Day2.R

RKC

2024-10-26

```
#### Continued part of DataFrame ####

x = c(NA, 2,3)
y = c(6,7,8)
z = c(10,11,NA)

d7 = data.frame(x=x, y=y, z=z);d7
```

```
##    x y  z
## 1 NA 6 10
## 2  2 7 11
## 3  3 8 NA
```

```
# brings out values in Bool
is.na(d7)
```

```
##          x     y     z
## [1,]  TRUE FALSE FALSE
## [2,] FALSE FALSE FALSE
## [3,] FALSE FALSE  TRUE
```

```
#brings out values in Bool (row level)
is.na(d7$x)
```

```
## [1]  TRUE FALSE FALSE
```

```
# Bring out count of values NA present in
sum(is.na(d7$x))
```

```
## [1] 1
```

```
#[1] 1
sum(d7)
```

```
## [1] NA
```

```
sum(d7, na.rm=T)
```

```
## [1] 47
```

```r
cleaned_data = na.omit(d7); cleaned_data
```

```
##   x y  z
## 2 2 7 11
```

```r
sum(cleaned_data)
```

```
## [1] 20
```

```r
#### DATA IMPORTING ####

data()  # Bring outs datasets available with RStudio
```

```r
# Dataset imported
Countries_Population

head(Countries_Population)  # importing Dataset
head(Countries_Region_Mapping)

# Gets working Directory
getwd() # [1] "C:/Users/RKC/OneDrive/Documents"

# Sets current working directory - Don't forget to change backslash's to forwardslash
setwd("A:/CDAC_SM_VITA/5_R_Programming/Day2/")

# Verfying  above
getwd()

# #-----------------READ TABLE(.txt)-----------------

# Reading table read.table()- to import .txt files
titanic = read.table("Titanic_space_separated.txt", header = TRUE); titanic

# SEPARATOR - helps separate columns using delimiter
orange = read.table('Orange_comma_separated - Copy.txt', header = TRUE, sep = ',')
View(orange)
```

```r
countries = read.csv("Countries Population.csv")
countries
```

```r
# install.packages('readxl') - to download excel package
library(readxl)

## read_excel()

countriesregion = read_excel("Countries Region Mapping.xlsx"); countriesregion
??read_excel

# using sheet to read diff sheets available in workbook
countriesregion2 =  read_excel('Countries Region Mapping.xlsx', sheet = "Sheet1"); countriesregion2
```

```
# install.packages('openxlsx')
# library(openxlsx)
# read.xlsx()
```

```
# Exporting in CSV
data = iris
View(data)

?write
#write(x, file = "data",
#       ncolumns = if(is.character(x)) 1 else 5,
#       append = FALSE, sep = " ")

#Write.csv function
write.csv(data, file = 'Iris.csv')

#Write.table function
write.table(data, 'export1.txt')

write.table(data, 'export2.txt', sep = "@") #can provide any sep value
```

```
library(randomNames)
Employee = data.frame(Name = randomNames(10), Age = sample(20:40,10),
                      Salary = sample(10000:75000, 10)); Employee

write.csv(Employee, 'Employee.csv')
#write.excel(Employee, 'Employee.xlsx')
#write

#### LISTS ####
#Create a lists containing strings, numbers, vectors and a logical

list_data = list('red', 'green',c(21,32,11), TRUE, 51.23, 119.31)

list_data
length(list_data)
print(list_data)


list_data = list(I_Quarter=c('Jan', 'Feb', 'Mar'),
                 A_Matrix = matrix(c(3,9,5,1,-2,8), nrow=2),
                 A_Inner_list = list('green', 12.3)); list_data

names(list_data)
list_data$I_Quarter
list_data$A_Matrix
list_data$A_Matrix[1,]
list_data$A_Matrix[1,1]
list_data$A_Inner_list
list_data[1]
list_data[1][1]
list_data[1][2] # Won't work as
```

```
list_data[[1]][1]
#[1] "Jan"
list_data[[1]][2]
#[1] "Feb"
list_data[[1]][3]
#[1] "Mar"

# Subsetting elements of matrix inside a list
list_data[[2]]
list_data[[2]][1,2]
list_data[[2]][,3]
list_data[[2]][,1]

# Subsetting inner list

list_data[[3]]
list_data[[3]][1]
list_data[[3]][2]
list_data[[3]][3]


# passing a vector inside a list whcih is embedded in another list
list_data1 = list(I_Quarter=c('Jan', 'Feb', 'Mar'),
                  A_Matrix = matrix(c(3,9,5,1,-2,8), nrow=2),
                  A_Inner_list = list(c('red', 'green'), 12.3)); list_data1

# Subsetting values at 3rd level of list
list_data1[[3]]
list_data1[[3]][[1]][1]

# Adding and removing elements from list

#removing
list_data[-1]

list_data[4] = 'new Element'; list_data
length(list_data)

list_data[4] = NULL; list_data
length(list_data)

print(list_data[4])

list_data[[3]][3] = 'Pranay Shah';list_data

list_data[3] = 'Updated List'; list_data

# create two list
list1 = list(1,2,3)
list2 = list('Sun', 'Mon', 'Tue')

mergedlist = c(list1, list2); mergedlist
```

```
class(mergedlist)

# APPEND

li = list('Java', 'Python');li
li2 = append(li, 2); li2
li3 = append(li2, 2:3); li3

# ?append

# Adding new element at specified position

li = list('Java', 'Python', 'C'); li

li2 = append(li, "R", after = 2); li2

new = 1:21

li3 = append(li2, new); li3

#### ARRAYS ####

# Arrays are the R Data object which can stire the data in more than two dimesnions,
# for example - if ee craete an array of dimension

# (2,3,4) then it craetes 4 rectangular matrices each with 2 rows and 3 columns

# An array is created using the array() function.
# It takes vectors as input and uses the values in the dim parameter

#?array

# create two vectors of diff length
vector1 = c(5,9,3)
vector2 = c(10,11,12,13,14,15)

result = array(c(vector1, vector2), dim = c(3,3,2))
print(result)

# create a vector of diff length

vector1 = c(5,9,3)
vector2 = c(10,11,12,13,14,15)

column.names = c("COL1", "COL2", "COL3")
row.names = c("ROW1", "ROW2", "ROW3")
matrix.names = c('Matrix1', 'Matrix2')

# take these vectors as input to arrays

result = array(c(vector1, vector2),
               dim = c(3,3,2),
               dimnames = list(row.names, column.names, matrix.names))
```

```
print(result)

# Element in third row of second matrix
result[3,,2]

# 2nd matrix
result[,,2]

# access Element from 1st matrix 2nd row and 2nd matrix 2nd row


# create a matrix 4*4*4 array "arr" filled with numbers from 1:64

?matrix
v1 = c(1:64)

column.names = c("COL1", "COL2", "COL3", "COL4")
row.names = c("ROW1", "ROW2", "ROW3", "ROW4")
matrix.names = c('Matrix1', 'Matrix2', 'Matrix3', 'Matrix4')

result = array(v1, dim = c(4,4,4))
print(result)

# replace all values in fourth layer with zeros
result[,,4] = 0 ; result

# Extract a 2*2*2 sub-array from the two-left corner of "arr"
result[1:2,1:2,1:2]

result[1:2,1:2,c(1,3)]

result[1:2,1:2,1]


#### FACTORS ####


# Factors are th data objects which are used to categorise the data and store
# it as levels, they can store both strings and integers
# They are useful in the columns whch have a limited numebr of unique values
# Like 'Male', 'Female' and TRUE, FALSE, etc. They are useful in data analysis for
# Stats modelling


data = c('East', "West", 'East',
          'North', 'North', 'East', 'West', 'West', 'West', 'East',
         'North'); data
unique(data)
is.factor(data)
#?factor
factor_data = factor(data); factor_data
class(factor_data)
is.factor(factor_data)
```

```
# Apply the levels of factor as per user defined required order

new_order_data = factor(factor_data, levels = c("East", 'West', 'North')); new_order_data
factor_data



?gl
# gl(n, k, length = n*k, labels = seq_len(n), ordered = FALSE)

v = gl(3,4,labels = c('Tampa', 'Seattle','Boston')); v

# Throws Error
v1 = gl(3,4, labels = c("Thane", "Borivali")); v1


#### NPUTS ####

# Taking inputs from users
var1 = as.numeric(readline("Enter first number :: ")); var1


{
  var1 = as.numeric(readline("Enter first number :: "))
  var2 = as.numeric(readline("Enter first number :: "))
  var3 = as.numeric(readline("Enter first number :: "))
  var4 = as.numeric(readline("Enter first number :: "))
}

# SCAN FUNCTION

?scan
# To get int  as input
x = scan(); x
class(x)
# To get characters a input
y = scan(what = character()); y

m = matrix(scan(nmax=9), 3,3); m

casefold(y, upper=FALSE)
casefold(y, upper = TRUE)
toupper(y)
tolower(y)
class(y)

# Create a vector by accepting age as an input from 5 of your classmates




# Ask teh user to enter the number of rows and columns for a matrix
```

```
# take element for each row using
nrow= scan(nmax = 1)
ncol = scan(nmax = 1)
# v = c(age = as.numeric(scan(nmax=5)), name = (scan(nmax=5)))
matrix(v, nrow, ncol)

# Use scan to take a 10 numeric input from users, representing their test scores
# Store thes escores in vectro scores
# Calculate and print the average of the scores
# Count and print how many scores are above 80



scores = c(scan()); scores
mean(scores)
above80 = scores[scores>80]; above80

cat(above80, "Scores are above 80")
print(paste(above80, "Scores are above 80"))



?cat# concat and print
?paste#    concat
```