Octave Lab

~~An~~ It is a programming language. ~~A~~ Any programming language has ~~many~~ mainly two language.

— (i) High level language
— (ii) Assembly language.

Actually the former is that type of language ~~to~~ by which we produce a code to the computer. Interpreter ~~for compiler~~ takes the code and convert it into ~~in~~ machine language. Then the processor starts work.

IDE ▷ Integrated Development Environment

It is the platform where we write the programme.

Grammer of Octave language: It is called Syntax. If the user make any mistake gramatically (not the english grammer. Here grammer means the language of the computer programming) then after compiling the code computer / the programming software shows "Syntax error"

Part of Octave programme:

It has mainly 3 parts —

① File browser (where all the files in your computer are displayed)

② Work Space (All the results obtained by running a programme are displayed in the work space)

③ Command history (where all the commands given by the user are displayed)

Programme

```
>> a = 5+7
   Enter
a = 12
>> b = 5*7
   Enter
b=35
```

users input

output

In the workspace it is displayed —

Name	class	Dimension	Value
a	double	1×1	12
b	double	1×1	35

Programme

```
>> S = "Hellow"
   Enter
```

* must be written in single coat (' ') or double coat (" ")

S = Hellow

In the workspace it is displayed

Name	class	Dimension	Value
S	char	1×6	Hellow

Here 'double' comes for all numerical and 'char' stands for character.

Prohibited variable

i, j, e, are prohibited variable because the octave already sets the value of those.

Programme

```
>> z = i
   Enter
```

z = 0 + 1i

Programme #1

```
>> x = e
   Enter
   x = 2.7182
>> format long
   e
   Enter
   ans = 2.718281828459045
```

* Actually octave shows the four digit after decimal but the rest digits store. If we use the command "format long" the full value is displayed.

Programme 2

```
>> a = 8;
```

Then a = 8 is not displayed as answer.

Let a particle is thrown with initial velocity 5 m/s. The acceleration is 10 m/s². Find the displacement at a time t = 2 s.

```
>> u = 5;
   f = 10;
   t = 2;
   s = u*t + 0.5*f*t^2
   Enter
   s = 30
```

Left side is variable and right side is assigned by the right side value.

To save it, write the programme in the editor → click → choose a folder → put a name of file → save.

To delete it,

File browse → click here → choose the folder → choose file → right click on the file → delete.

To rename it,

File browser → click on → choose the folder → choose file → right click on the file → Rename

Some information: Computer is able to read ASCII code. If I write 'x' then the computer takes the corresponding ASCII code (120 is the ASCII code of x).

About interpreter and compiler

Both act as a bridge between the machine language and the ~~langug~~ language given by me to the computer. (Actually the computer is able to understand binary code which is not understood by user. So it is necessary to convert the text language given by user to the machine language. ASCII code is the most common character encoding format for text data in computer and on the internet). Assembly language is the correction language of ASCII code. But the problem of assembly language is ~~portab~~ portability — that means, the language of a machine does not understand by the other machine.

There are some programming like Octave, Python which are interpreter based but the programming like C, C++, Java are compiler based. Interpreter reads the programme line by line so in case of an interpreter based software, ram usage is higher than a compiler based programming because the compiler reads the programming code as a hole.

To clear the workspace used command
"clear"

Programme (In IDE)

```
>> X = 5
>> Y = 3
>> clear X
>> clear Y
```

output (In Workspace)

name	class	dimension	value
X	double	1x1	5
Y	"	"	3
Y	"	"	3

To clear the IDE used command
"clc"

If user type clc and then press the 'enter' button, the whole IDE (where the user wrote the programme till now) is cleared.

Function in Octave

There are some inbuild function in Octave. Those are — sin, cos, tan, log (base 10 and e both).

Programme (In IDE)	output (In command window)	output (In Workspace) name	class	dimen	value
`sin(pi/2)`	ans = 1	ans	double	1x1	1
`sin(pi/3)`	ans = 0.8660254037844439	ans	double	1x1	0.866
`cos(pi/2)`	ans = 6.123031769111886e-17	ans	double	1x1	6.12 — taken as 0
`cos(pi/4)`	ans = 0.7071067811865548	ans	double	1x1	0.707
`clear`					
`a = sin(pi/6)`	a = 0.500000000000	a	double	1x1	0.500
`b = cos(pi/3)`	b = 0.500000000000	a	double	1x1	0.5000
		b	double	1x1	0.500

Programme (In IDE)	output (in command window)	output (in workspace) Name	class	dimen	Value
>> clear ↵					
>> log(10) ↵	ans = 2.302585092994046	ans	double	1x1	2.302
>> log(20) ↵	ans = 2.995732273553991	ans	"	"	2.995
>> log10(10) ↵	ans = 1	ans	"	"	1
>> log10(50) ↵	ans = 1.698970004336019	ans	"	"	1.698
>> log10(100) ↵	ans = 2	ans	"	"	2

Note:- log( ) → Natural log (base e)

~~log10~~ log10( ) → 10 base log

~~To in~~

Implementation of Boolean algebra related programme

~~Prog~~

Code in IDE	output in command window	output in workspace Name	class	dim	value
>> a=1; ↵		a	double	1x1	1
>> b=0; ↵		b	"	"	0
>> or(a,b) ↵	ans = 1	a	"	1x1	1
		ans	"	"	1
		b	"	"	0
>> a||b ↵	ans = 1	a	"	1x1	1
		ans	"	"	1
		b	"	"	0
>> a=1; ↵		a	double	1x1	1
>> b=1; ↵		b	"	"	1
>> and(a,b) ↵	ans = 1	a	"	"	1
		ans	"	"	1
>> a&&b ↵	ans = 1	b	"	"	1
		a	"	"	1
		ans	"	"	1
		b	"	"	1
>> a=1; ↵		a	"	"	1
>> not(a) ↵	ans = 0	a	"	"	1
		ans	"	"	0
>> !a ↵	ans = 0	a	"	"	1
		ans	"	"	0

Boolean OR operation

Boolean AND operation

Boolean NOT operation

To get any help use command "help".
To get any help on a special case like dot (.), & etc.. use command 'help .' ↵, 'help &' ↵
help < > dot, &, etc

If it is show that "EOL error" — it means that ← "End of Line error" → There is an error at the end of the specific line.

Implementation of Matrix in Octave

To specify the element of a row we use 'comma' (,) and for a column we use 'semi colon' (;).

code in IDE	output in command window	output in workspace			
		Name	class	Dimension	value
[1,2,3; 4,5,6; 7,8,9]	ans = 1 2 3 4 5 6 7 8 9	ans	double	1x1	[1,2,3;4,...
a=[1,2,3; 9,4,6; 7,8,5]	a = 1 2 3 9 4 6 7 8 5	a	double	1x1	[1,2,3;9,...

To get a matrix with all element zero use command — zeros( ). Order of matrix is given in bracket. Suppose if we put the command zeros(4), we get a 4x4 matrix which has all the elements are zero.

To get a unit matrix of order n use command — eye(n)
of order n

To get a matrix with all element one use command — ones(n) ~~or~~
or zeros(n)+1

Implementation of a matrix with random number

For a random number use command — rand(1). To get a matrix of order n with random number use command — rand(n). For a matrix — (of order 4)

```
>> a = rand(4) ↵   (4x4 matrix)
>> a' ↵            (transpose of matrix 'a')
>> issymmetric(a) ↵   (to identify if the matrix 'a' is symmetric or not)
>> b = rand(5) ↵
>> b' ↵
>> S = 0.5*(b+b') ↵   (To get symmetric part of the matrix)
>> issymmetric(S) ↵   (To check if 'S' is symmetric or not)
```

Here if 'ans = 1' is output then it means that "yes, 'S' is a symmetric matrix".

```
>> AS = 0.5*(b-b') ↵   (To get anti symmetric part of the matrix)
>> issymmetric(AS) ↵   (To check if 'b' is anti symmetric or not)
```

Here if 'ans = 0' is output then it means that "No, 'b' is not a symmetric matrix"

Empty matrix → A matrix which has no row and column. It is 0×0 order all time.

Programme	output in IDE
>> a = [ ] ↵	a = [ ] (0×0)
>> b = [ ] (5,1)	error

The dimension is not defined. So the error of empty matrix is shown.

Actually by the code "b=[ ](5,1)", a user instruct to the computer (octave) that create an empty matrix which has 5 no. of rows and 1 no. of column. Note that any empty matrix has no row and column but by the code user create a contradiction. So, error show.

If loop

The general rule of if loop is —

```
if (condition)
  'body'
endif
```

'body' means display 'something'

If-else

The general rule of if else is —

```
if (condition)
  'body'
else
  'body'
endif
```

If the condition under 'if' is not true then the other possible way is expressed by 'else'. It is used for mainly two cases — like if a number is not even then it is odd.

If-elseif-else

~~The If the cond~~

It is used when multiple conditions have to check. At first the interpreter of octave reads the condition under 'if'. If that is true then it prints the body and the loop is closed. But if that is not true then the interpreter checks the condition under 1st 'elseif'. If that is true then prints the body under 1st 'elseif' and the loop is closed. But if that is not true, the interpreter

checks the conditions under each 'elseif'. Which condition is true the corresponding body is printed and the loop is closed.

Programme: check the roots of a quadratic equation (without time)

```
a = input ('enter the number:' )
b = input ('enter the number:' )
c = input ('enter the number:' )
d = b^2 - 4ac
r1 = (-b + sqrt(d))/(2*a)
r2 = (-b - sqrt(d))/(2*a)
if (d>=0 && r1>0 && r2>0)
   disp 'both roots are +ve'
elseif (d>=0 && r1>0 && r2<0)
   disp 'r1 is +ve but r2 is -ve'
elseif (d>=0 && r1<0 && r2>0)
   disp 'r1 is -ve but r2 is +ve'
elseif (d>=0 && r1<0 && r2<0)
   disp 'both roots are -ve'
else (d<0)
   disp 'both roots are imaginary'
endif
```

Programme: Take a number and check if it is divisible by some other number

Here we take a variable n which has to check if the number is divisible or not by the number a=2, b=3, c=5. You may take other numbers instead of 2, 3, 5.

programme

input code

```
n=input ('Enter the number:')
if (rem(n,2)==0) && rem(n,3)==0) && rem(n,5
if(rem(n,2)==0) && rem(n,3)==0) && rem(n,5)==0)
        printf 'n is divisible by 2,3,5'
  elseif (rem(n,2)==0) && rem(n,3)==0)
        pritf 'n is divisible by 2 and 3 only'
  elseif (rem(n,3)==0 && rem(n,5)==0)
        printf 'n is divisible by 3 and 5 only'
  elseif (rem(n,5)==0 && rem(n,2)==0)
        printf 'n is divisible by 5 and 2 only'
  elseif(rem(n,2)==0)
        printf 'n is divisible by 2 only'
  elseif (rem(n,3)==0)
        printf 'n is divisible by 3 only'
  elseif (rem(n,5)==0)
        printf 'n is divisible by 5 only'
  else
        printf 'n is not divisible by 2,3,5'
endif
```

Programme: check if a number is +ve odd or even or -ve odd or even.

Algorithm: A number is +ve or -ve is determined by if it is >0 or <0 and ~~odd~~ even or ~~even~~ odd is determined by if it is ~~no~~ divisible by 2 or not.

Input code

```
n = input ('enter the number: ')
If (n>0)
   disp 'the number is +ve'
   if (rem(n,2)==0)
      disp 'and even'
   else
      disp 'but odd'
   endif
else
   disp 'the number is -ve'
   if (rem(n,2)==0)
      disp 'and even'
   else
      disp 'but odd'
   endif
endif
```

It is called 'Nested if' - that means there is an 'if' loop inside an 'if' loop. Here there are two 'if' loop under one 'if' loop. The 1st 'if' loop checks if the number is +ve or -ve and the two sub 'if' loop checks if the number is odd or even.

Programme: check if ~~a triangle~~ it is possible to make a triangle by three given sides.

Algorithm: If the three sides of a triangle are a, b, c such that the ~~some~~ sum of any two sides is greater than the other side then the triangle formation is possible by the three sides.

Input code

```
a = input ('enter the side:');
b = input ('enter the side:');
c = input ('enter the side:');
if (a+b>c && b+c>a && c+a>b)
   printf ('making of triangle is possible
            by the three sides ')
                  ↳ give a <space>
   if (a==b || b==c || c==a)
      printf ('and the triangle is isoscales')
   else
      printf ('but the triangle is not isoscales')
   endif
elseif (a+b>c && b+c>a)
   printf ('making of triangle is not possible
            by the three sides')
elseif (b+c>a && c+a>b)
   printf ('making of triangle is not possible
            by the three sides')
elseif (c+a>b && a+b>c)
   printf ('making of triangle is not possible
            by the three sides')
endif
```

It is an application or an example of 'Nested if' discussed in former page.

Error in octave programme

(1) **Syntax error** ⇒ It is mainly gramatical mistake by the user. If Octave follows definite gramatical rule but if the user inputs a code which does not follow the gramatical language of Octave then "Systax error" is displayed when the programme is running.

(2) **Parse error** ⇒ A parse error occurs if Octave cannot understand something user has typed. For most parse error Octave uses a carret (^) to mark the point on line where it was unable to make sense of your input.

(3) **Source error** ⇒ If the programming code or the input text file of the given or running programme consist an error then octave gives a message — "error: source: error sourcing file 'E:/folder/name.m'"

- drive address
- name of folder
- name of file saved before running
- extension

(4) **Evaluation error / Runtime error** ⇒ (google)

Switch & case

It is a condition like language in octave. under one Switch there is a numbers of cases. At first we have to define a variable on which switch acts. In the cases there is given some condition and corresponding a body which is printed. The which condition is matched, those corresponding body is printed by the octave.

Programe: Type a single digit number and the computer will write its value in words.

Algorithm: At first we introduce a variable. Then we have to act 'switch' on the variable. There are some cases. which case matches with the variable under 'switch' function.

Programme:

```
a = input ('enter the number')
switch (a)
    case 1
        printf ('a = one')
    case 2
        printf ('a = two')
    case 3
        printf ('a = three')
    case 4
        pritf ('a = four')
    otherwise
        printf ('a is greater than or equal to five')
endswitch
```

output

```
enter the number 2
a = 2
a = two
```

```
enter the number 10
a = 10
a is greater than or equal to five
```

Another example

Programme

```
a = input ('enter the number')
switch (a)
  case {1,2,3}
    printf ('a belongs one to three')
  case {4,5,6}
    printf ('a belongs four to six')
  case {7,8,9}
    printf ('a belongs seven to nine')
  case {10,11,12}
    printf ('a belongs ten to twelve')
  case {13,14,15}
    printf ('a belongs thirteen to fifteen')
  otherwise
    printf ('a is greater than fifteen')
endswitch
```

Running process

While the code is run, the computer will ask the user to give a number. User input a number. Then the Octave will check the number, given by the user, matched with which case. The number matched with which case, the corresponding body is printed. If no case match, then the body of 'otherwise' is printed.

output

```
enter the number 11
a = 11
a belongs ten to twelve
```

```
enter the number 20
a = 20
a is greater than
fifteen.
```

~~Programme~~

Programme: Type a code which will convert the marks into percentage of marks and write the corresponding letter grade.

Algorithm: At first we have to introduce two variable which give the obtained marks and full marks. Then calculate the percentage of marks. Now introduce 'switch' function but changing the name — 'Switch true'. ~~The switch~~ Switch true acts on the percentage of marks. ~~How~~ Then some cases are given and which case matches with percentage of marks, the corresponding body is printed by octave.

Programme:

```
a=input ('enter the ~~numbe~~ marks obtained')
b=input ('enter the full marks')
c=(a/b)*100
Switch true
  case {c>=90 && c<=100}
    printf ('O [Remarks: outstanding]')
  case {c>=80 && c<=89}
    printf ('A+ [Remarks: awesome]')
  case {c>=70 && c<=79}
    printf ('A [Remarks: excellent]')
  case {c>=60 && c<=69}
    printf ('B+ [Remarks: very good]')
  case {c>=50 && c<=59}
    printf ('B [Remarks: good]')
```

```
case {c>=40 && c<=49}
    printf ('c+ [Remarks: okay]')
case {c>=30 && c<=39}
    printf ('c [Remarks: try to improve]')
 otherwise
    printf ('D [Remarks: fail]')

endswitch
```

output

```
enter the marks obtained 250
a = 250
enter the full marks 400
b = 400
c = 62.500
B+ [Remarks: very good]
```

argument → programme ↓ due GUI,

Programme: Type a code which displays the natural numbers and also their sum of squares.

Algorithm: At first we have to introduce three variable, one gives total number of natural numbers on the 2nd variable 'do' loop runs and third variable is count the sum.

Programme:

```
n = input ('enter the number :');
i = 0;
Sum = 0;
do
    printf ('the natural number is : %d\n', i);
    x = i^2;
    printf ('the square of the natural number
            is: %d\n', x)
    Sum = Sum + x
    i++
Untill (i>n)
printf ('the sum of squares of natural number
        is : %d', Sum);
```

'\n' দেওয়া হলে 'the natural number is:' এবং 'the square of natural number is:' এই দুটি line দুটি আলাদা row তে print হবে না হলে দুটি row তে দুবার print হবে যেটা আমরা চাই না।

'%d' এর মাধ্যমে computer বুঝতে পারে 'the natural number is' এর 'i'-variable-এ value টি print হবে, 'the square of the natural number is:' এর 'x'-variable-এ value টি print হবে।

about 'do' loop

It is a loop. The structure of it is —

```
i = 0
do
{body}
i++
until (i==10)
```

i = 0 → variable on which loop runs
do - until loop.
there is a body.
i++ → increament of i 1 time and 1 unit
until (i==10) → a condition, how much the loop runs.

Two important Sign

%d ⇒ to connect the variable with appropriate statement

\n ⇒ to change the row line for 2nd line which is to be printed.

Programme: Find N random numbers within the interval [0,1] and find their variance.

Algorithm: At first we have to introduce some random numbers let 5 or 10 nos of random numbers between 0 and 1. It looks like an 1d array. Then we calculate mean of the elements of the array and then using do loop we calculate variance. Then we verify the result by the inbuild function of Octave

Programme/code: (using 'do' loop)

```
n= input ("enter the number:")
x = rand (1,n)
m = mean (x)
i = 1;
Sum = 0;
do
   S = [x(i) - m]^2;
   Sum = Sum + S
   i++;
until (i>n)
Variance = Sum/n-1
vari  = var (x)
```

Program/code: (using 'for' loop)

```
n = input ("enter the number:")
x = rand (1,n)
m = mean (x);
S = 0;
for i = 1:n
   S = S + (x(i) - m)^2;
endfor
v = S/(n-1)
V1 = var (x)
```

Programme: If n uniformly distributed random numbers are generated between +1 and -1, find the number (b) of non negative numbers generated

Algorithm: At first we have to generate a 1d array from 0 to 1. Then multiplying each element by 2 and then subtracting 1 we get a new 1d array b/w +1 to -1. Then by using proper way we have to determine non -ve numbers.

Program/code:

```
n = input ("enter the number:")
a = rand (1, n)
b = 2*a -1
s = 0
for i = 1: n;
   if b(i) >= 0 ;
      s = s+1;
   endif
endfor
disp (s)
```

Program/code (using do loop)

```
n = input ("enter the number:");
a = rand (1, n);
b = 2*a - 1
s = 0;
i = 1;
do
   if b(i) >= 0
      s = s+1;
   endif
   i++;
until (i > n)
s
```

While loop

For loop + if condition = While loop.

First of all a variable should be defined on which the loop runs. Then start the loop. The form of the loop is —

```
i = a number
While (i<'or'>n)
 body
 i++
endWhile
```

condition - either i>n, or i<n

if the condition is true then and only then the loop runs, otherwise not run.

Programm: write a program which will display reciprocal of natural numbers starting from 1 until the reciprocal contains a non zero digit in 3rd decimal place.

Algorithm: At first we have to define a variable which starts from 1. Then a 'while loop' starts with proper condition. Then increment of i is done inside the loop.

Program/code:

```
output_precession(4); # decimal place -> এর 3 টা digit এর পর চলবে।
i=1
while mod(1000/i,10)!=0
   i=i+1;
endwhile
disp(i)
```

Program: Find the value of exp(-iθ) for a given θ accurate upto 6th decimal place

Algorithm: First of all, we have to build the sine and cosine series. Then, adding them in complex number we get exp(-iθ).

Program/Code:

```
clear;
output_precision(7)
x = input('enter the value of angle:')
i = 1;
j = 1;
s1 = 0;
s2 = 0;
t1 = 1;
t2 = 1;
while (t1 > 1e-6)
t1 = (-1)^(i+1) * (x-0)^(2*i-1) / factorial(2*i-1);
s1 = s1 + t1;
i++;
endwhile
while (t2 > 1e-6)
t2 = (-1)^(j) * (x-0)^(2*j) / factorial(2*j);
s2 = s2 + t2;
j++;
endwhile
disp(s1)
disp(s2)
z = complex(s2, -s1)
z = complex(0, -x);
r = exp(z)
```

↳ sine series

↳ cosine series

sine

cosine

to make (-ix)

e^{-ix}

Program: Find the value of ln(2.5) accurate upto 4th decimal place by series ~~sup~~ Summation

Algorithm: We have to notice the series of ln x. Then by using while loop we get the series summation of ln x.

Program/code:

```
x = input ("enter the value:")
i = 1;
s = 0;
t = 1;
while (t>1e-4)
t = 2*((x-1)/(x+1))^(2*i-1)/(2*i-1);
s = s+t;
i++;
endwhile
disp (s)
```

→ series of ln(x)

i → কত বার run করার পর precision reach করে নেবে বা কত বার run করার পর থেকে further run করলে 4th place-র digit আর change করছে না।

এখানে t-র প্রথম একটা value choose করতে হবে যেটা 1e-4 থেকে বড়, but not necessarily 1. Then while-র condition টা satisfy করবে তাহলে while loop-র ভিতরে ঢুকে একবার t-র value টা change হয়ে যাবে, তবে এখানে ln x-র series টা place করতে গেলে দিতে হবে, Then সেটা s-র মধ্যে store করতে হবে তারপর i-র increment হবে, ~~সবশেষে t এর value 1e-4 থেকে বড় হলে~~

Blank page (Probably due)

Nested 'do' loop

do loop inside a do loop is nested do loop.

Program: Generate an mxn matrix and an nxk matrix and add them using element wise relation of matrix addition.

Algorithm: At first we have to define two same order matrix. Then using loop we have to sum elementwise, as it is done in the matrix algebra.

$A=[a_{ij}], B=[b_{ij}] \Rightarrow c=A+B$

$c_{ij}=a_{ij}+b_{ij}$

$c_{11}=a_{11}+b_{11} \quad c_{12}=a_{12}+b_{12}$

$c_{21}=a_{21}+b_{21} \quad c_{22}=a_{22}+b_{22}$

program/code: (using for loop)

```
a = randi (10, 3)
b = randi (10, 3)
c = zeros (3)
l = size(a)(1);
for i=1:l;
   for j=1:l:
      c(i,j) = a(i,j) +b(i,j);
   endfor
endfor
disp(c)
```

randi = to choose integer randomly.

randi(p, q)
- p denotes upper limit of our choosen area.
- q denotes the order of square matrix

randi(10,3) gives a 3x3 square matrix whose elements belong in [1, 10].

size (a) (1)
- identify the matrix on which size function acts.
- nos. of rows

size (a) (1) → nos. of rows
size (a) (2) → nos. of colourns
for a 3x4 matrix

Program: Generate an mxn matrix and an nxk matrix and multiply them using element wise relation of matrix multiplication.

Algorithm: At first we have to define two same order matrix. Then using loop we have to multiply element wise as it is done in matrix algebra.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \xrightarrow{\times} AB = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & \cdots & \\ \vdots & & \\ & & \end{bmatrix}$$

Code: (using for loop)

```
A = randi(10, 3)
B = randi(10, 3)
c = zeros(3)
l = size(a)(1)
for i = 1:l
    for j = 1:l
        for k = 1:l
            c(i,j) = c(i,j) + A(i,k) * B(k,j);
        endfor
    endfor
endfor
disp(c)
```

Break and continue:

The two in built command. If any condition is true the prest program is read by the octave otherwise not.

Program: construct an array of values for x from 0 to 2 with interval size 0.1. Write the corresponding ~~code~~ values of $(x^2-1)/(x-1)$ where it has a definite value.

Algorithm: First of all we have to create a 1d array whose elements are from 0 to 2 and the gap between each element is 0.1. Then a loop is has to be used.

code:

```
x = 0:0.1:2
n = numel(x)
for i=1:n
   if x(i)-1 ==0
      continue;
   endif
t= (x(i)^2 -1)/(x(i)-1)
printf("x= %.g and the ans. is = %g\n", x(i), t)
endfor
```

numel: It is an in built function in octave which display the nos. of elements in an object. here numel(x) gives the total nos. of elements in the 1d array x.

%.g = to place the corresponding float value.
%.d = that is for integer value

x = a:b:c gives the ~~rand~~ number from a (starting value) to c (end value) with equal gap b

x = linspace(a:b:c) gives c nos. of numbers having equal gap from the start value a to the end value b

Program: construct a matrix of 3x3 dimension with random values except at diagonal positions where the values are unity.

Algorithm: At first we have to create a 3x3 matrix with randomly choosen elements. Then we have to tell the octave that for the positions $i \neq j$ the program runs continue but for $i=j$ the element is $a_{ij}=1$.

code:

```
a = rand (3,3)
for i = 1:3
  for j = 1:3
    if i != j
      continue;
    endif
  a(i,j) = 1;
  endfor
endfor
a
```

Program: Make the value of x halved repeatedly for 100 times. Stop this process if the value of 1/x exceeds ~~1e6~~ 1 million.

Algorithm: First of all we have to give the value of x. Then a range, which is, 1e6 (1 million), is to be defined. Then a loop is started but when the variable of the loop is greater than range then the running process is stopped.

code: (for loop)

```
clear;
x = 1;
r = 1e6
for i = 1:100
    x = x/2;
    if 1/x > r
        break
    endif
endfor
x
i
```

code: (while loop)

```
clear;
i = 1;
x = 1
r = 1e6;
while (i<100)
    x = x/2;
    if 1/x > r
        break
    endif
    i++;
endwhile
x
i
```

code: (do loop)

```
clear;
i = 1;
x = 1
r = 1e6;
do
    x = x/2
    if 1/x > r
        break;
    endif
    i++
until (i>100)
x
i
```

Program: Assign a random value x within [0,1]. Now generate random values for y continuously within [0,1] unless $|x-y| < 0.01$. Stop the process if more than ~~two~~ 1000 values are required to be generated.

Algorithm: At first we have to generate the array of x. Then the array of y is to be generated under the value of loop variable < 1000. Then calculate $|x-y|$. Now if $|x-y| < 0.01$ then ~~break~~ end the program.

code: (while loop)

```
n=1;
x = rand (1) ;
while (n<1000)
   y = rand(1);
t=abs (x-y);
   if t <0.01
     break
   endif
n=n+1;
endwhile
disp(t)
disp(n)
```

output

octave stores the value of the variable n is 1. Then octave generates a random number. Then it checks if n<1000 or not? if yes then loop runs otherwise not. Let n<1000. Then a random number is generated. Then calculate mod of (x-y). Then check is it <.01 or not. If yes then loop ends otherwise loop runs continuely until t < .01 and in each step n is incremented by +1.

code: (For loop)

```
x= rand (1)
for i = 1:1000
  y=rand (1)
  t= abs (x-y);
  if t < 0.01
    break;
  endif
endfor
t
i
```

output

octave generates a random number. Then the loop runs. First value of loop variable i=1. For i=1, a random number y is generated, t is calculated. Now octave checks if t<.01 or not. if yes then loop ~~ends~~ is ended and if no, then for i=i+1 new y is generated and process goes on until t<0.01.

rand (n): to get a nxn matrix with randomly choosen numbers.

rand (m,n): to get a mxn matrix with randomly choosen numbers.

code: (do loop)

```
x=rand (1)
i = 1;
do
  y=rand (1)
  t= abs (x-y)
  if t<0.01
    break
  endif
i++;
until (i>1000)
i
```

User defined function

The user may create function in octave. Later the user may call or use the function with inserting proper values of the variables. The process or code to build a function is —

Code to build a function:

① Function output = addnumbers (x, y)

z = x + y

output = z

endfunction

output → variable

addnumbers → name of the function

x, y → variable of the function

② Function r = reciprocal (x)

r = 1/x

endfunction

Application

② x = input ('enter the number:')
a = reciprocal (x)

① x = input ('enter the number:')
y = input ('enter the number:')
z = addnumbers (x, y)

output

enter the number: 5
x = 5
r = 0.2000
a = 0.2000

Here it is not necessary that the input file of function building and the input file of its application are saved in same file.

Single input case

Program: Write down a function file Sinsq.m with single input giving rise to the value of $\sin^2 x$ as output use the above Sinsq.m to find the values of $\sin^2 x$ for an equispaced array x within $[0, \pi]$

Algorithm: At first we have to build a function which has the name Sinsq. It takes one variable only. Then we have to make an array x from 0 to 2π. Then using the function and a loop we may get the square of sine values of the array.

Code:

Function building.

```
function y = Sinsq(x)
y = (sin(x))^2;
endfunction
```

Application

```
x = linspace(0, pi/4, 2*pi)
l = numel(x)
for i = 1:l
  y(i) = Sinsq(x(i));
endfor
y
```

multiple input case

Program: Write down the function file sinwt.m with single input t and with single parameter w resulting the function sinwt. Find the value of the function at $t=\pi$ for the parameter values $w = 0, 0.25, 0.5, 0.75, 1.0$.

Algorithm: At first we have to generate a function file having name sinwt.m. Then using it we may get the result.

Code:

Function building

```
function output = sinwt(t,w)
    x = sin(w*t);
    output = x;
endfunction
```

Application

```
t = pi;
w = [0, 0.25, 0.5, 0.75, 1]
a = [ ]
for i = 1 : numel(w)
    a(i) = sinwt(t, w(i));
endfor
a
plot(w, a) ⟶ to get a graphical plot w vs a.
```

Program: write a function file hyp.m to evaluate the function $f(x,y,z) = x^2 - y^2 + z^2$.

Code:

Function building

```
function f = hyp(x,y,z)
    f = x^2 - y^2 + z^2
endfunction
```

Application

```
a = input ('enter the value:')
b = input ('enter the value:')
c = input ('enter the value:')
t = f hyp (a,b,c)
```

Notice that the function variable is x, y, z but in application the variable is a, b, c. Inspite of that the code runs successfully, in the form of the function hyp.m.

Program: Write down a function to find the average of given data stored in an 1d array.

Algorithm:

code:

Function building

```
function a=avg(x)
    Sum=0
    for i=1:numel(x)
      Sum=Sum+x(i);
    endfor
a=Sum/numel(x)
endfunction
```

Application

```
x=1:1:100 input('enter the range:')
m=avg(x)
```

multiple output function

In this case we ~~have~~ input one but we get output many, or more than one. That's why it is multiple output. ~~fun~~

Program: Find the mean and standard deviation of a given 1d array of numbers using a function file

Algorithm: Using in build function 'mean' and 'std' for average and standard deviation we create a function file which gives the two output at a time.

Code:

Function build

```
function [m, sd] = mnsd(x)
    m = mean(x);
    sd = std(x);
    printf
endfunction
```

Application

```
x = input('enter the range:')
[k, l] = mnsd(x);
printf("mean = %d \n", k)
printf("standard deviation
        = %d \n", l)
```

ASS=3.2.1-2

Anonymous function

Program: use anonymous function to find the trace of square of matrix.

code :

```
a = input('enter the matrix:')
f = @ trace(x)
b = f(a)
```

Program: Find the values of a polynomial at different values of independent variable(s) using anonymous function

Code (for one independent variable)

```
a = 2
f = @(x)5*x^2 +3*x -2
b = f(a)
```

code (for more than one independent variable)

```
a = 2
c = 3
f = @(x,y) 5*y^2 + 3*x - 2
b = f(a,c)
```

Note:

f-এর পর bracket-এ যতগুলো variable-এর value দেওয়া থাকবে x আর যতগুলো y সেই ধরনের order-এ লেখা হবে তার উপর base করে, এখানে x=a আর y=c নেওয়া, যদি f = @(y,x) (function) b = f(a,c) লেখা হত তাহলে y=a, x=c হত। যদি f = @(x,y) b = f(c,a) লেখা হতো তাহলে x = c আর y = a হত।