- write code to create a matrix, data frame, list, array

  vect1 <- c(10, 20, 30, 40)
  vect2 <- c(50, 60, 70, 80, 90, 100)

  Array
  B <- array(c(vect1, vect2), dim = c(3, 3, 2))

  Matrix
  matrix_c <- matrix(vect1, byrow = FALSE, ncol = 3)

  DataFrame – multi datatype
  df <- data.frame(vect1, vest2, stringAsFactor = False)

  List – multi datatype
  a <- list(1,2,4,2,4,5,2,4)

- explain functions that examine features of scalars, vectors

  Scalars
  In computer programming, scalar refers to an **atomic quantity** that can **hold only one value at a time**. Scalars are the most basic data types that can be used to construct more complex ones
  > x <- 1
  > y <- 2.5

  Vector
  A vector is a sequence of data elements of the same basic type.
  p <- c("one","two","three","four","five","six") # Character vector

- Write program in R showing elements in a character vector:
  "paul"=1, "jones"=2, "tom"=3

  a1 <- c("paul"="1", "jones"="2", "tom"="3"); a1; class(a1)

- explain the relational operators(<,>,<=,>=), arithmetic operators(+,-,*,/), logical (&&&,|,||) operators

  relational
  < less than
  > greater than
  <= less than or equal to
  >= greater than or equal to
  arithmetric
  + addition
  - subtraction
  * multiplication
  / division
  Logical
  c(1, 2, 3) & c(2,3,3)
  c(1, 2, 3) && c(2,3,3)

  & logical AND - TRUE TRUE TRUE
  && logical AND (short circuits on the first element) TRUE
  | logical OR - TRUE TRUE TRUE
  || logical OR (short circuits on the first element) TRUE

- Explain NULL, NA and NaN
  NULL

  NA

  NaN

- write program to show Slicing, Transpose , Diagonal, Identity
- write program to create array with matrix, with vectors. show change of names

```
# Create two vectors of different lengths.
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 the array.
result <- array(c(vector1, vector2),dim = c(3,3,2), dimnames =
list(row.names,column.names, matrix.names))
print(result)
```

- write program that read input and calculate factorial

```
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
factorial = 1
# check is the number is negative, positive or zero
if(num < 0) {
  print("Sorry, factorial does not exist for negative numbers")
} else if(num == 0) {
  print("The factorial of 0 is 1")
} else {
  for(i in 1:num) {
    factorial = factorial * i
  }
  print(paste("The factorial of", num ,"is",factorial))
}
```

- show binding in dataframe

```
# Install dplyr package
install.packages("dplyr")

# Load dplyr package
library("dplyr")

# Create three data frames
data1 <- data.frame(x1 = 1:5,
                    x2 = letters[1:5])
data2 <- data.frame(x1 = 0,
                    x3 = 5:9)
data3 <- data.frame(x3 = 5:9,
                    x4 = letters[5:9])

# Apply bind_rows function
bind_rows(data1, data2, id = NULL)
```

- create table

```
trial <- matrix(c(34,11,9,32), ncol=2)
colnames(trial) <- c('sick', 'healthy')
rownames(trial) <- c('risk', 'no_risk')
trial <- as.table(trial);trial
```

- explain time series, recycling
- explain apply(), sapply(), lapply(), tapply(), mapply()

```
rep.int(x, times) replicate x by no of times
```

apply() takes Data frame or matrix as an input and gives output in vector, list or array. apply() Function is primarily used to avoid explicit uses of loop constructs. It is the most basic of all collections can be used over a matrice.

```
m1 <- matrix(C<-(1:10),nrow=5, ncol=6)
m1
a_m1 <- apply(m1, 2, sum)
a_m1
```

```
   [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6    1    6    1    6
[2,]    2    7    2    7    2    7
[3,]    3    8    3    8    3    8
[4,]    4    9    4    9    4    9
[5,]    5   10    5   10    5   10
> a_m1 <- apply(m1, 2, sum)
> a_m1
[1] 15 40 15 40 15 40
```

lapply() function is useful for performing operations on list objects and returns a list object of same length of original set. lappy() returns a list of the similar length as input list object, each element of which is the result of applying FUN to the corresponding element of list. lapply() takes list, vector or data frame as input and gives output in list.

```
movies <- c("SPYDERMAN","BATMAN","VERTIGO","CHINATOWN")
movies_lower <-lapply(movies, tolower)
str(movies_lower)
```

```
List of 4
 $ : chr "spyderman"
 $ : chr "batman"
 $ : chr "vertigo"
 $ : chr "chinatown"
```

sapply() function takes list, vector or data frame as input and gives output in vector or matrix. It is useful for operations on list objects and returns a list object of same length of original set. sapply() function does the same job as lapply() function but returns a vector.\

```
dt <- cars
lmn_cars <- lapply(dt, min)
smn_cars <- sapply(dt, min)
lmn_cars
```

```
> lmn_cars <- lapply(dt, min)
> lmn_cars
$speed
[1] 4

$dist
[1] 2

> dt <- cars
> smn_cars <- sapply(dt, min)
> smn_cars
speed  dist
   4     2
```

tapply() computes a measure (mean, median, min, max, etc..) or a function for each factor variable in a vector. It is a very useful function that lets you create a subset of a vector and then apply some functions to each of the subset.

```
data(iris)
tapply(iris$Sepal.Width, iris$Species, median)
```

```
tdata <- as.data.frame(cbind(c(1,1,1,1,1,2,2,2,2,2), my.matrx))
colnames(tdata)
## [1] "V1" "V2" "V3" "V4"
Now let's use column 1 as the index and find the mean of column 2
tapply(tdata$V2, tdata$V1, mean)
## 1 2
## 3 8
```

mapply() function is **a multivariate apply**() of sorts which applies a function in parallel over a set of arguments. lapply()iterate over a single R object but What if you want to iterate over multiple R objects in parallel then mapply() is the function for you

```
Q2 <- mapply(rep,1:4,4)
# Print `Q2`
print(Q2)
```

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

	Function	Arguments	Objective	Input	Output
	apply	apply(x, MARGIN, FUN)	Apply a function to the rows or columns or both	Data frame or matrix	vector, list, array
	lapply	lapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	list
	sapply	sappy(X FUN)	Apply a function to all the elements of the input	List, vector or data frame	vector or matrix

- list characteristic of dataframe

  The column names should be non-empty.
  The row names should be unique.
  The data frame can hold the data which can be a numeric, character or of factor type.
  Each column should contain the same number of data items

- explain abs(),sqrt(), ceiling(), floor(), round() with program

  abs()
  returns the absolute value for the number provided as an input.

```
> abs(-10)
[1] 10
> p <- c(-12, -14, -16, -18)
> abs(p)
[1] 12 14 16 18
```

  Sqrt()
  square root of a positive real number

```
p <- sqrt(12.28)
> print(p)
[1] 3.504283
```

  Ceiling()
  truncate the given float value to a nearest integer

```
ex_1 <- ceiling(23.48)
> print(ex_1)
[1] 24

> ex_2 <- ceiling(-18.25)
> print(ex_2)
[1] -18
```

  Floor()
  opposite to the **ceiling()**

```
ex_1 <- floor(23.48)
> print(ex_1)
[1] 23

> ex_2 <- floor(-18.25)
> print(ex_2)
[1] -19
```

Round()
round up (ceiling() function) and round down (floor() function) any given real number

```
> ex_1 <- round(23.55)
> print(ex_1)
[1] 24

> ex_2 <- round(-18.25)
> print(ex_2)
[1] -18
```

- explain S3 &S4

– S3 is used to overload any function. Therefore, we can call different names of the function. And, it depends upon the type of input parameter or the number of a parameter.

– An important characteristic of OOP is S4. However, it poses a limitation as it is quite tricky to debug. An alternate for the S4 is the reference class.

- S3 class is somewhat primitive in nature. It lacks a formal definition and object of this class can be created simply by adding a class attribute to it.
- most of the R built-in classes are of this type.
- With the help of the S3 class, you can avail its ability to implement
- easier to implement
- In S3 class, the generic function makes the call to the method. S3 is very casual and does not have any formal definition of classes.
- S3 requires very less knowledge on the part of the programmer

- S3 class has no formal, predefined definition.

- Basically, a list with its class attribute set to some class name, is an S3 object. The components of the list become the member variables of the object.

- In R S3 system, it's pretty ad hoc. You can convert an object's class according to your will with objects of the same class looking completely different. It's all up to you.

- **S4 class** are an improvement over the S3 class. They have a formally defined structure which helps in making object of the same class look more or less similar.
- Class components are properly defined using the **setClass()** function and objects are created using the new() function.
- S4 Class is a bit similar to S3 but it is **more formal** than the latter. It differs from S3 in two different ways
- Firstly, in S4, there are **formalclass definitions** that provide description and representation of classes. Furthermore, it has special helper functions for defining methods and generics.
- S4 also **facilitates multiple dispatches.** This means that the generic functions are able to pick up methods based on the class comprising of multiple arguments.

- what does plot() do use example
  save plot as .png in directory
- what dev.off() do

  dev.off() shuts down the specified (by default the current) device. If the current device is shut down and any other devices are open, the next open device is made current. It is an error to attempt to shut down device 1.

- explain length(), nchar(), toupper(), tolower(), casefold(), chartr(), with program
-

`nchar()`	number of characters
`tolower()`	convert to lower case
`toupper()`	convert to upper case
`casefold()`	case folding
`chartr()`	character translation

```
chartr("a", "A", "This is a boring string")
#> [1] "This is A boring string"
```

```
casefold(x, upper = FALSE)
```

practical

- write program to create a matrix, data frame, list, array
- write program to show +,-,*,/ in array
- merge dataframe