
[3,] 99 40 99 99
[4,] 50 99 99 99

```

[Execution complete with exit code 0]

C[C > 50] <- 99

```

cat("\nMatrix C after replacing values > 50 with 99:\n"); print(C)

```

iv. Concatenate both arrays vertically and then split into equal parts

```

1 # Generate even numbers between 20 and 100
2 print("K.Nichal has RegNo:22BCE9651")
3 even_nums <- seq(20, 100, by = 2)
4
5 # Create two 4x4 matrices
6 B <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
7 C <- matrix(sample(even_nums, 16, replace = TRUE), nrow = 4)
8
9 cat("Matrix B:\n"); print(B)
10 cat("Matrix C:\n"); print(C)
11
12 # i. Element-wise multiplication
13 elem_mult <- B * C
14 cat("\nElement-wise multiplication:\n"); print(elem_mult)
15
16 # ii. Dot product (matrix multiplication)
17 dot_product <- B %*% C
18 cat("\nDot product (B %*% C):\n"); print(dot_product)
19
20 #iii. Replace all values greater than 50 in the second array with 99
21 C[C > 50] <- 99
22 cat("\nMatrix C after replacing values > 50 with 99:\n"); print(C)
23
24 # iv. Concatenate vertically and split into equal parts
25 combined <- rbind(B, C)
26 cat("\nCombined matrix:\n"); print(combined)

```

Program input

Output

```

Matrix C after replacing values > 50 with 99:
[1,] [2,] [3,] [4,]
[1,] 46 99 99 99
[2,] 99 99 36 99
[3,] 26 99 99 20
[4,] 99 99 48 34

```

Combined matrix:

```

[1,] [2,] [3,] [4,]
[1,] 84 56 24 56
[2,] 86 42 96 92
[3,] 90 60 38 84
[4,] 80 76 82 24
[5,] 46 99 99 99
[6,] 99 99 36 99
[7,] 26 99 99 20
[8,] 99 99 48 34

```

[Execution complete with exit code 0]

iv. Concatenate vertically and split into equal parts

```

combined <- rbind(B, C)
cat("\nCombined matrix:\n"); print(combined)

```

3. Create a structured NumPy array to store employee records with attributes: Employee_ID (integer), Name (string, max length 15), Department (string, max length 10), and Salary (float).

The screenshot shows the RStudio interface with two panes: Program input and Output.

```

1 print("K.Nichal haas RegNo:22BCE9651")
2 #3. Create a structured NumPy array to store employee records with attributes: Employee_ID (integer), Name (string, max length 15), Dep
3 # Create structured employee data frame
4 employees <- data.frame(
5   Employee_ID = c(101, 102, 103, 104),
6   Name = c("Alice", "Bob", "Charlie", "David"),
7   Department = c("HR", "IT", "Finance", "IT_Manager"),
8   Salary = c(70000, 80000, 65000, 90000)
9 )
10 cat("Original Employee Data:\n")
11 print(employees)

```

Output

```

[1] "K.Nichal haas RegNo:22BCE9651"
Original Employee Data:
  Employee_ID     Name Department Salary
1          101    Alice        HR 70000
2          102     Bob        IT 80000
3          103  Charlie    Finance 65000
4          104  David  IT_Manager 90000

```

[Execution complete with exit code 0]

```

# Create structured employee data frame
employees <- data.frame(
  Employee_ID = c(101, 102, 103, 104),
  Name = c("Alice", "Bob", "Charlie", "David"),
  Department = c("HR", "IT", "Finance", "IT_Manager"),
  Salary = c(70000, 80000, 65000, 90000)
)
cat("Original Employee Data:\n")
print(employees)

```

i. Add a new employee record to the array.

The screenshot shows the RStudio interface with two panes: Program input and Output.

```

1 print("K.Nichal haas RegNo:22BCE9651")
2 #3. Create a structured NumPy array to store employee records with attributes: Employee_ID (integer), Name (string, max length 15), Dep
3 # Create structured employee data frame
4 employees <- data.frame(
5   Employee_ID = c(101, 102, 103, 104),
6   Name = c("Alice", "Bob", "Charlie", "David"),
7   Department = c("HR", "IT", "Finance", "IT_Manager"),
8   Salary = c(70000, 80000, 65000, 90000)
9 )
10 cat("Original Employee Data:\n")
11 print(employees)
12
13 # i. Add new employee record
14 new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
15 employees <- rbind(employees, new_emp)
16 cat("\nAfter adding new employee:\n")
17 print(employees)

```

Output

```

[1] "K.Nichal haas RegNo:22BCE9651"
Original Employee Data:
  Employee_ID     Name Department Salary
1          101    Alice        HR 70000
2          102     Bob        IT 80000
3          103  Charlie    Finance 65000
4          104  David  IT_Manager 90000

```

After adding new employee:

Employee_ID	Name	Department	Salary
101	Alice	HR	70000
102	Bob	IT	80000
103	Charlie	Finance	65000
104	David	IT_Manager	90000
105	Eve	Manager	95000

[Execution complete with exit code 0]

```

# i. Add new employee record
new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
employees <- rbind(employees, new_emp)
cat("\nAfter adding new employee:\n")
print(employees)

```

ii. Extract all employees whose salary is above 75,000.

```

1 print("K.Nichal has RegNo:22BCE9651")
2 #3. Create a structured NumPy array to store employee records with attributes: Employee_ID (integer), Name (string, max length 15), Dep
3 # Create structured employee data frame
4 employees <- data.frame(
5   Employee_ID = c(101, 102, 103, 104),
6   Name = c("Alice", "Bob", "Charlie", "David"),
7   Department = c("HR", "IT", "Finance", "IT_Manager"),
8   Salary = c(70000, 80000, 65000, 90000)
9 )
10 cat("Original Employee Data:\n")
11 print(employees)
12
13 # i. Add new employee record
14 new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
15 employees <- rbind(employees, new_emp)
16 cat("\nAfter adding new employee:\n")
17 print(employees)
18
19 # ii. Extract employees with salary > 75,000
20 high_salary <- subset(employees, Salary > 75000)
21 cat("\nEmployees with Salary > 75,000:\n")
22 print(high_salary)

```

Program input

Output

	Employee_ID	Name	Department	Salary
2	102	Bob	IT	80000
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000

After adding new employee:

	Employee_ID	Name	Department	Salary
1	101	Alice	HR	70000
2	102	Bob	IT	80000
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

Employees with Salary > 75,000:

	Employee_ID	Name	Department	Salary
2	102	Bob	IT	80000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

[Execution complete with exit code 0]

```

high_salary <- subset(employees, Salary > 75000)
cat("\nEmployees with Salary > 75,000:\n")
print(high_salary)

```

iii. Sort the array based on Employee_ID in ascending order.

```

1 print("K.Nichal has RegNo:22BCE9651")
2 #3. Create a structured NumPy array to store employee records with attributes: Employee_ID (integer), Name (string, max length 15), Dep
3 # Create structured employee data frame
4 employees <- data.frame(
5   Employee_ID = c(101, 102, 103, 104),
6   Name = c("Alice", "Bob", "Charlie", "David"),
7   Department = c("HR", "IT", "Finance", "IT_Manager"),
8   Salary = c(70000, 80000, 65000, 90000)
9 )
10 cat("Original Employee Data:\n")
11 print(employees)
12
13 # i. Add new employee record
14 new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
15 employees <- rbind(employees, new_emp)
16 cat("\nAfter adding new employee:\n")
17 print(employees)
18
19 # ii. Extract employees with salary > 75,000
20 high_salary <- subset(employees, Salary > 75000)
21 cat("\nEmployees with Salary > 75,000:\n")
22 print(high_salary)
23
24 # iii. Sort by Employee_ID (ascending)
25 sorted_emp <- employees[order(employees$Employee_ID), ]
26 cat("\nSorted by Employee_ID:\n")
27 print(sorted_emp)

```

Program input

Output

	Employee_ID	Name	Department	Salary
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

Employees with Salary > 75,000:

	Employee_ID	Name	Department	Salary
2	102	Bob	IT	80000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

Sorted by Employee_ID:

	Employee_ID	Name	Department	Salary
1	101	Alice	HR	70000
2	102	Bob	IT	80000
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

[Execution complete with exit code 0]

iii. Sort by Employee_ID (ascending)

```

sorted_emp <- employees[order(employees$Employee_ID), ]
cat("\nSorted by Employee_ID:\n")
print(sorted_emp)

```

iv. Increase the salary of employees in the 'IT' department by 10%.

```

3 # Create structured employee data frame
4 employees <- data.frame(
5   Employee_ID = c(101, 102, 103, 104),
6   Name = c("Alice", "Bob", "Charlie", "David"),
7   Department = c("HR", "IT", "Finance", "IT_Manager"),
8   Salary = c(70000, 80000, 65000, 90000)
9 )
10 cat("Original Employee Data:\n")
11 print(employees)
12
13 # i. Add new employee record
14 new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
15 employees <- rbind(employees, new_emp)
16 cat("\nAfter adding new employee:\n")
17 print(employees)
18
19 # ii. Extract employees with salary > 75,000
20 high_salary <- subset(employees, Salary > 75000)
21 cat("\nEmployees with Salary > 75,000:\n")
22 print(high_salary)
23
24 # iii. Sort by Employee_ID (ascending)
25 sorted_emp <- employees[order(employees$Employee_ID), ]
26 cat("\nSorted by Employee_ID:\n")
27 print(sorted_emp)
28
29 # iv. Increase salary of IT department employees by 10%
30 employees$Salary[employees$Department == "IT"] <-
31   employees$Salary[employees$Department == "IT"] * 1.10
32 cat("\nAfter 10% raise for IT department:\n")
33 print(employees)

```

Program input

Output

	Employee_ID	Name	Department	Salary
5	105	Eve	Manager	95000

Sorted by Employee_ID:

	Employee_ID	Name	Department	Salary
1	101	Alice	HR	70000
2	102	Bob	IT	80000
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

After 10% raise for IT department:

	Employee_ID	Name	Department	Salary
1	101	Alice	HR	70000
2	102	Bob	IT	88000
3	103	Charlie	Finance	65000
4	104	David	IT_Manager	90000
5	105	Eve	Manager	95000

[Execution complete with exit code 0]

iv. Increase salary of IT department employees by 10%

```

employees$Salary[employees$Department == "IT"] <-
  employees$Salary[employees$Department == "IT"] * 1.10
cat("\nAfter 10% raise for IT department:\n")
print(employees)

```

v. Retrieve the names of employees who have 'Manager' in their department.

```

8   Salary = c(70000, 80000, 65000, 90000)
9 }
10 cat("Original Employee Data:\n")
11 print(employees)
12
13 # i. Add new employee record
14 new_emp <- data.frame(Employee_ID = 105, Name = "Eve", Department = "Manager", Salary = 95000)
15 employees <- rbind(employees, new_emp)
16 cat("\nAfter adding new employee:\n")
17 print(employees)
18
19 # ii. Extract employees with salary > 75,000
20 high_salary <- subset(employees, Salary > 75000)
21 cat("\nEmployees with Salary > 75,000:\n")
22 print(high_salary)
23
24 # iii. Sort by Employee_ID (ascending)
25 sorted_emp <- employees[order(employees$Employee_ID), ]
26 cat("\nSorted by Employee_ID:\n")
27 print(sorted_emp)
28
29 # iv. Increase salary of IT department employees by 10%
30 employees$Salary[employees$Department == "IT"] <-
31 employees$Salary[employees$Department == "IT"] * 1.10
32 cat("\nAfter 10% raise for IT department:\n")
33 print(employees)
34
35 # v. Retrieve names of employees who have 'Manager' in their department
36 manager_names <- employees>Name[grep("Manager", employees$Department)]
37 cat("\nEmployees with 'Manager' in Department:\n")
38 print(manager_names)

```

Program input

Employee_ID	Name	Department	Salary
1	Alice	HR	70000
2	Bob	IT	80000
3	Charlie	Finance	65000
4	David	IT_Manager	90000
5	Eve	Manager	95000

Output

After 10% raise for IT department:

Employee_ID	Name	Department	Salary
1	Alice	HR	70000
2	Bob	IT	88000
3	Charlie	Finance	65000
4	David	IT_Manager	90000
5	Eve	Manager	95000

Employees with 'Manager' in Department:

[1] "David" "Eve"

[Execution complete with exit code 0]

v. Retrieve names of employees who have 'Manager' in their department

```

manager_names <- employees>Name[grep("Manager", employees$Department)]
cat("\nEmployees with 'Manager' in Department:\n")
print(manager_names)

```