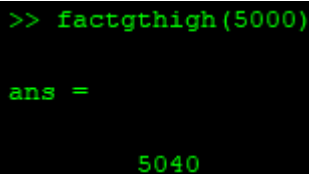


“while ... end” loop

factgthigh.m

```
% function facgt = factgthigh (high)
% % factgthigh returns the first factorial > input
% % Format : factgthigh (inputInteger)
%
% i=0 ;
% fac=1;
% while fac <= high
%     i=i+1;
%     fac = fac * i;
% end
% facgt = fac;
% end
```

Output



```
>> factgthigh(5000)

ans =

    5040
```

Input in “while” loop

whileposnum.m

```
% Prompts the user and echo prints the numbers entered
% until the user enters a negative number
inputnum=input ( ' Enter a positive number: ' ) ;
while inputnum >= 0
    fprintf ( 'You entered a %d. \n\n' , inputnum)
    inputnum = input ( ' Enter a positive number: ' ) ;
end
```

You entered a 6.

```
fprintf ( 'OK! \n' )
```

OK!

Counting in “while” loop

countposnum.m

```
% Prompts the user for positive numbers and echo prints as
% long as the user enters positive numbers
% Counts the positive numbers entered by the user
```

```

counter=0;
inputnum=input ( ' Enter a positive number: ' ) ;
while inputnum >= 0
    fprintf ( 'You entered a %d. \n\n' , inputnum)
    counter = counter+1;
    inputnum = input ( ' Enter a positive number: ' ) ;
end

```

You entered a 4.

You entered a 11.

```

fprintf ( 'Thanks, you entered %d positive numbers. \n' , counter)

```

Thanks, you entered 2 positive numbers.

readonenum.m

```

% Loop until the user enters a positive number
inputnum=input ( 'Enter a positive number: ' ) ;

while inputnum < 0
    inputnum = input ( 'Invalid! Enter a positive number: ! ' ) ;
end
fprintf ( ' Thanks, you entered a %.1f \n' , inputnum)

```

Thanks, you entered a 44.0

```

Enter a positive number: -22
Invalid! Enter a positive number: ! 44

```

Practise question

Practice 5.1

Write a for loop that will print a column of five *'s.

```

% Prints a column of five *'s

n=5;
for i=1:n
    fprintf('* \n');
end

```

```

*
*
*
*
*

```

Practice 5.2

Write a function `prodnnums` that is similar to the `sumnnums` function, but will calculate the product of the numbers entered by the user.

```
% prodnnums calculates the sum of the n numbers entered by the user
n = randi ( [3 10]) ;
runprod = 1 ;
for i = 1:n
    inputnum = input ( 'Enter a number: ' ) ;
    runprod = runprod * inputnum;
end
fprintf('The sum is %.2f \n' , runprod)
```

The sum is 1366180992.00

```
Enter a number: 33
Enter a number: 1.1
Enter a number: 810
Enter a number: 44
Enter a number: 33
Enter a number: 1.1
Enter a number: 810
Enter a number: 11
Enter a number: 4
Enter a number: 2
Enter a number: 3
Enter a number: 4
```

Practice 5.3

For each of the following (they are separate), determine what would be printed. Then, check your answers by trying them in MATLAB.

```
mat = [7  11  3;  3:5];
[r, c] = size(mat);
for i = 1:r
    fprintf('The sum is %d\n', sum(mat(i,:)))
end

-----

for i = 1:2
    fprintf('%d: ', i)
    for j = 1:4
        fprintf('%d ', j)
    end
    fprintf('\n')
end
```

In the first script, variables *r* and *c* store the matrix dimensions 2 and 3. Then *for* loop in the script prints the sum of the rows of the matrix as

The sum is 21

The sum is 12

In the second script, since a *for* loop is called two times using a *for* loop the inner *for* loop is executed two times and prints the following.

1: 1,2,3,4

2: 1,2,3,4

Practice 5.4

Write a function `mymatmin` that finds the minimum value in each column of a matrix argument and returns a vector of the column minimums. An example of calling the function follows:

```
>> mat = randi(20,3,4)
mat =
    15    19    17     5
     6    14    13    13
     9     5     3    13

>> mymatmin(mat)
ans =
     6     5     3     5
```

```
% displays the vector of minimums in columns
% function output=mymatmin(matrix)
% [r, c]=size(matrix);
% output=zeros(1,c);
% for j=1:c
%     output(1,j)=min(matrix(:,j));
% end
```

```
>> mat = randi(20,3,4)

mat =

    14     1    16    10
     7     9    16     9
    20     8     4    13

>> mymatmin(mat)

ans =

     7     1     4     9
```

Practice 5.5

Write a script `avenegnum` that will repeat the process of prompting the user for negative numbers, until the user enters a zero or positive number, as just shown. Instead of `echo` printing them, however, the script will print the average (of just the negative numbers). If no negative numbers are entered, the script will print an error message instead of the average. Use the programming method. Examples of executing this script follow:

```
>>avenegnum
```

Enter a negative number: 5

No negative numbers to average.

```
>>avenegnum
```

Enter a negative number: -8

Enter a negative number: -3

Enter a negative number: -4

Enter a negative number: 6

The average was -5.00

```
% calculates the average of the negative numbers
innum=input ( 'Enter a positive number: ' ) ;
if innum>=0
    fprintf ( 'No negative numbers to average \n' )
end
n=0 ;
sum=0 ;
while innum<0
    sum=sum+innum;
    n=n+1;
    innum=input ( 'Enter a positive number: ' ) ;
end
average=sum/n;
```

```
if average<0
    fprintf ( 'The average is %.2f\n', average) ;
end
```

The average is -5.00

```
Enter a positive number: 5
Enter a positive number: -8
Enter a positive number: -3
Enter a positive number: -4
Enter a positive number: 6
|
```