FOR LOOP

```
for <indexmin:indexstep:indexmax>
   {statements}
end
```

Exercise: Define a vector $\mathbf{z} \in \mathbb{R}^{10} (= \mathbb{R}^{10\times 1})$ s.t. $z_j = 2^j$ for $j = 1, ..., 10$.

Solution. Create a new script wiht the following instructions:

```
for j=1:1:10
z(j)=2^j;
end
z=z'
```

otherwise

```
z=zeros(10,1);
for j=1:1:10
z(j)=2^j;
end
```

Remark. If the array z does not exist, the for-loop creates a row vector, by default.

When `indexstep=1`, it can be skipped: `for j=1:10`

Save the script with the name `test1.m`. From the matlab-command window:

```
>> test1
```

Exercise: Create the Hilbert matrix of size 5×5. It is defined as follows: $H_{ij} = \frac{1}{i+j-1}$, for $i = 1, ..., 5$ and $j = 1, ..., 5$.
Solution. Create a new script with the following instructions:

```
for i=1:5
for j=1:5
H(i,j)=1/(i+j-1);
end
end
disp('The Hilbert matrix of size 5x5 is:'), H
```

How to compute a sum with a for-loop.

Ex. Given $n = 10$, compute $s = \sum_{i=1}^{n} \frac{1}{i}$.

In our mind we do: $s = 0; s = s + 1; s = s + 1/2; s = s + 1/3; .....$
$s = s + 1/n;$

In the editor window

```
% compute the sum
n=10; s=0;
for i=1:n
s=s+1/i;
end
fprintf('The sum is %8.6f \n',s)
```

`fprintf` is the command to print with format,
`%8.6f` is the format type I want to use to print `s`
`\n` says to matlab to begin a new paragraph.

Remark. When we compute a sum whose addends have all the same sign, but range from very small to very large values, it is better to start the sum from the smaller numbers:

```
s=0; for i=20:-1:1 s=s+1/i; end
```

Remark. When we compute a sum whose addends have different sign, it would be better to separate positive and negative numbers and to compute two partial summations, then add them at the end.

Matlab scripts and functions

m-file or **script**
is a list
of MATLAB commands

function
is a program unit
with input and output

GLOBAL ← VARIABLES → *LOCAL*

the file extension is .m in both cases.
How to run a script (m-file):

`>>filename` ↵

How to run a function:

`>>[out]=filename(in)` ↵

where `out`=is the list of output parameters
`in`=is the list of input parameters. I suggest to give to the function the same name of filename.m

Now we write a function to compute a sum of n numbers

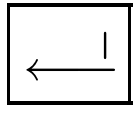
Input: `n`

Output: `s`.

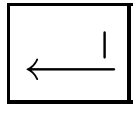
In the editor window:

```
function [s]=summation(n)
s=0;
for i=n:-1:1
s=s+1/i;
fprintf('The sum is %8.6f \n',s)
end
```

From the Matlab-command window:

```
>> s=summation(20);
```


or

```
>> n=20; s=summation(n);
```


IF, IF .. ELSE, IF... ELSEIF ... ELSE

```
if <condition>
  {statements}
end

if <condition>
  {statement 1}
else
  {statement 2}
end

if <condition 1>
  {statement 1}
elseif<condition 2>
  {statement 2}
else
  {statement 3}
end
```

Relational operators

<	less than
>	grather than
<=	less than or equal to
>=	grather than or equal to
==	equal to
~=	different from

Remark. ~: ALT 126 Logical operators

&	and
|	or
~	not

Example: If $0 < x < 1$ then compute and print $y = 1/x$

```
if x>0 & x<1
y=1/x ; fprintf('x= %f, 1/x=%f\n',x,y)
end
```

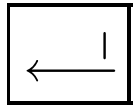
Example: If $x < 0$ or $x \geq 10$ then compute and print $y = sin(x)$ otherwise compute and print $y = cos(x)$.

```
if x<0 | x>=10
y=sin(x); fprintf('x= %f, sin(x)=%f\n',x,y)
else
y=cos(x); fprintf('x= %f, cos(x)=%f\n',x,y)
end
```

Example: Compute the factorial of a positive integer number (without recursion):

```
function [f]=factorial(n)
% Compute the factorial of a positive integer number
if n< 0
f=[];
warning('The number is negative')
elseif n==0
f=1;
else
f=prod(1:n);
end
```

```
>>n=...; f=factorial(n);
```



WHILE -loop

```
while <condition>
  {statements}
end
```

Example. Compute the smallest floating-point number $\epsilon_M > 1$ such that $1 + \epsilon_M > 1$. This number is called machine-precision

Solution. We set $\epsilon_M = 1$,
loop: compute $1 + \epsilon_M$ and if $1 + \epsilon_M > 1$ we divide ϵ_M by 2.
At the end, we have to multiply ϵ_M by 2 (the last value s.t. $1 + \epsilon_M > 1$.

```
epsilon=1;
while epsilon+1 > 1
epsilon=epsilon/2;
end
epsilon=epsilon*2;
```

Remark. Inside the while-loop we must modify the variable used in the while-condition (`epsilon` in our case), otherwise the loop does not stop.
In any case, to stop an infinite loop: CTRL+C from the Matlab command window.

Exercise. Write a function to compute the product between two matrices.
Given $A \in \mathbb{R}^{n \times m}$, and $b \in \mathbb{R}^{m \times p}$ compute

$$C = A\ B : c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

How to create a movie

Plot $f(x, y, t) = \frac{1}{5}\sin(x)y\cos(t)$
when $(x, y) \in [-\pi, \pi]^2$ and $t \in [0, 2\pi]$

```
[x,y]=meshgrid(-pi:.5:pi);
f=@(x,y,t)[sin(x).*y/5*cos(t)];
nframes=20;
tt=linspace(0,2*pi,nframes);
figure(1); clf
Mv=moviein(nframes);
for n=1:nframes
  t=tt(n); z=f(x,y,t); surf(x,y,z);
  axis([-pi pi -pi pi -1 1]);
  Mv(:,n) = getframe;
end
movie(Mv,4);
```