Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE & COMMERCE (AUTONOMOUS)

OPTIMIZATION TECHNIQUE

Name: Somnath Mugdal
Roll No: 10437
Class: MSc Data Science and Artificial Intelligence part II (Semester III)

Ramniranjan Jhunjhunwala College of Arts, Science and Commerce

Department of Data Science and Artificial Intelligence

CERTIFICATE

This is to certify Mr. Somnath Mugdal of Msc. Data Science and Artificial Intelligence Roll No 10437 has successfully completed the practical of OPTIMIZATION TECHNIQUE during the Academic Year 2024-2025.

Date:

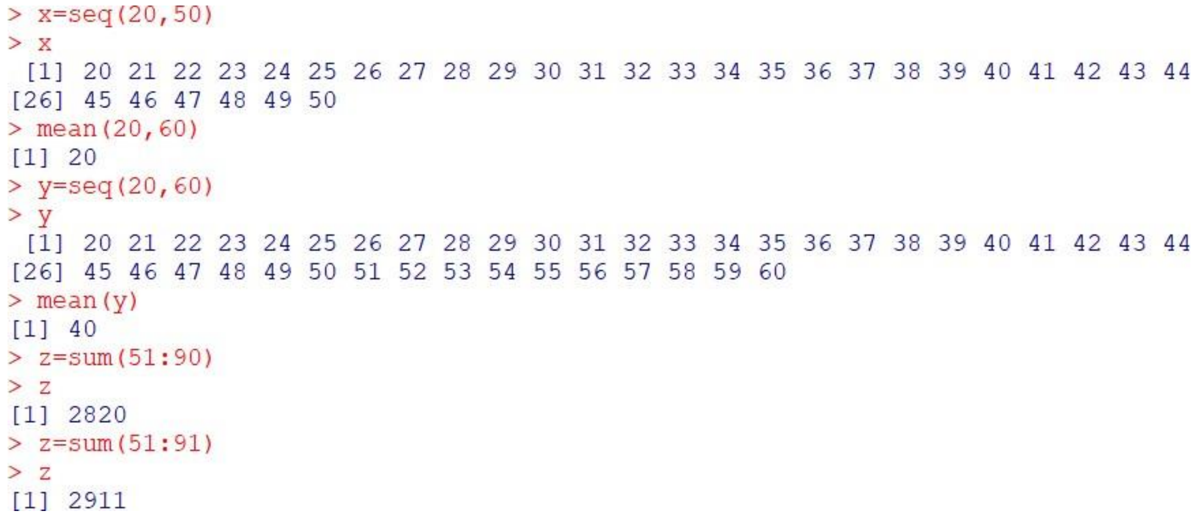
(Prof. Ashwini mam) **External Examiner**

Prof-In-Charge

INDEX

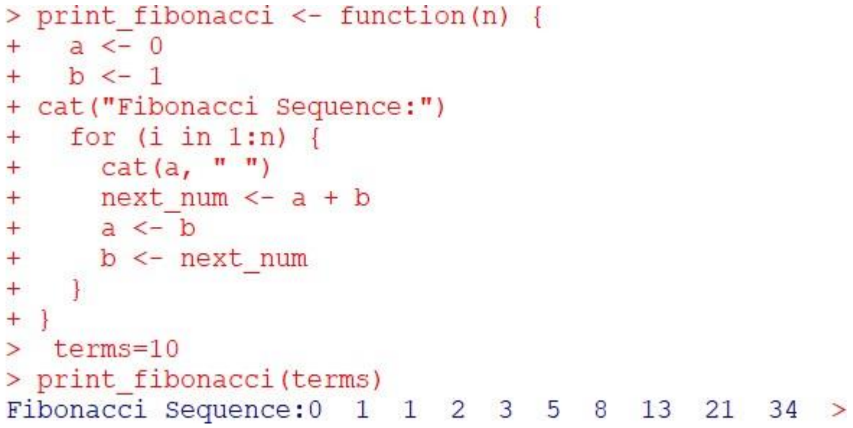
Practical 1: Basic R-Programming

1. write a R program to create sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 90



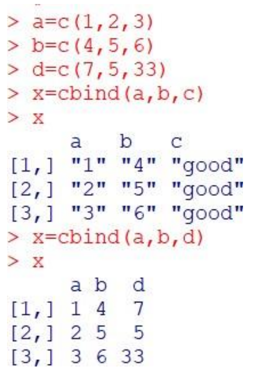
```
> x=seq(20,50)
> x
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
[26] 45 46 47 48 49 50
> mean(20,60)
[1] 20
> y=seq(20,60)
> y
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
[26] 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
> mean(y)
[1] 40
> z=sum(51:90)
> z
[1] 2820
> z=sum(51:91)
> z
[1] 2911
```

2. write r program to get l1st 10 fibonacci numbers



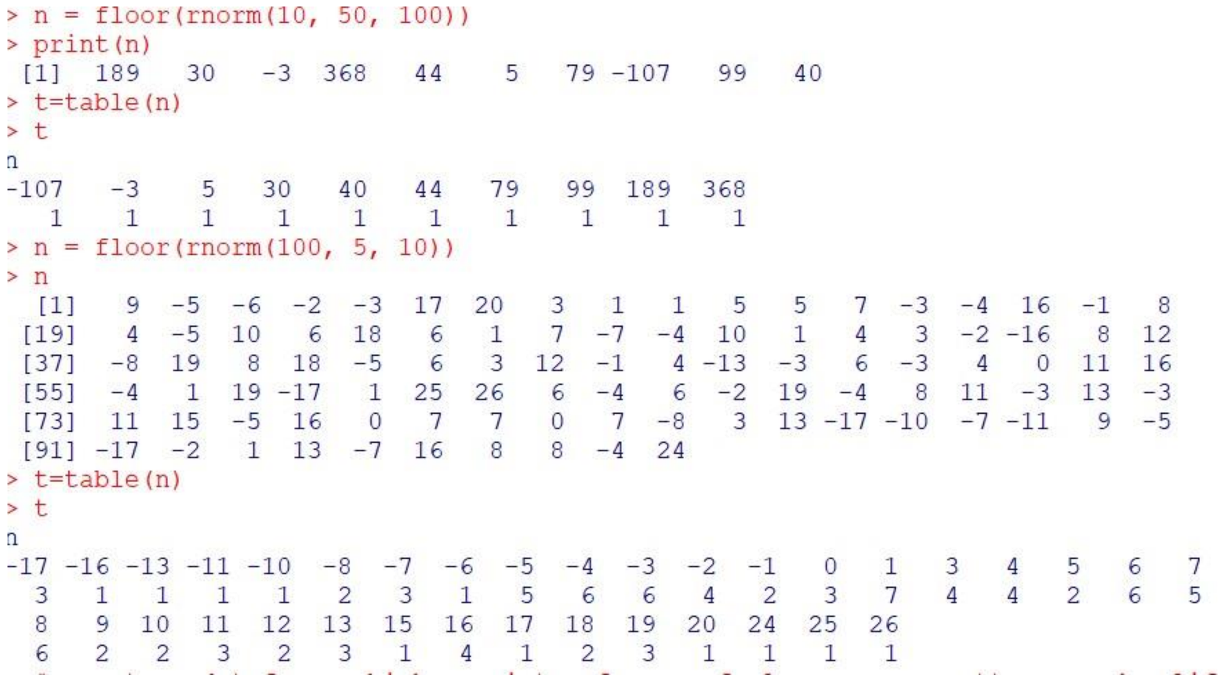
```
> print_fibonacci <- function(n) {
+   a <- 0
+   b <- 1
+ cat("Fibonacci Sequence:")
+   for (i in 1:n) {
+     cat(a, " ")
+     next_num <- a + b
+     a <- b
+     b <- next_num
+   }
+ }
>  terms=10
> print_fibonacci(terms)
Fibonacci Sequence:0  1  1  2  3  5  8  13  21  34  >
```

3. write a r program for 2 vector a b c with 3 integers combine 3 vectors to become 3x3 matrix where each column represent a vector print the required matrix



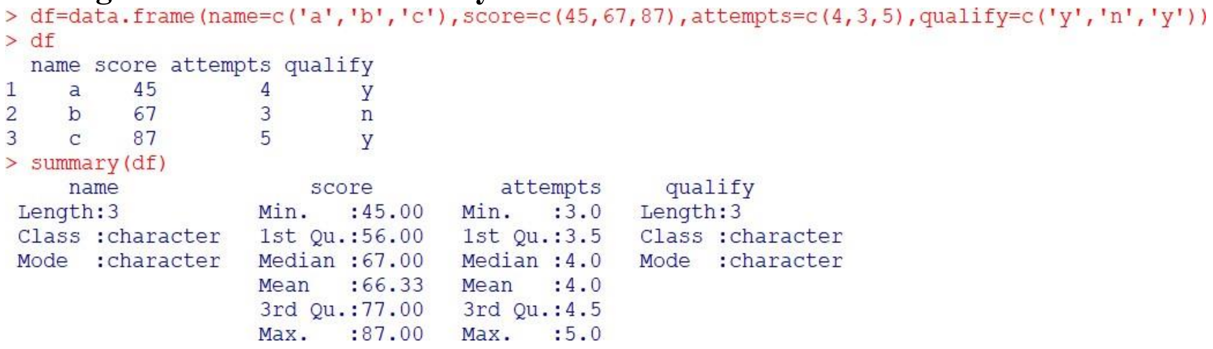
```
> a=c(1,2,3)
> b=c(4,5,6)
> d=c(7,5,33)
> x=cbind(a,b,c)
> x
     a   b   c
[1,] "1" "4" "good"
[2,] "2" "5" "good"
[3,] "3" "6" "good"
> x=cbind(a,b,d)
> x
     a b  d
[1,] 1 4  7
[2,] 2 5  5
[3,] 3 6 33
```

4. to create a list of random numbers in a normal distribution and count occurrence of each value



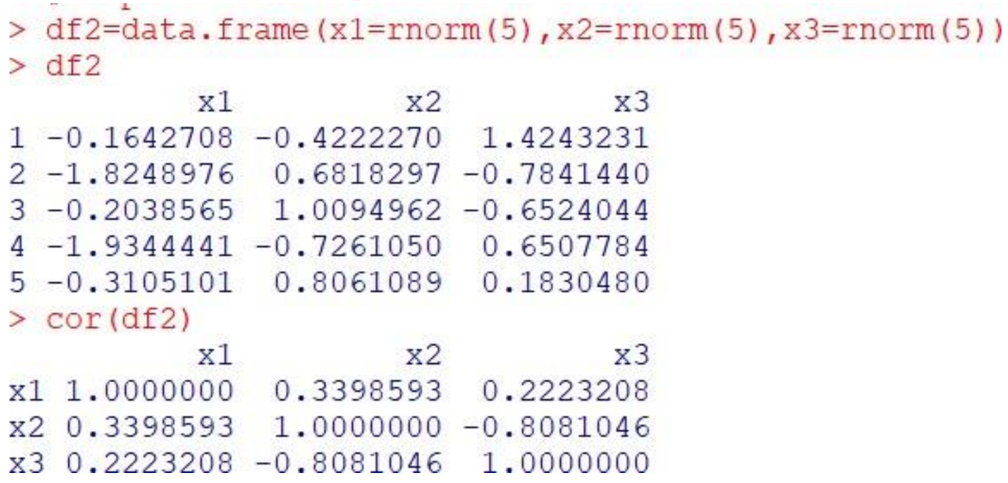
```
> n = floor(rnorm(10, 50, 100))
> print(n)
 [1]  189   30   -3  368   44    5   79 -107   99   40
> t=table(n)
> t
n
-107   -3    5   30   40   44   79   99  189  368 
   1    1    1    1    1    1    1    1    1    1 
> n = floor(rnorm(100, 5, 10))
> n
  [1]   9  -5  -6  -2  -3  17  20   3   1   1   5   5   7  -3  -4  16  -1   8
 [19]   4  -5  10   6  18   6   1   7  -7  -4  10   1   4   3  -2 -16   8  12
 [37]  -8  19   8  18  -5   6   3  12  -1   4 -13  -3   6  -3   4   0  11  16
 [55]  -4   1  19 -17   1  25  26   6  -4   6  -2  19  -4   8  11  -3  13  -3
 [73]  11  15  -5  16   0   7   7   0   7  -8   3  13 -17 -10  -7 -11   9  -5
 [91] -17  -2   1  13  -7  16   8   8  -4  24
> t=table(n)
> t
n
-17 -16 -13 -11 -10  -8  -7  -6  -5  -4  -3  -2  -1   0   1   3   4   5   6   7 
  3   1   1   1   1   2   3   1   5   6   6   4   2   3   7   4   4   2   6   5 
  8   9  10  11  12  13  15  16  17  18  19  20  24  25  26 
  6   2   2   3   2   3   1   4   1   2   3   1   1   1   1 
```

5. create a dataframe which consists of name of player score ,attempts and qualify find or get the statistical summary



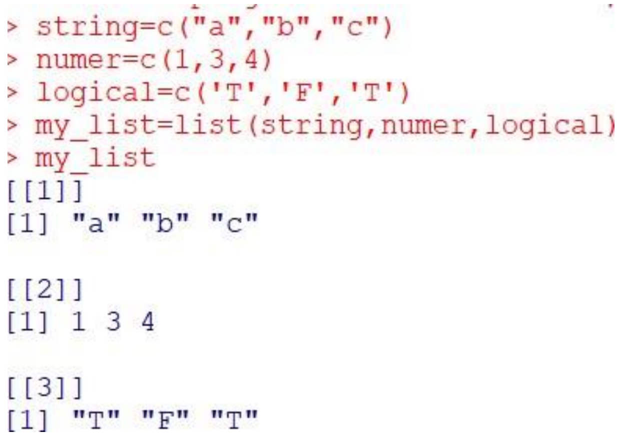
```
> df=data.frame(name=c('a','b','c'),score=c(45,67,87),attempts=c(4,3,5),qualify=c('y','n','y'))
> df
  name score attempts qualify
1    a    45        4       y
2    b    67        3       n
3    c    87        5       y
> summary(df)
     name               score          attempts      qualify         
 Length:3           Min.   :45.00   Min.   :3.0   Length:3          
 Class :character   1st Qu.:56.00   1st Qu.:3.5   Class :character  
 Mode  :character   Median :67.00   Median :4.0   Mode  :character  
                    Mean   :66.33   Mean   :4.0                     
                    3rd Qu.:77.00   3rd Qu.:4.5                     
                    Max.   :87.00   Max.   :5.0                     
```

6. creates a correlation matrix from dataframe which consists of x1,x2,and x3 and all this are random sample from normal distribution of count5



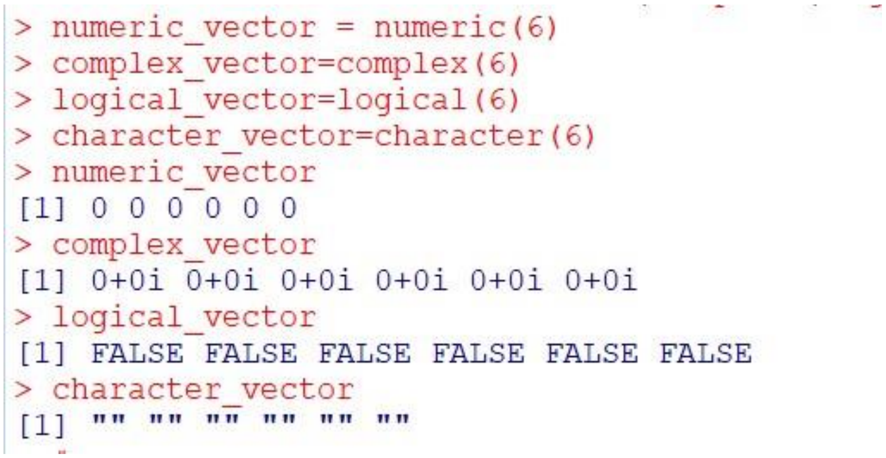
```
> df2=data.frame(x1=rnorm(5),x2=rnorm(5),x3=rnorm(5))
> df2
          x1         x2         x3
1 -0.1642708 -0.4222270  1.4243231
2 -1.8248976  0.6818297 -0.7841440
3 -0.2038565  1.0094962 -0.6524044
4 -1.9344441 -0.7261050  0.6507784
5 -0.3105101  0.8061089  0.1830480
> cor(df2)
          x1         x2         x3
x1 1.0000000  0.3398593  0.2223208
x2 0.3398593  1.0000000 -0.8081046
x3 0.2223208 -0.8081046  1.0000000
```

write R program to create list ,strings numbers and logical value



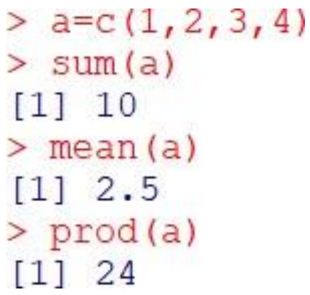
```
> string=c("a","b","c")
> numer=c(1,3,4)
> logical=c('T','F','T')
> my_list=list(string,numer,logical)
> my_list
[[1]]
[1] "a" "b" "c"

[[2]]
[1] 1 3 4

[[3]]
[1] "T" "F" "T"
```

to create a vector of numeric ,complex ,logical and character type of length of 6



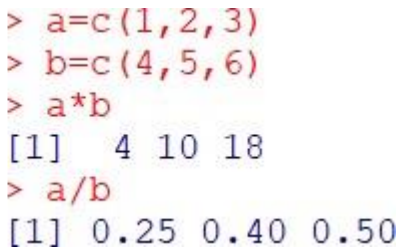
```
> numeric_vector = numeric(6)
> complex_vector=complex(6)
> logical_vector=logical(6)
> character_vector=character(6)
> numeric_vector
[1] 0 0 0 0 0 0
> complex_vector
[1] 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i
> logical_vector
[1] FALSE FALSE FALSE FALSE FALSE FALSE
> character_vector
[1] "" "" "" "" "" ""
```

9. Find the sum mean and product of vector



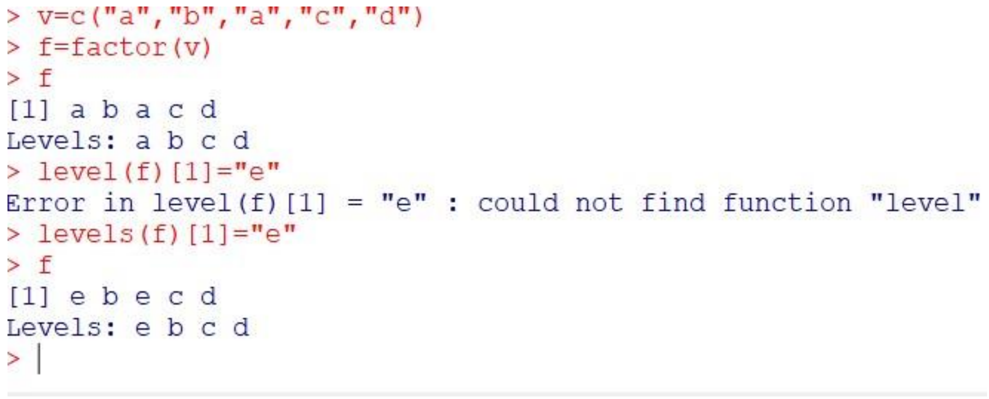
```
> a=c(1,2,3,4)
> sum(a)
[1] 10
> mean(a)
[1] 2.5
> prod(a)
[1] 24
```

10.Multiple by and divide two vectors of integers type and length 3



```
> a=c(1,2,3)
> b=c(4,5,6)
> a*b
[1]  4 10 18
> a/b
[1] 0.25 0.40 0.50
```

11.change the first level of vector with another level “e” of given factor



```
> v=c("a","b","a","c","d")
> f=factor(v)
> f
[1] a b a c d
Levels: a b c d
> level(f)[1]="e"
Error in level(f)[1] = "e" : could not find function "level"
> levels(f)[1]="e"
> f
[1] e b e c d
Levels: e b c d
> |
```

Signature: _______________

Practical 2: Write your first Matlab program and basic looping

1 Write your first Matlab program

```
a = 3; b
= 5; c =
a+b
```

Output:

```
8
```

2 The meaning of "a = b"

```
a = 3;
b = a;
b
```

Output:

```
3
```

3 Basic math operations

```
a = 3; b = 9; c = 2*a+b^2-
a*b+b/a-10
```

Output:

```
53
```

4 The meaning of "a = b", continued

```
a = 3; a
= a+1; a
```

Output:

```
4
```

5 Formatted output

```
fprintf('Hello')
```

Output:

```
Hello
```

6 Formatted output

```
a = 3; b =
a*a; c =
a*a*a; d =
sqrt(a);
fprintf('%4u square equals %4u \r', a, b) fprintf('%4u
cube equals %4u \r', a, c) fprintf('The square root of
%2u is %6.4f \r', a, d)
```

Output:

```
 3 square equals 9
 3 cube equals 27
 The square root of 3 is 1.7321
```

7 Arrays

```
a = [3 6 7];
b = [1 9 4];
c = a + b
```

Output:

```
4 15 11
```

8 Extracting an individual element of an array

```
a = [3 6 7]; b =
[1 9 4 5]; c =
a(2) + b(4)
```

Output:

```
c = 11
```

9 Comment

```
%
% This program demonstrates how to "comment out"
% a segment of code
%
A = 3;
B = A*A;
%
% B = 2*B <--- This statement is not executed
% C =

A+B
```

Output:

```
c = 12
```

10 Continuation to next line

```
summation1 = 1 + 3 + 5 + 7 ...
 + 9 + 11
```

11 "Clear" a variable

```
c1 = 3; c2
= c1+5;
clear c1

c1
```

Output:

```
Unrecognized function or variable 'c1'.
```

11 Intrinsic math functions and constants

```
x = pi; y = sin(pi/2) z = exp(-sin(pi/2))
```

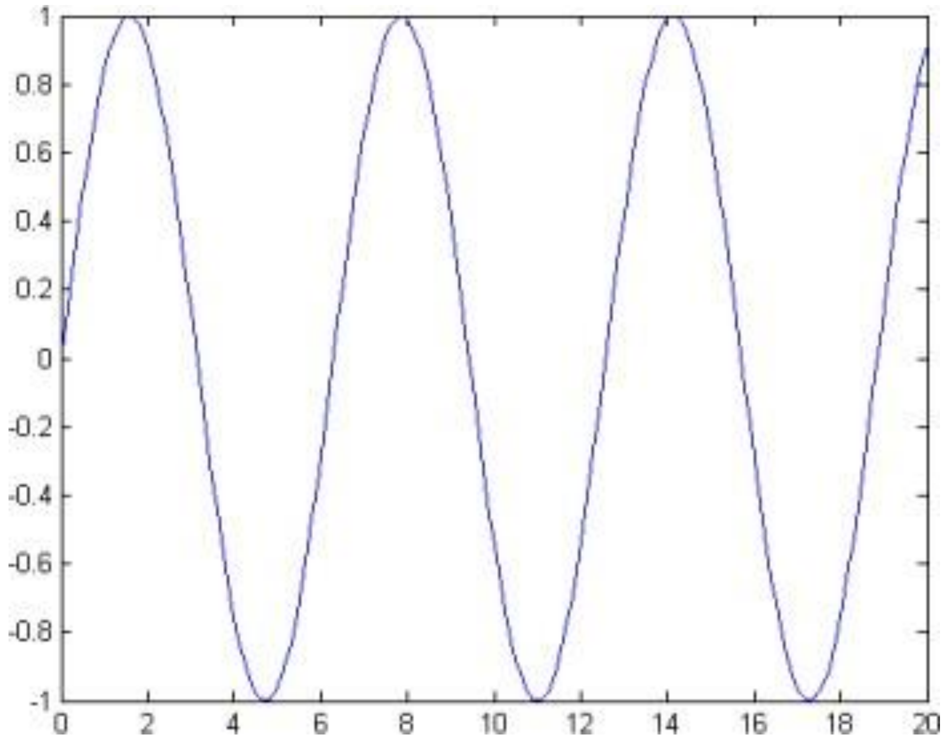
Output:

```
y = 1
z = 0.3679
```

13 Making a quick plot

```
x = [0:0.1:20];
y = sin(x);
plot(x,y)
```

The outcome will be the following plot:



14 Loop: Using for command

```
b = 3; for k
= 1:5
 b
end
```

Output:

```
3
3
3
3
3
```

15 *For* loop: Utility of the dummy index

```
b = 3; for k
= 1:5
 b^k
end
```

Output:

```
3
9
27
81
243
```

16 *For* loop: More on the dummy index

```
sum1 = 0;
for k = 1:9
 sum1 = sum1+k;
end sum1
```

Output:

```
45
```

17 *For* loop: More on the dummy index

```
sum1 = 0;
for k = 1:2:9
 sum1 = sum1+k;
```

```
end sum1
```

Output:

```
25
```

18 Treatment of array within a loop

```
b = [3 8 9 4 7 5]; sum1 =
0;
for k = 1:4
 sum1 = sum1+b(k);
end sum1
```

Output:

```
24
```

19 Treatment of array within a loop

```
b = [3 8 9 4 7 5];
sum1 = 0;
for k = 1:2:5
 sum1 = sum1+b(k);
end sum1
```

Output:

```
19
```

20 Double loop

```
sum1 = 0; for n
= 1:2
for m = 1:3
 sum1 = sum1+n*m; end
end sum1
```

Output:

```
sum1 = 18
```

21 Double loop

```
for n = 1:2
 for m = 1:3 fprintf('n = %3u m = %3u
 \r', n, m)
end end
```

Output:

```
 n = 1 m = 1 n
 = 1 m = 2 n =
 1 m = 3 n = 2
 m = 1 n = 2 m
 = 2 n = 2 m =
 3
0
```

22 More complicated use of loop and index

```
b = [2 5 7 4 9 8 3];
c = [2 3 5 7]; sum1 = 0;
for k = 1:4
 sum1 = sum1+b(c(k));
end sum1
```

Output:

```
24
```

Signature: ________________

Practical 3: Basic branching, Array and Matrix

1 The *if* command

```
num1  =  7;  if
(num1 > 5)
 fprintf('%4u is greater than 5 \r', num1)
else fprintf('%4u is less than or equal to 5 \r', num1)
end
```

Output:

```
 7 is greater than 5
```

Same program, but change first line to "num1 = 3;"

Output:

```
 3 is less than or equal to 5
```

2 *if - elseif - else* (This example is self-explanatory. Try to change the given value of num1 and observe the outcome.)

```
num1   =   4;   if
(num1 >= 5)
 fprintf('%4i is greater than or equal to 5 \r', num1)
elseif (num1 > 1) fprintf('%4i is less than 5 but greater
 than 1 \r', num1)
elseif  (num1   ==   1)  fprintf('%4i
 equals 1 \r', num1)
elseif (num1 > -3)
 fprintf('%4i is less than 1 but greater than -3 \r', num1)
else fprintf('%4i is less than or equal to -3 \r',
num1) end
```

Output

```
4 is less than 5 but greater than 1
```

3 An application - determine whether a given year is a leap year (try to change the given value of nyear and observe the outcome)

```
nyear      =      1975;      if
(mod(nyear, 400) == 0)
 fprintf('%6u is a leap year', nyear)
elseif (mod(nyear,4) == 0) & (mod(nyear,100) ~= 0)
 fprintf('%6u is a leap year', nyear)
else fprintf('%6u is not a leap year', nyear)
end
```

Output:

```
 1975 is not a leap year
```

4 Combine looping and branching

```
sum1 = 0;
sum2 = 0; N
= 9
for k = 1:N
 sum1 = sum1+k; if
 (mod(k,3)   ==   0)
 sum2 = sum2+k;
 end
end sum1

sum2
```

Output:

```
sum1 = 45

sum2 = 18
```

5 The *while* loop

```
x = 3; while (x <
100)
```

```
 x = x*3;
end
x

Output: x

= 243
```

6 Assign the content of a (one-dimensional) array; Addition of two arrays

```
a = [2 12 25];
b = [3 7 4]; c
= a+b Output:
c = 5 19 29
```

7 Assign the content of a matrix; Addition of two matrices

```
a = [3 4; 1 6]; b
= [5 2; 11 7]; c
= a+b Output:

c = 8 6
 12 13
```

8 Multiplication involving a scalar and an array (or a matrix)

```
a = [3 5; 1 4]; b = 2*a Output:

b = 6 10
 2 8
```

9 Element-by-element multiplication involving two 1-D arrays or two matrices of the same dimension

```
a = [2 3 5]; b
= [2 4 9]; c =
a.*b Output:
```

```
c = 4 12 45
```

10 Element-by-element multiplication of two matrices

```
a = [2 3; 1 4]; b
= [5 1; 7 2]; c =
a.*b Output:
```

```
c = 10 3
      7 8
```

11 Direct (not element-by-element) multiplication of two matrices

```
a = [2 3; 1 4]; b
= [5 1; 7 2]; c =
a*b Output:
```

```
c = 31 8
 33 9
```

12 Elementary functions with a vectorial variable

```
a = [2 3 5]; b = sin(a)
```

Output:

```
b = 0.9092 0.1411 -0.9589
```

13 Another example of elementary functions with a vectorial variable

```
a = [2 3 5]; b = 2*a.^2+3*a+4
```

Output:

```
b = 18 31 69
```

14 An efficient way to assign the content of an array

```
a = [0:0.5:4];
```

a *Output*:

```
a = 0 0.5 1 1.5 2 2.5 3 3.5 4
```

15. Extracting the individual element(s) of a matrix

```
A = [3 5; 2 4]; c =
A(2,2)+A(1,2)
```

Output:

```
c = 9
```

16 Another example for the usage of index for a matrix

```
A = [3 5; 2 4];
norm1 = 0; for
m = 1:2
for n = 1:2
 norm1 = norm1+A(m,n)^2;
end end norm1 =
sqrt(norm1)
```

Output:

```
norm1 = 7.348
```

17 Solving a system of linear equation

```
A = [4 1 2; 0 3 1; 0 1 2]; b
= [17 ; 19 ; 13]; x =
inv(A)*b
```

Output:

```
x = 1
```

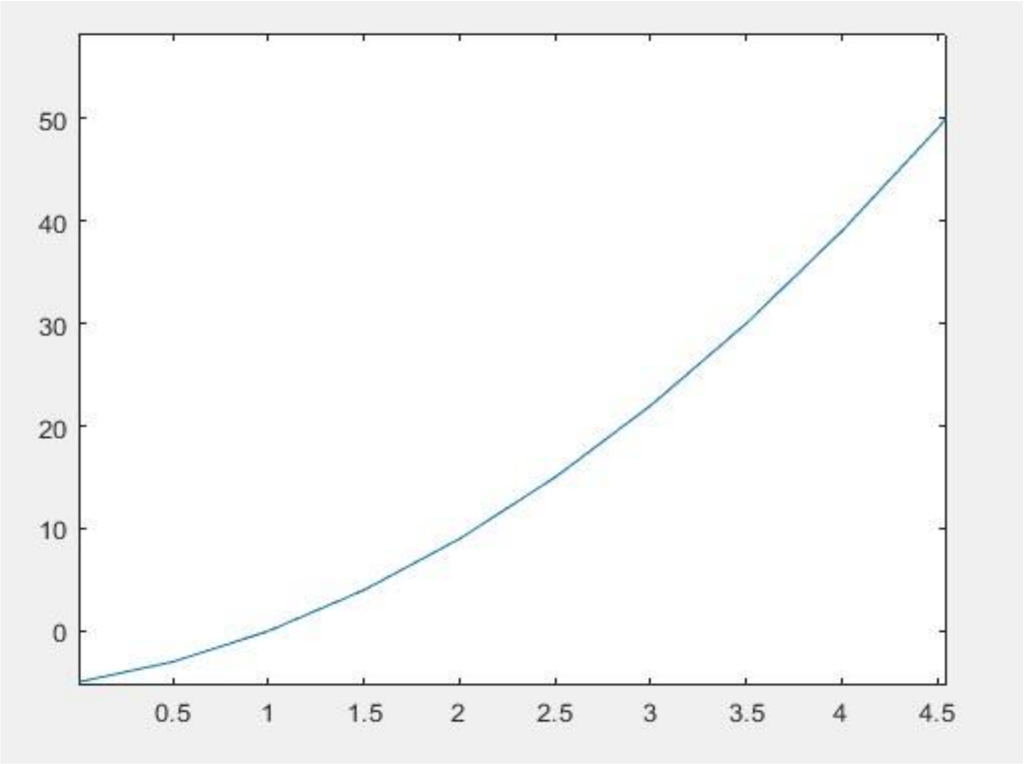
5
4

Signature: _______________

Practical 4: Basic graphical operation

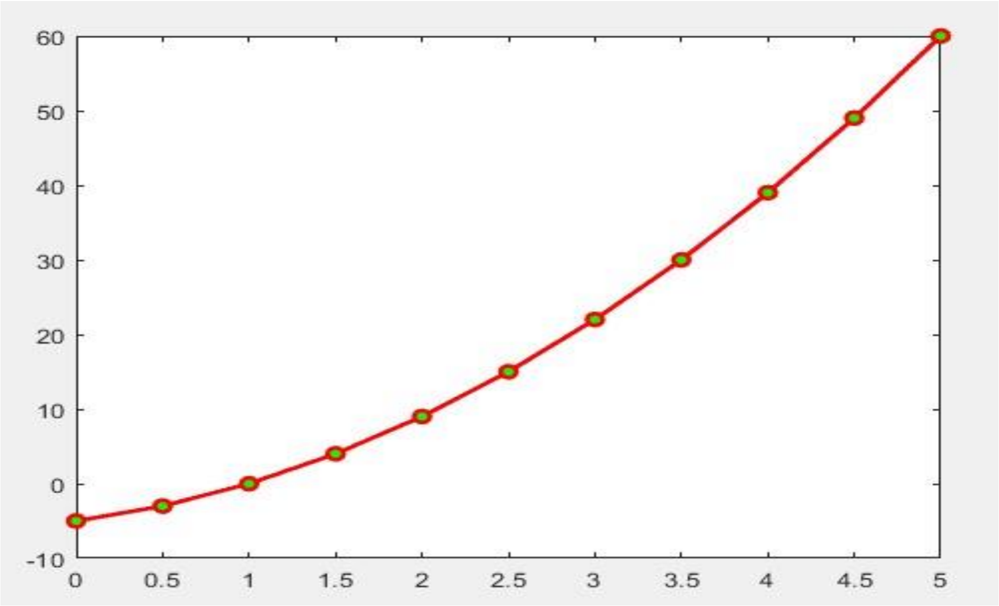
1. Draw a curve

```
a=[0:0.5:5]; b=2*a.^2+3*a-
 5; plot(a,b)
```



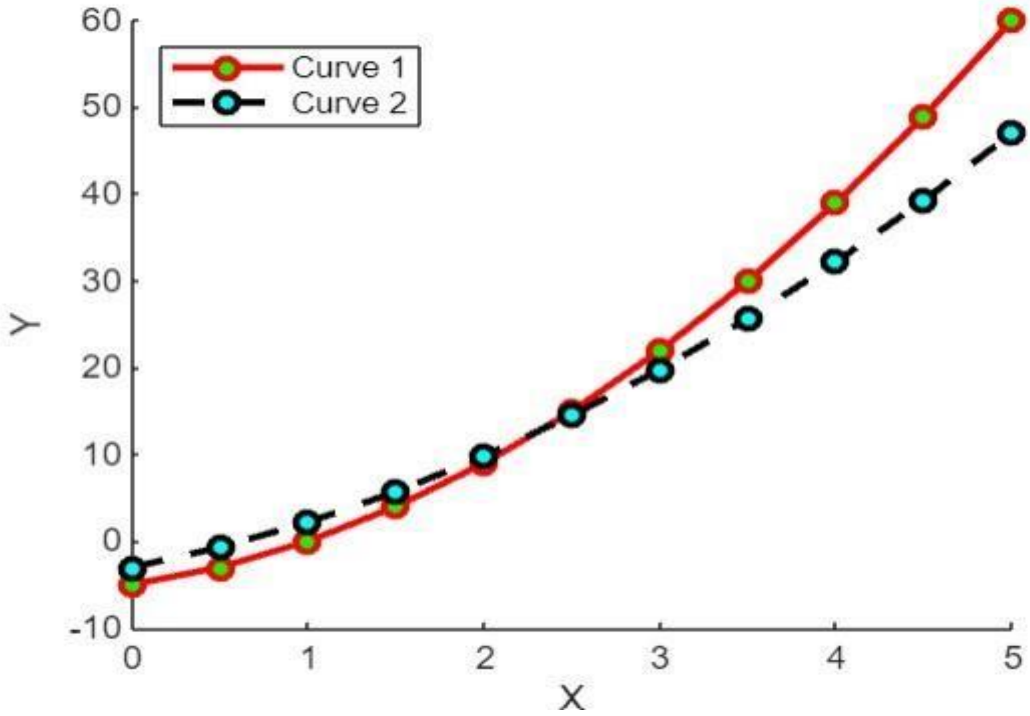
2. Refine the plot: Line pattern, color, and thickness

```
a=[0:0.5:5];
b=2*a.^2+3*a-5;
 plot(a,b,'-or','MarkerFaceColor','g','LineWidth',2)
xlabel('X');ylabel('Y');legend('Test','Location','NorthWest')
```



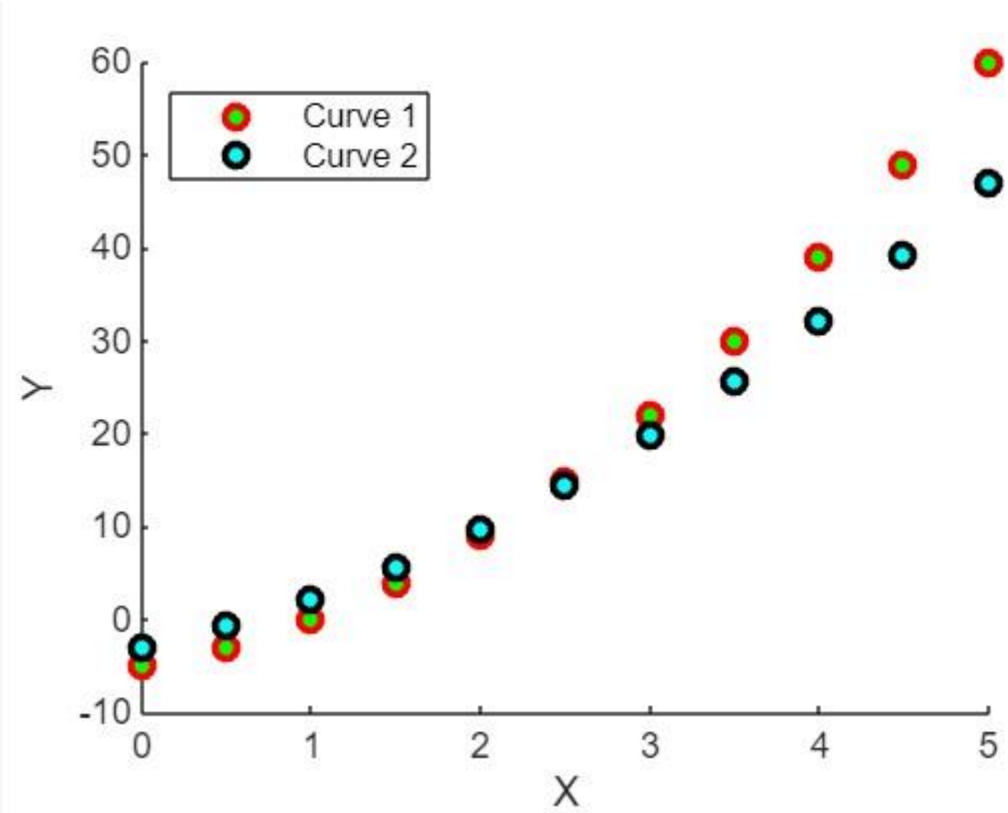
3. Draw multiple curves

```
a = [0:0.5:5]; b = 2*a.^2 + 3*a -5; c = 1.2*a.^2+4*a-3; hold on plot(a,b,'-
or','MarkerFaceColor','g','LineWidth',2) plot(a,c,'--
ok','MarkerFaceColor','c','LineWidth',2) xlabel('X'); ylabel('Y');
legend('Curve 1','Curve 2','Location','NorthWest')
```



4. Draw symbols

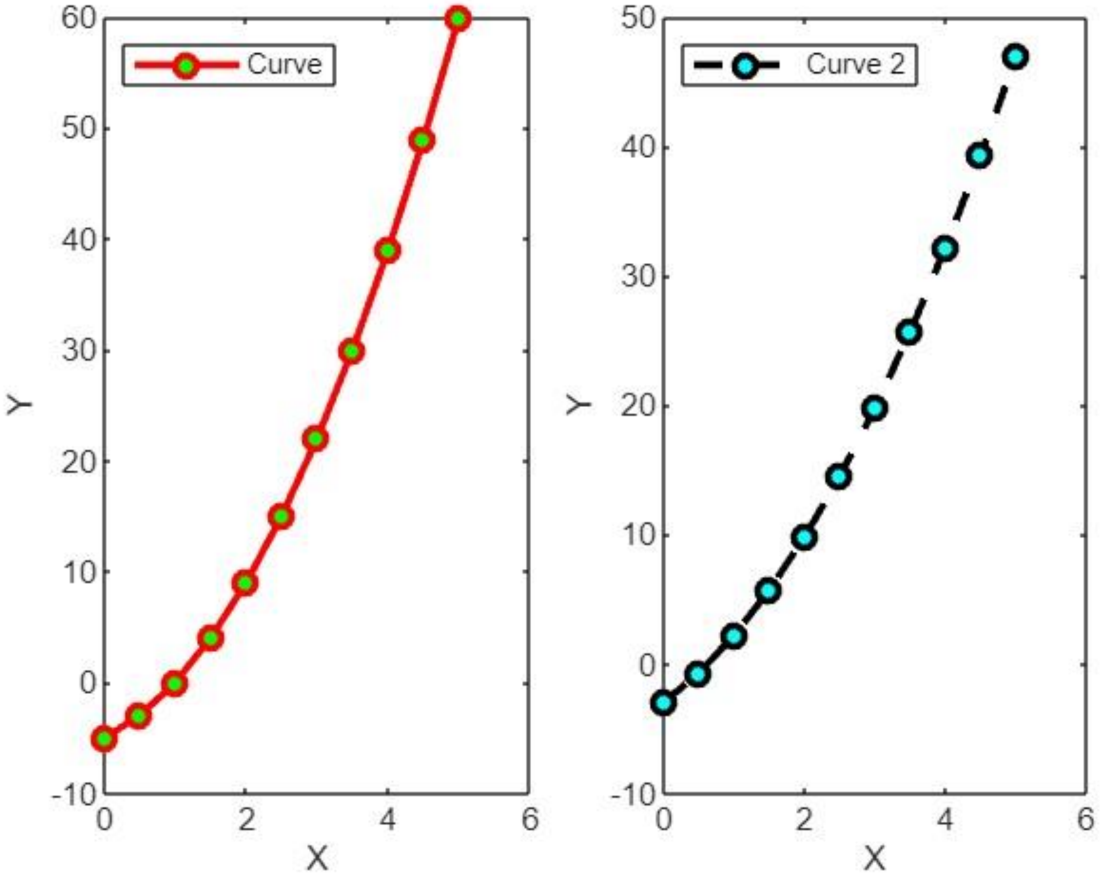
```
a = [0:0.5:5]; b = 2*a.^2 + 3*a -5; c = 1.2*a.^2+4*a-
3;                    hold                    on
plot(a,b,'or','MarkerFaceColor','g','LineWidth',2)
plot(a,c,'ok','MarkerFaceColor','c','LineWidth',2)
xlabel('X'); ylabel('Y'); legend('Curve 1','Curve

        2','Location','NorthWest')
```



5. Plot with multiple panels

```
a   =   [0:0.5:5];   b   =
2*a.^2 + 3*a -5; c =
1.2*a.^2+4*a-3;
subplot(1,2,1)
plot(a,b,'-or','MarkerFaceColor','g','LineWidth',2)          xlabel('X');
ylabel('Y');  legend('Curve  ','Location','NorthWest');  subplot(1,2,2)
```

```
plot(a,c,'--ok','MarkerFaceColor','c','LineWidth',2)        xlabel('X');
ylabel('Y'); legend('Curve 2','Location','NorthWest')
```



Signature: ________________

Practical 5: External (user-defined) function, Use external files and prompt for input and output

1 How to construct and use an external function

First, create the Matlab program for an external function as a separate file. The filename has to be identical to the name of the user-defined function

myfunc1.m

```
function outcome = myfunc1(x)
outcome = 2*x^2 + 3*x + 7;
```

This program defines a function, f(x) ≡ 2 x^2 + 3 x + 7. The "outcome" in the program is a dummy variable that is used to indicate the relation between the input parameter ("x") and the outcome after the evaluation of the function. After an external function is defined, it can be used in a main program that calls it. For example:

mainprog1.m

```
for n = 1:5
 x = n*0.1; z =
 myfunc1(x);
 fprintf('x = %4.2f f(x) = %8.4f \r',x,z) end
```

Output:

```
x = 0.10 f(x) = 7.3200 x
= 0.20 f(x) = 7.6800 x =
0.30 f(x) = 8.0800 x =
0.40 f(x) = 8.5200 x =
0.50 f(x) = 9.0000
```

2 A function with multiple input parameters

myfunc2.m

```
function outcome = myfunc2(x,a,b,c) outcome = a*x^2 + b*x + c;
```

This program defines a function, f(x) ≡ a x^2 + b x + c.

mainprog2.m

```
                                 for n = 1:5
        x = n*0.1;
                                  z = myfunc2(x,2,3,7);

                                  fprintf('x = %4.2f f(x) = %8.4f \r',x,z) end
```

Output:

```
x = 0.10 f(x) = 7.3200 x
= 0.20 f(x) = 7.6800 x =
0.30 f(x) = 8.0800 x =
0.40 f(x) = 8.5200 x =
0.50 f(x) = 9.0000
```

3 Open a file and write the output to the file

```
fID1 = fopen('myoutput1.txt','w'); for n
= 1:4
 b1 = n ; b2 = n^2 ; b3 = n^3;
fprintf(fID1,'%7u %7u %7u \r',b1,b2,b3); end
```

This program will produce no output on the screen but will create a file called "myoutput.txt" (under the same directory where the Matlab program itself is stored). The content of the file is

```
1 1 1
2 4 8
3 9 27
```

```
4 16 64
```

4 Read data from an existing file

```
fID1 = fopen('myoutput1.txt','r'); for n
= 1:4
 b = fscanf(fID1,'%7u %7u %7u \r',3); btotal =
 b(1)+b(2)+b(3);
 fprintf('%7u + %7u + %7u = %7u \r', b(1), b(2), b(3), btotal)
end
```

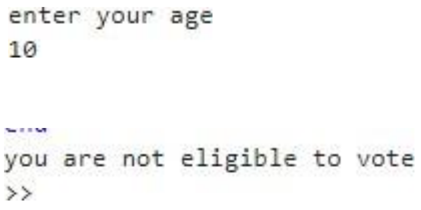
Output:

```
1 + 1 + 1 = 3
2 + 4 + 8 = 14
3 + 9 + 27 = 39
4 + 16 + 64 = 84
```

5 Create a prompt to request user input

```
num1 = input('Enter your age'); if
(num1 > 17)
 fprintf('You are eligible to vote')
else fprintf('You are not eligible to
 vote')
end
```

Output



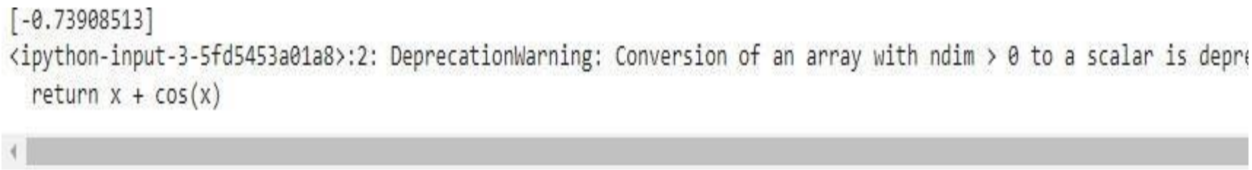
```
enter your age
10

you are not eligible to vote
>>
```

Signature: ________________

Practical 6: Linear Programming

1.find root of the equation x + cos(x) :

```
from scipy.optimize import root from math import
cos

 def equ(x): return x +
cos(x)

 myroot=root(equ,0)
print(myroot.x)
```



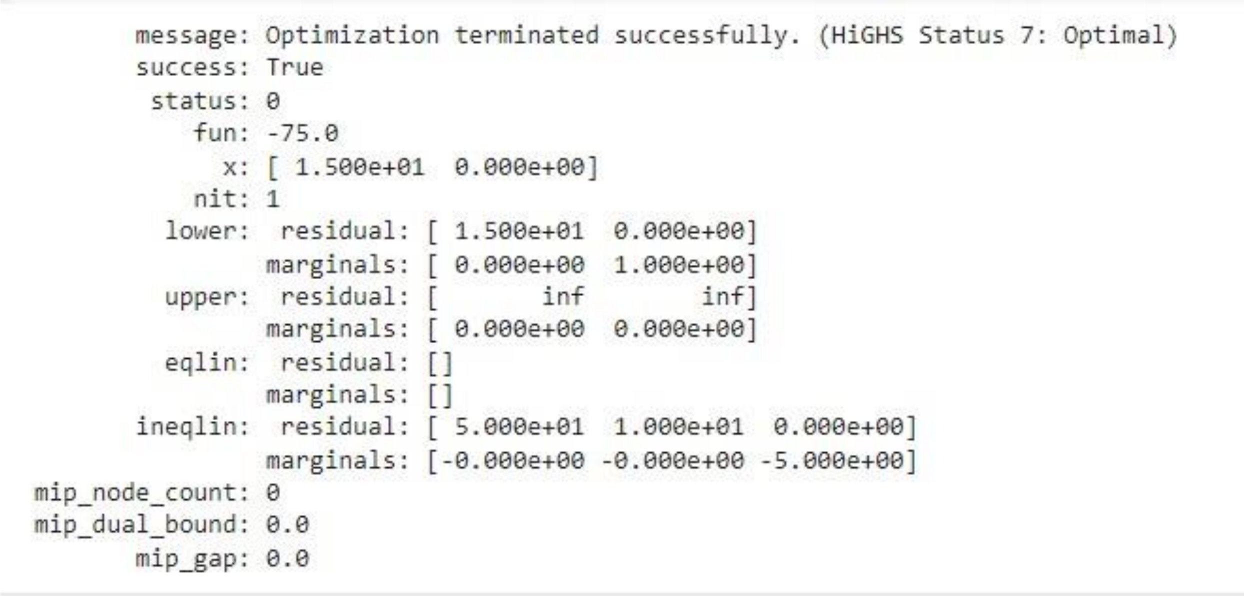
```
[-0.73908513]
<ipython-input-3-5fd5453a01a8>:2: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is depre
  return x + cos(x)
```

2.maximize: Z=5x +4y

Constraints: 2y <= 50,x + 2y <= 25,x + y <= 15

x >=0 and y >= 0

```
from scipy.optimize import linprog #obj function coefficient matrix obj= [-5,-4]

#constraints left side x and y coefficient matrix lhs_ineq = [[0, 2], #0x1 + 2x2
                         [1, 2], #x1 + 2x2
                         [1, 1],] #1x1 + 1x2

#right sdie values matrix rhs_ineq =[50, #...<=50
                       25, #..<=25
                       15] #...<=15

#inbuilt function <linprog> will
#solve the problem optimally
#passing the each coefficient's matrices opt=linprog(c=obj,
                         A_ub=lhs_ineq, b_ub=rhs_ineq,
                    method="highs")

#printing the solution print(opt)
```



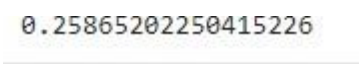
```
        message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
        success: True
         status: 0
            fun: -75.0
              x: [ 1.500e+01  0.000e+00]
            nit: 1
          lower:  residual: [ 1.500e+01  0.000e+00]
                  marginals: [ 0.000e+00  1.000e+00]
          upper:  residual: [       inf        inf]
                  marginals: [ 0.000e+00  0.000e+00]
          eqlin:  residual: []
                  marginals: []
        ineqlin:  residual: [ 5.000e+01  1.000e+01  0.000e+00]
                  marginals: [-0.000e+00 -0.000e+00 -5.000e+00]
 mip_node_count: 0
 mip_dual_bound: 0.0
        mip_gap: 0.0
```

Signature: ________________

Practical 7: Nonlinear programming

find the smallest positive root of the function f(x) = x3 -2x + 0.5 using newton-raphson method

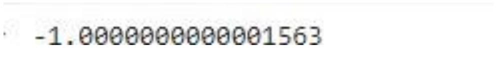
```
from scipy.optimize import newton #objective function
 def f1(x): return x*x*x - 2*x + 0.5

#newton function will return the root
print(newton(f1,0))
```



```
0.25865202250415226
```

find the root of f(x)=x^3-x^2+2 using bisection method with initial value a = -200 b=300

```
from scipy.optimize import bisect #objective function
 def f1(x): return (x**3) - (x**2) + 2

#bisect function will return the root print(bisect(f1,-
200,300))
```



```
-1.0000000000001563
```

Signature: ________________