Week\_5\_lab

Joseph

2024-07-07

#1.Lists  
# Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it.   
# A list can also contain a matrix or a function as its elements. List is created using list() function.  
# # - Creating a list: list(element1, element2, ...)  
student\_list <- list(name = "John", age = 20, grades = c(85, 90, 95))  
# - Accessing list elements: list\_name$element\_name or list\_name[[index]]  
print(student\_list$name)

## [1] "John"

print(student\_list[[2]])

## [1] 20

# - Modifying list elements: list\_name$element\_name <- new\_value or list\_name[[index]] <- new\_value  
student\_list$age <- 21  
print(student\_list$age)

## [1] 21

#4. Data Frames:  
# Data Frames are data displayed in a format as a table.  
# Data Frames can have different types of data inside it.  
#While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.  
  
#Use the data.frame() function to create a data frame:  
  
# Creating a data frame: data.frame(column1, column2, ...)  
student\_data <- data.frame(name = c("Alice", "Bob", "Charlie"), age = c(18, 19, 20))  
  
# Accessing data frame columns: data\_frame\_name$column\_name or data\_frame\_name[["column\_name"]]  
print(student\_data$name)

## [1] "Alice" "Bob" "Charlie"

print(student\_data[["age"]])

## [1] 18 19 20

# Adding a new column to a data frame: data\_frame\_name$new\_column\_name <- new\_column\_data  
student\_data$new\_column <- c(90, 85, 95)  
print(student\_data)

## name age new\_column  
## 1 Alice 18 90  
## 2 Bob 19 85  
## 3 Charlie 20 95

#5. Arrays:  
# Arrays are the R data objects which can store data in more than two dimensions. For example  
# If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type.  
# An array is created using the array() function.   
#It takes vectors as input and uses the values in the dim parameter to create an array.  
# Creating an array: array(data, dim = c(rows, columns))  
student\_array <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3))  
# Printing the array  
print(student\_array)

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

# Accessing array elements: array\_name[row\_index, column\_index]  
print(student\_array[1, 2])

## [1] 3

# 2. Tuples:  
# R tuple is basically an entity containing items belonging to different data types. The integral values can be clubbed with the Boolean or the string variables under one parameter in R programming language  
student\_tuple <- tuple(name = "Mary", age = 22, Gender = "Female")  
  
print(student\_tuple)

## (name = "Mary", age = 22, Gender = "Female")

print(student\_tuple$name)

## [1] "Mary"

#Accessing tuple elements: tuple\_name$element\_name or tuple\_name[[index]]  
print(student\_tuple[[2]])

## [1] 22

print(student\_tuple[2])

## (age = 22)

print(student\_tuple$Gender)

## [1] "Female"

#3. Sets:  
# A set in mathematics is defined as the collection of unique elements and the   
# order of the elements does not matter.   
  
# Creating a set: set(element1, element2, ...)  
student\_set <- set("Alice", "Bob", "Charlie")  
  
# Adding elements to a set: set\_name <- union(set\_name, new\_element)  
student\_set <- union(student\_set, "Dave")  
# Printing the sets  
print(student\_set)

## [[1]]  
## [1] "Alice"  
##   
## [[2]]  
## [1] "Bob"  
##   
## [[3]]  
## [1] "Charlie"  
##   
## [[4]]  
## [1] "Dave"