Procesamiento Digital de Imágenes (PDI)

Universidad Nacional de Asunción

M. Sc. José Luis Vázquez Noguera

jlvazquez@pol.una.py

2da Clase de Laboratorio

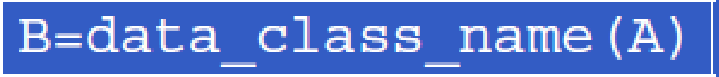
Capitulo 2 (Cont.)

Fundamentos

Contenido

- Conversión entre clases de datos y tipos de imágenes
- La indexación de matrices
- Algunas matrices estándar importantes
- Introducción a la programación de scripts y funciones

Conversión entre clases de datos



```
B=data_class_name(A)
```

Si C es un arreglo de clase double en el cuál todos los valores están en el rango [0 255], se convierte a un arreglo uint8 con el comando
D = uint8(C).

Si un arreglo de clase double tiene valores fuera del rango [0 255], y se convierte a clase uint8, Matlab convierte a 0 todos los valores que son menores que 0, y convierte a 255 todos los valores que son mayores que 255.

Conversión entre clases de datos y tipos de imágenes

Name	Converts Input to:	Valid Input Image Data Classes
im2uint8	uint8	logical, uint8, uint16, and double
im2uint16	uint16	logical, uint8, unit16, and double
mat2gray	double (in range [0,1])	double
im2double	double	logical, uint8, uint16, and double
im2bw	logical	uint8, uint16, and double

Conversión entre clases de datos y tipos de imágenes

```
>> f=[-0.5 0.5;0.75 1.5]

f =

   -0.5000    0.5000
    0.7500    1.5000

>> g=im2uint8(f)

g =

    0  128
  191  255
```

Conversión entre clases de datos y tipos de imágenes

```
>> h=uint8([25 50; 128 200]);
>> g=im2double(h)

g =

    0.0980    0.1961
    0.5020    0.7843
```

Conversión entre clases de datos y tipos de imágenes

```
>> f=[1 2; 3 4];
>> g=mat2gray(f)

g =

         0    0.3333
    0.6667    1.0000

>> gb=im2bw(g, 0.6)

gb =

     0     0
     1     1
```

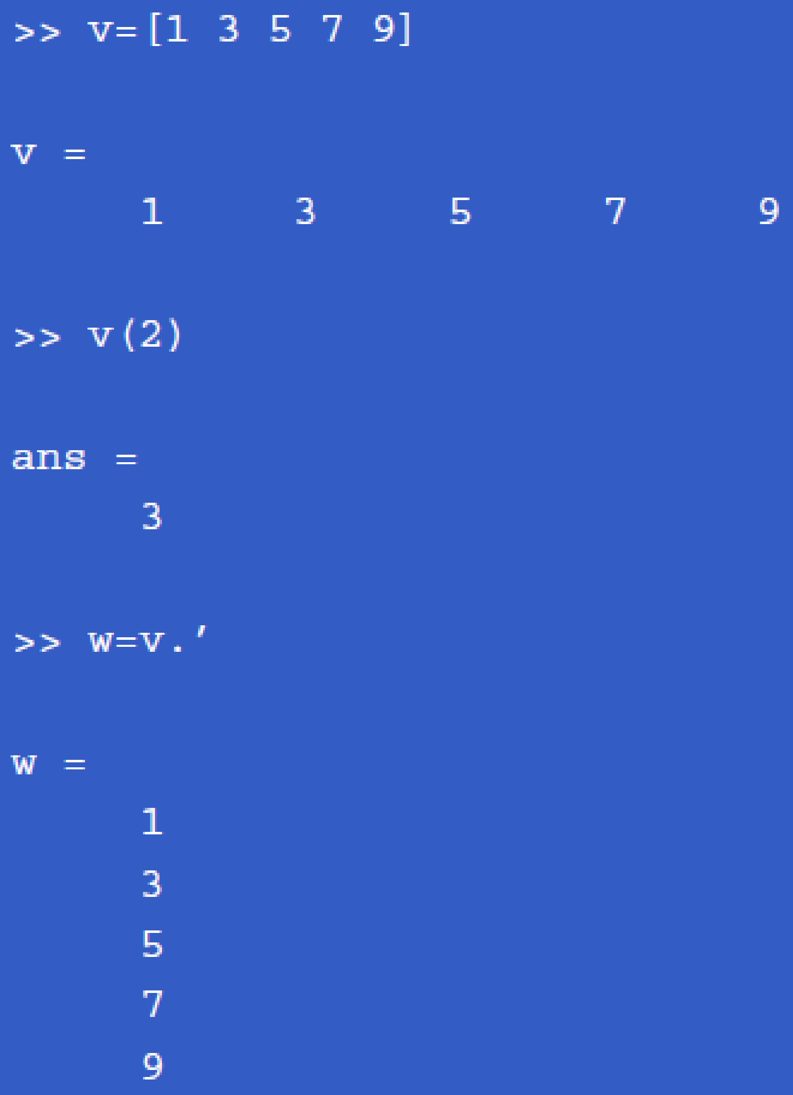
Conversión entre clases de datos y tipos de imágenes

```
>> gb=f>2

gb =

     0     0
     1     1

>> gbv=islogical(gb)

gbv =

     1
```

La indexación de matrices

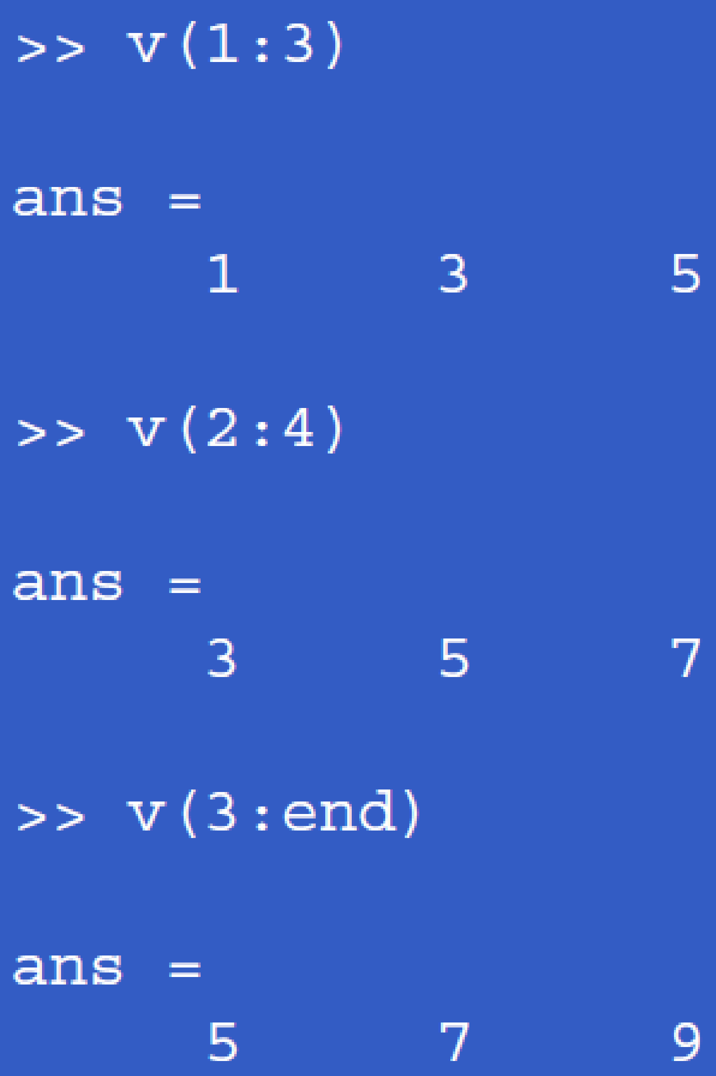
- La indexación de vectores
- La indexación de matrices
- Seleccionando la dimensión de la matriz

La indexación de vectores



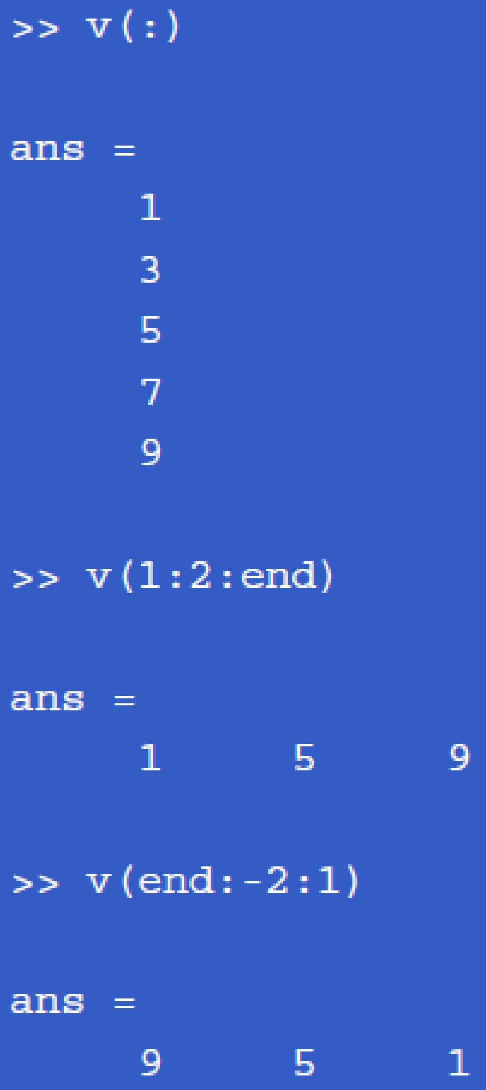
```
>> v=[1 3 5 7 9]

v =
     1     3     5     7     9

>> v(2)

ans =
     3

>> w=v.'

w =
     1
     3
     5
     7
     9
```

La indexación de vectores



```
>> v(1:3)

ans =
     1     3     5

>> v(2:4)

ans =
     3     5     7

>> v(3:end)

ans =
     5     7     9
```

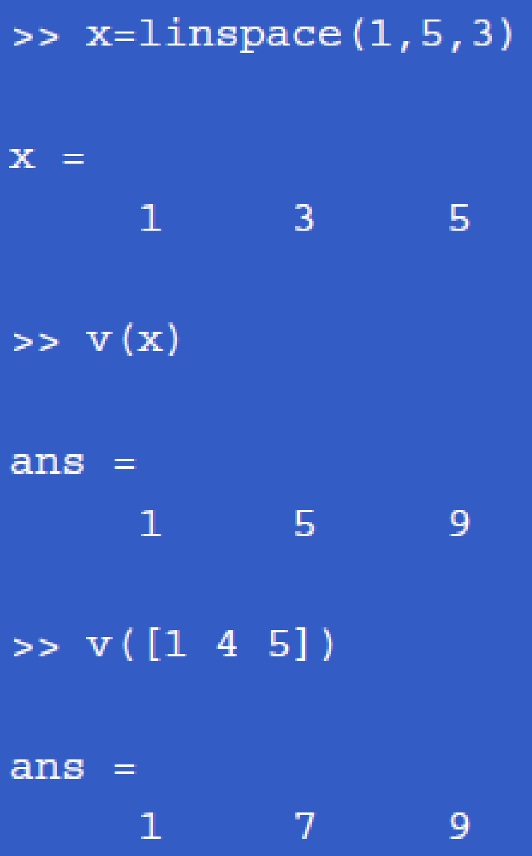
La indexación de vectores



```
>> v(:)

ans =
     1
     3
     5
     7
     9

>> v(1:2:end)

ans =
     1     5     9

>> v(end:-2:1)

ans =
     9     5     1
```

La indexación de vectores

```
linspace(a, b, n)
```

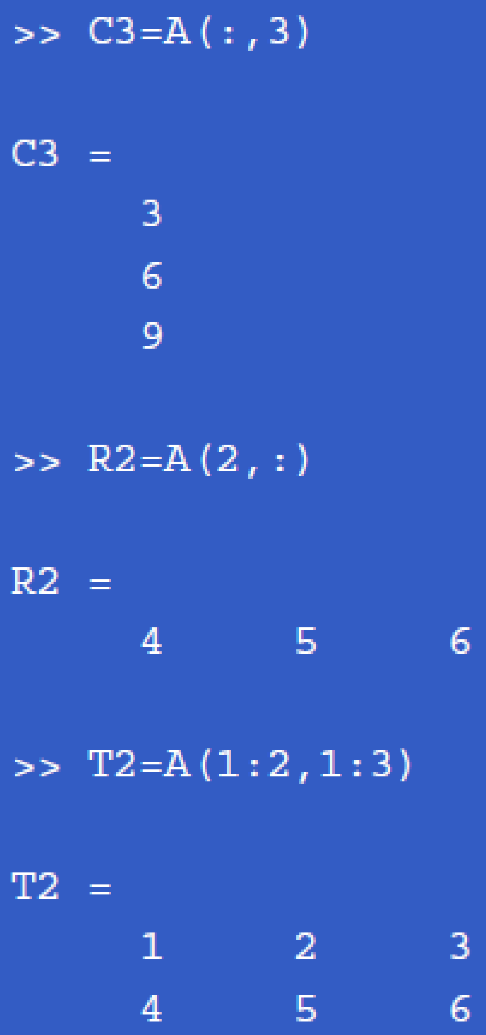


```
>> x=linspace(1,5,3)

x =
     1     3     5

>> v(x)

ans =
     1     5     9

>> v([1 4 5])

ans =
     1     7     9
```

La indexación de matrices

```
>> A=[1 2 3; 4 5 6; 7 8 9]

A =
     1     2     3
     4     5     6
     7     8     9

>> A(2,3)

ans =
     6
```

La indexación de matrices

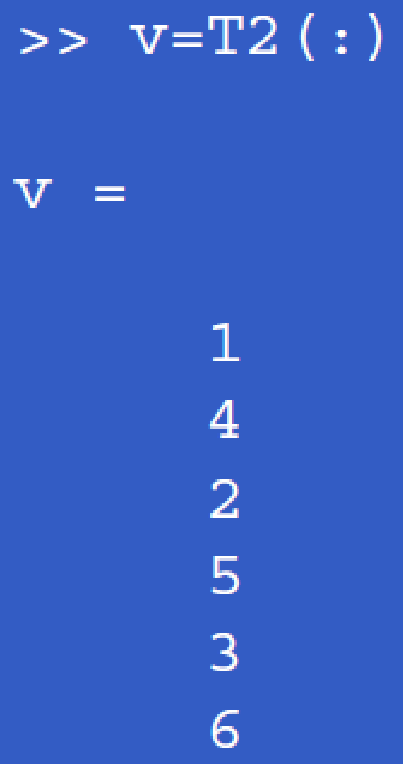


```
>> C3=A(:,3)

C3 =
     3
     6
     9

>> R2=A(2,:)

R2 =
     4     5     6

>> T2=A(1:2,1:3)

T2 =
     1     2     3
     4     5     6
```

La indexación de matrices

```
>> D=logical([1 0 0; 0 0 1; 0 0 0])

D =

     1     0     0
     0     0     1
     0     0     0

>> A(D)

ans =

     1
     6
```

La indexación de matrices



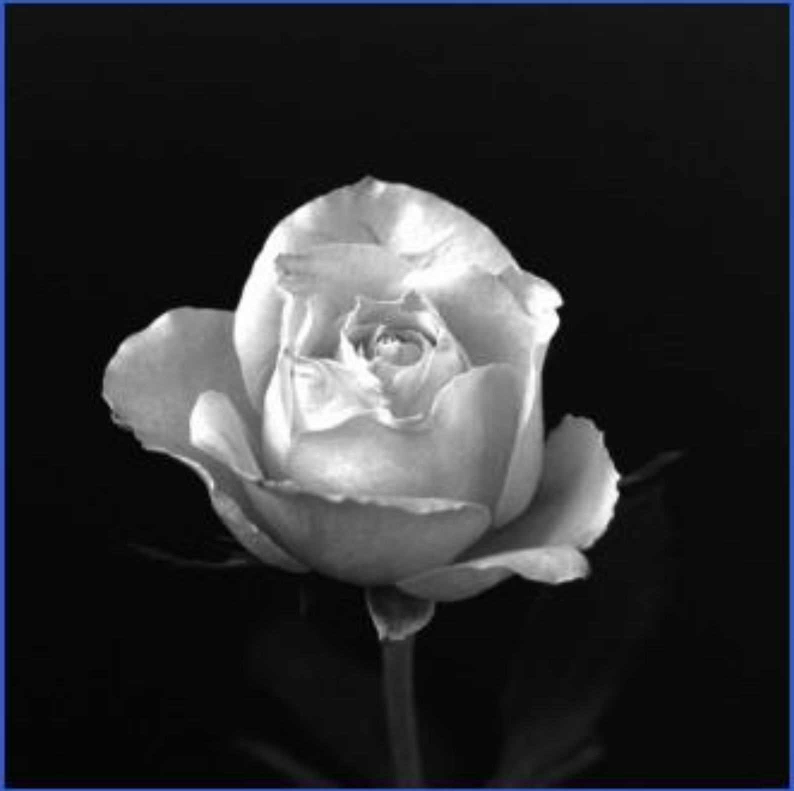
```
>> v=T2(:)

v =

     1
     4
     2
     5
     3
     6
```

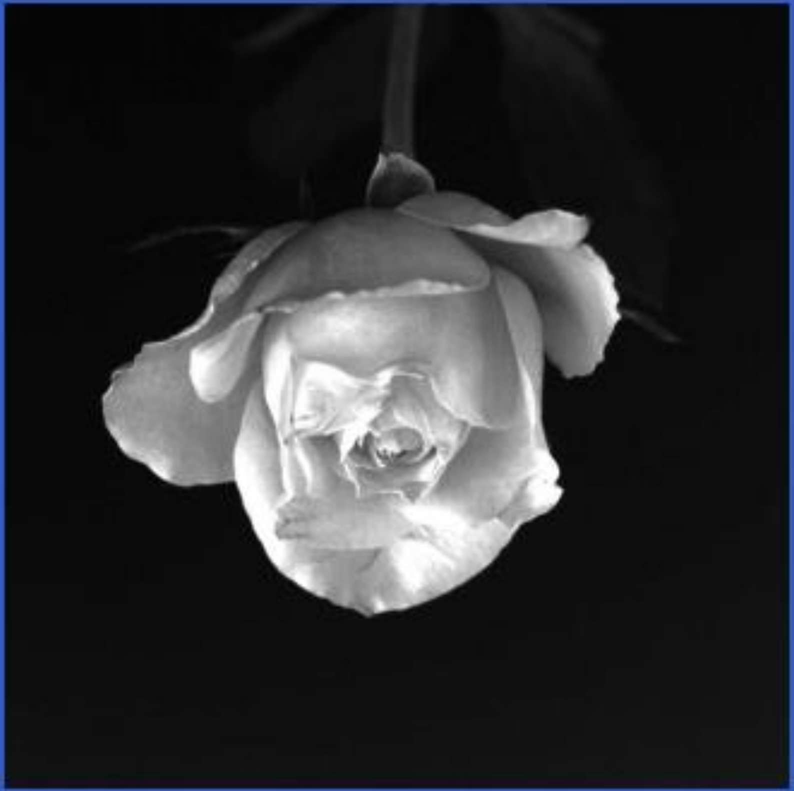
La indexación de matrices

```
>> s=sum(A(:))

s =
    45

>> s1=sum(A)

s1 =
    12      15      18

>> s2=sum(sum(A))

s2 =
    45
```

La indexación de matrices

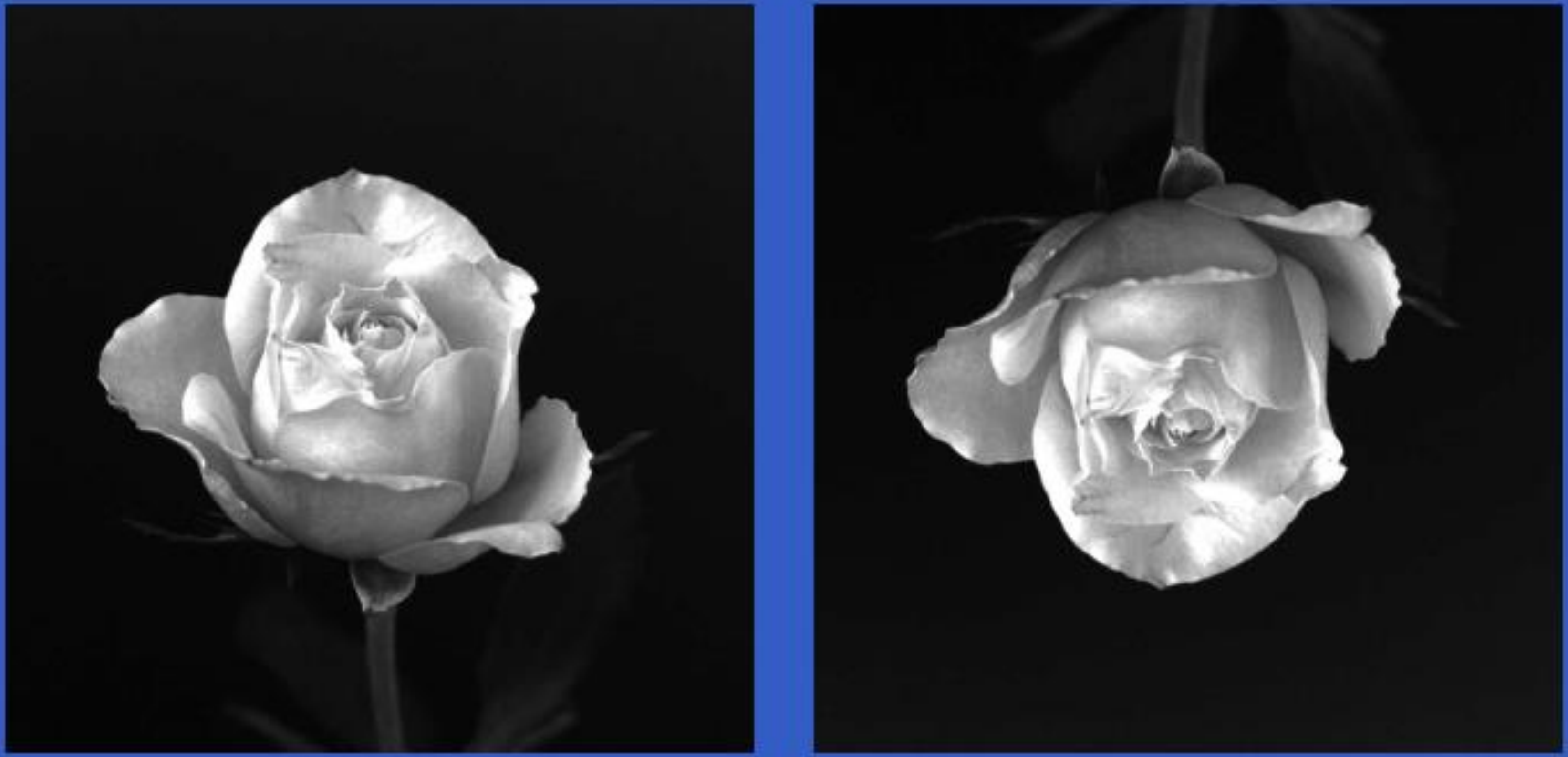
```
>> f=imread(’rose.tif’);
>> fp=f(end:-1:1,:);
```





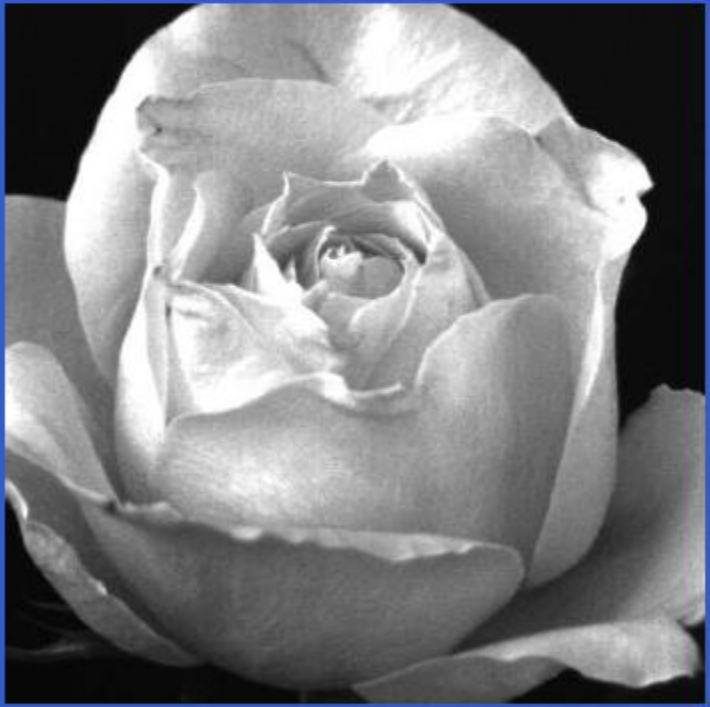
La indexación de matrices

```
>> f=imread(’rose.tif’);
>> fp=f(end:-1:1,:);
```



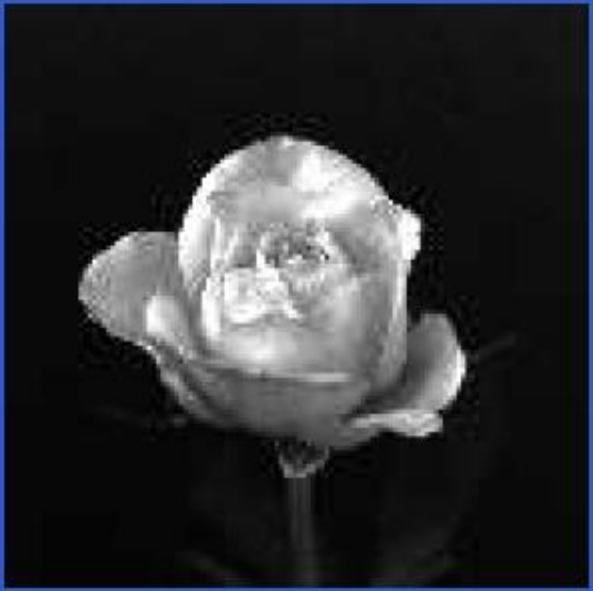
La indexación de matrices

```
>> fc=f(257:768,257:768);
```



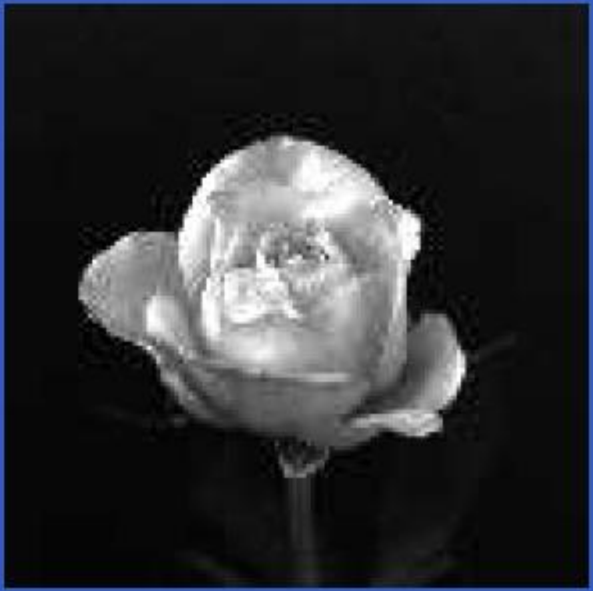
La indexación de matrices

```
>> fs=f(1:8:end,1:8:end);
```

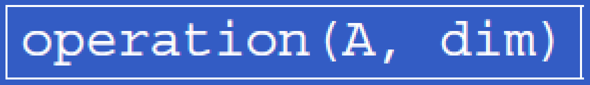


La indexación de matrices

```
>> fs=f(1:8:end,1:8:end);
```

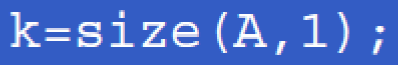


Seleccionando la dimensión de la matriz



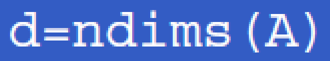
```
operation(A, dim)
```

Donde operation denota un operación aplicable en Matlab, A es una matriz y dim es un escalar.



```
k=size(A,1);
```

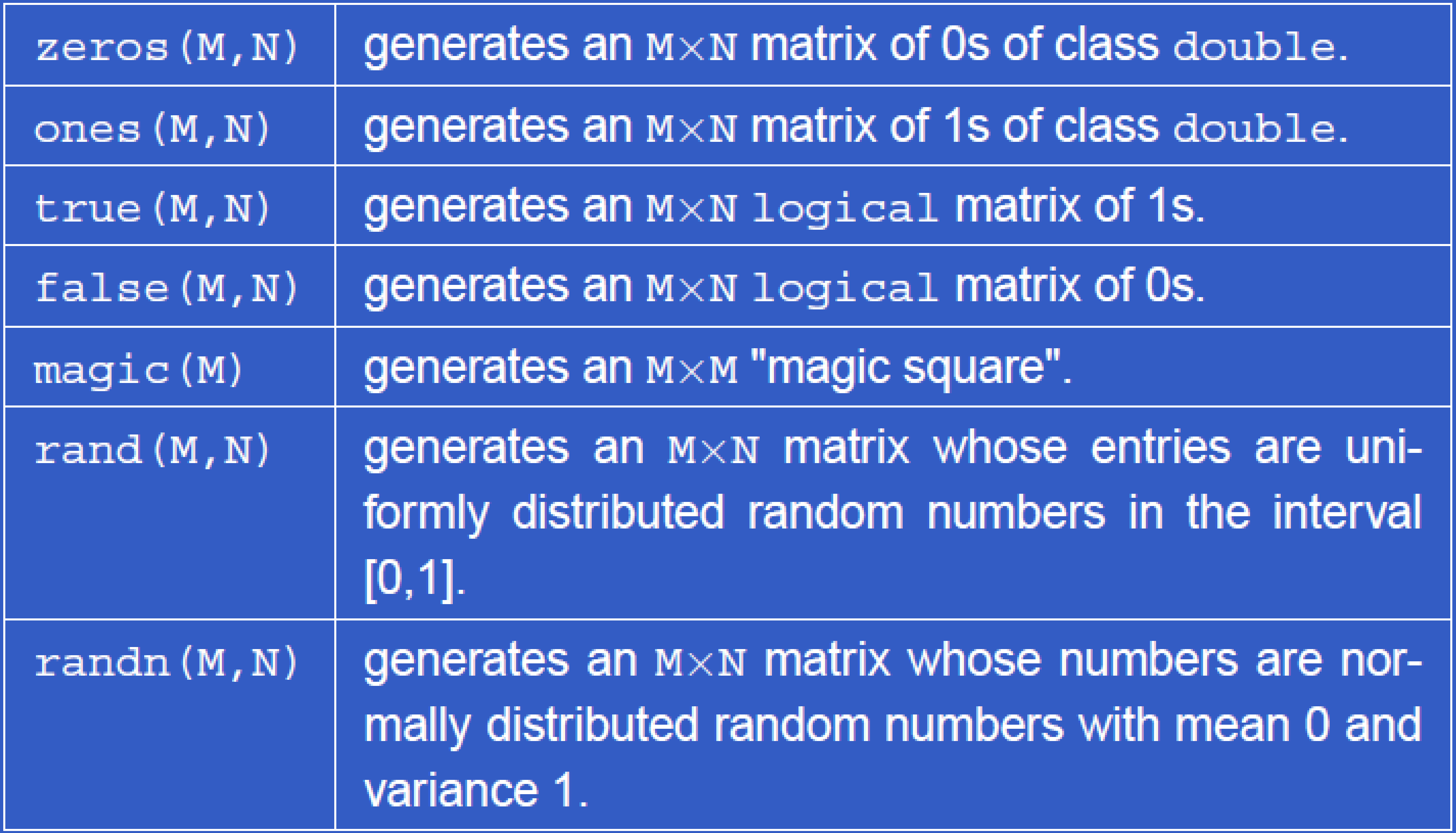
Obtiene el tamaño de A en su primera dimensión.



```
d=ndims(A)
```

La función ndims, obtiene el número de dimensiones de la matriz A

Algunas matrices estándar importantes



`zeros(M,N)`	generates an $M \times N$ matrix of 0s of class `double`.
`ones(M,N)`	generates an $M \times N$ matrix of 1s of class `double`.
`true(M,N)`	generates an $M \times N$ `logical` matrix of 1s.
`false(M,N)`	generates an $M \times N$ `logical` matrix of 0s.
`magic(M)`	generates an $M \times M$ "magic square".
`rand(M,N)`	generates an $M \times N$ matrix whose entries are uniformly distributed random numbers in the interval [0,1].
`randn(M,N)`	generates an $M \times N$ matrix whose numbers are normally distributed random numbers with mean 0 and variance 1.

Algunas matrices estándar importantes

```
>> A=5*ones(3)
A =
     5     5     5
     5     5     5
     5     5     5

>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2

>> B=rand(2,4)
B =
    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185
```

Introducción a a la programación de scripts y funciones

- Archivos M
- Operadores
- Control de Flujo

Archivos M

Archivos M en Matlab pueden ser

- **Scripts** que simplemente ejecuta una serie de sentencias Matlab

- **Funciones** que puede aceptar argumentos y puede producir una o más salidas

Archivos M

Los componentes de un Archivo M son

- La línea de definición de la función
- Una línea H1
- Texto de ayuda
- El cuerpo de la función
- Comentarios adicionales.

Archivos M

```
function s = supcirc(r)
% SUPCIRC Superficie de un circulo
% SUPCIRC(R), donde R es una matriz, calcula la superficie de los círculos
% cuyos radios son los coeficientes de la matriz.

%Valor de pi es 3,14
s = pi * r.^2;
```

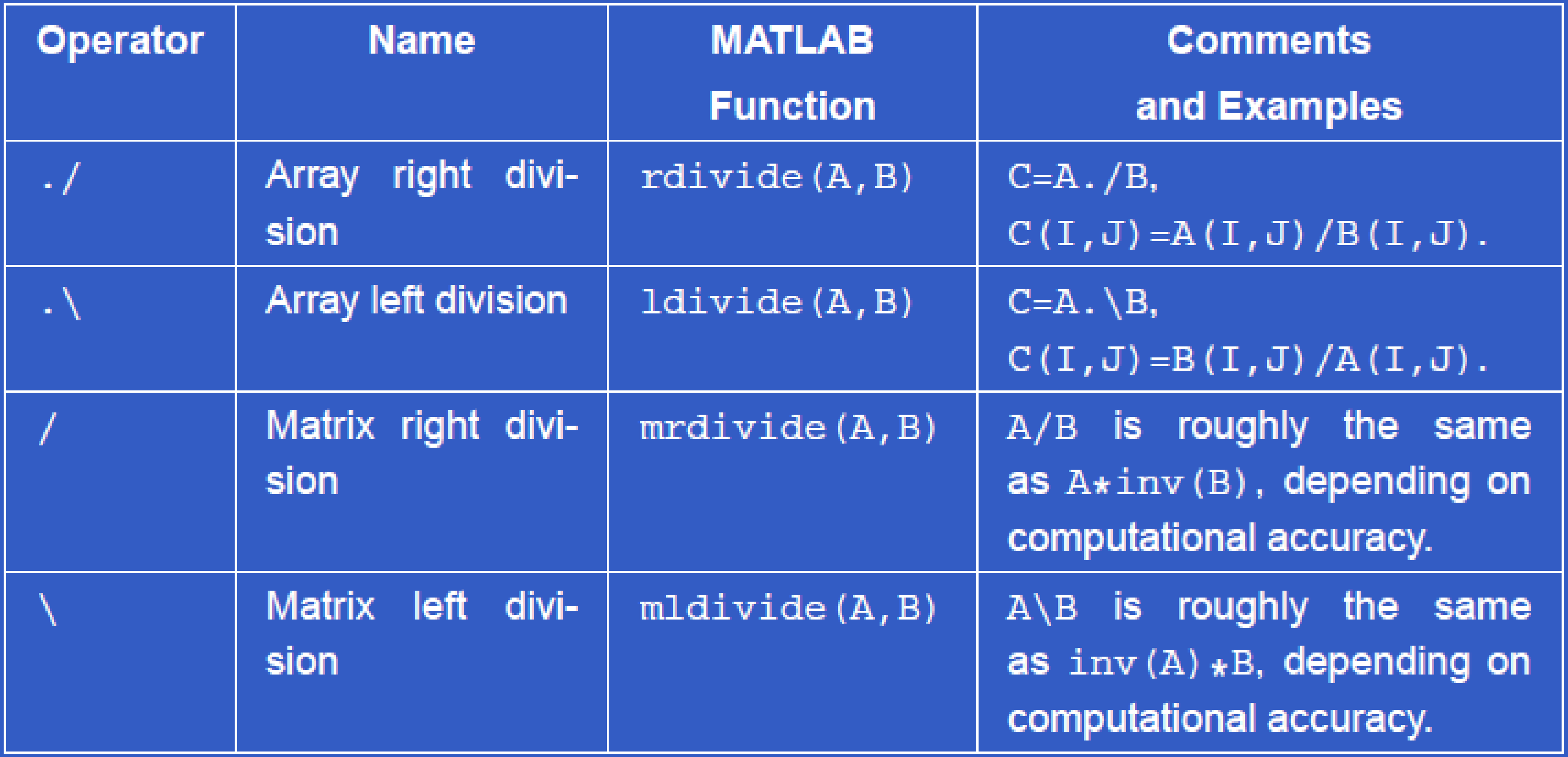
Operadores

- Operadores aritméticos
- Operadores relacionales
- Operadores logicos

Operadores aritméticos

Operator	Name	MATLAB Function	Comments and Examples
+	Array and matrix addition	plus(A,B)	a+b, A+B, or a+A.
-	Array and matrix subtraction	minus(A,B)	a-b, A-B, A-a.
.*	Array multiplication	times(A,B)	C=A.*B, C(I,J)=A(I,J)*B(I,J).
*	Matrix multiplication	mtimes(A,B)	A*B, standard matrix multiplication, or a*A, multiplication of a scalar times all elements of A.

Operadores aritméticos



Operator	Name	MATLAB Function	Comments and Examples
./	Array right division	rdivide(A,B)	C=A./B, C(I,J)=A(I,J)/B(I,J).
.\	Array left division	ldivide(A,B)	C=A.\B, C(I,J)=B(I,J)/A(I,J).
/	Matrix right division	mrdivide(A,B)	A/B is roughly the same as A*inv(B), depending on computational accuracy.
\	Matrix left division	mldivide(A,B)	A\B is roughly the same as inv(A)*B, depending on computational accuracy.

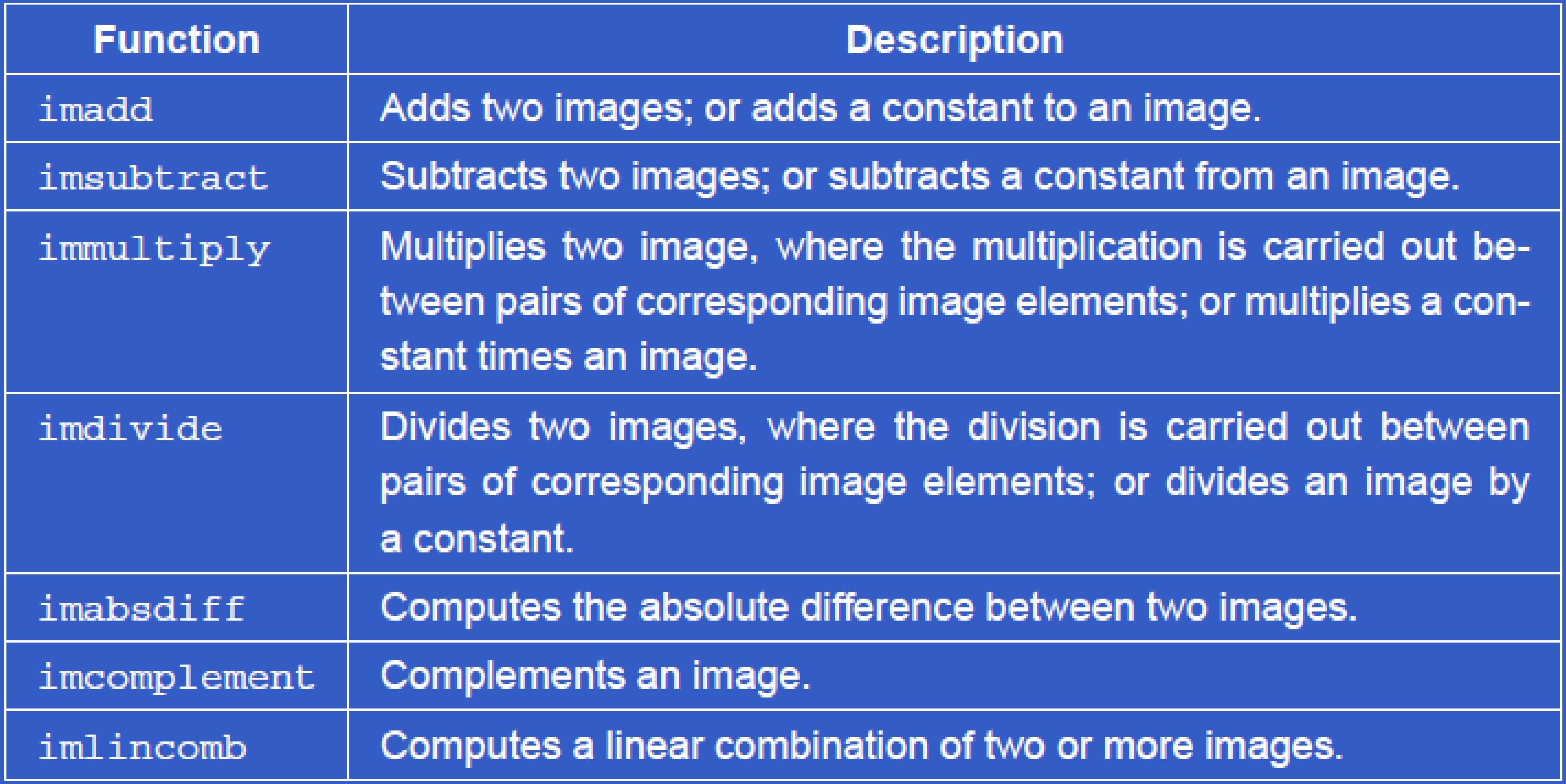
Operadores aritméticos

Operator	Name	MATLAB Function	Comments and Examples
`.^`	Array power	`power(A,B)`	If `C=A.^B`, then `C(I,J)=A(I,J)^B(I,J)`.
`^`	Matrix power	`mpower(A,B)`	Square matrix to the scalar power, or scalar to the square matrix power.
`.'`	Vector and matrix transpose	`transpose(A)`	`A.'`. Standard vector and matrix transpose.
`'`	Vector and matrix complex conjugate transpose	`ctranspose(A)`	`A'`. Standard vector and matrix conjugate transpose.

Operadores aritméticos

Operator	Name	MATLAB Function	Comments and Examples
`.^`	Array power	`power(A,B)`	If `C=A.^B`, then `C(I,J)=A(I,J)^B(I,J).`
`^`	Matrix power	`mpower(A,B)`	Square matrix to the scalar power, or scalar to the square matrix power.
`.'`	Vector and matrix transpose	`transpose(A)`	`A.'.` Standard vector and matrix transpose.
`'`	Vector and matrix complex conjugate transpose	`ctranspose(A)`	`A'.` Standard vector and matrix conjugate transpose.

Funciones aritméticas entre imágenes

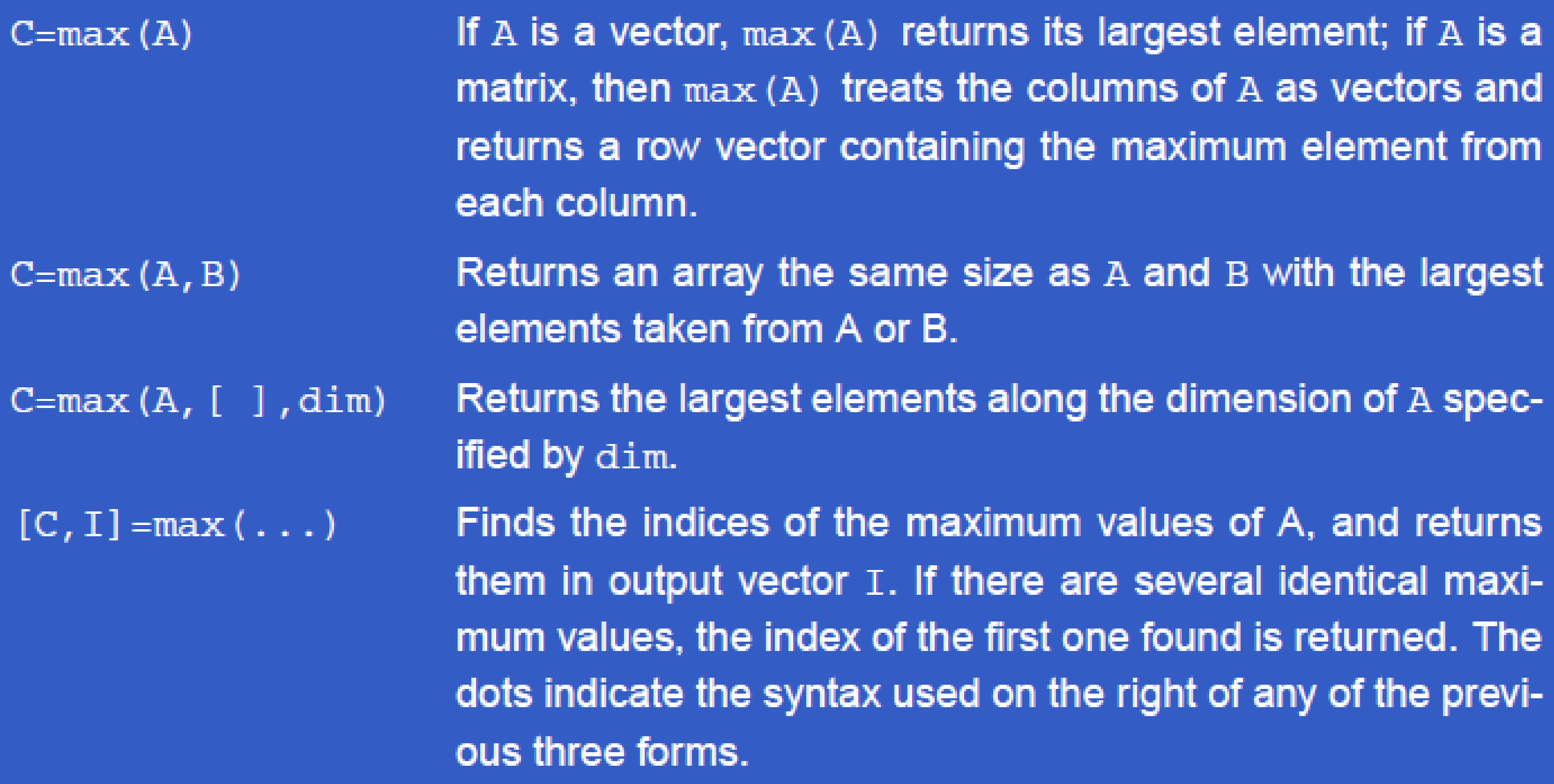


Function	Description
`imadd`	Adds two images; or adds a constant to an image.
`imsubtract`	Subtracts two images; or subtracts a constant from an image.
`immultiply`	Multiplies two image, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
`imdivide`	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
`imabsdiff`	Computes the absolute difference between two images.
`imcomplement`	Complements an image.
`imlincomb`	Computes a linear combination of two or more images.

Un ejemplo

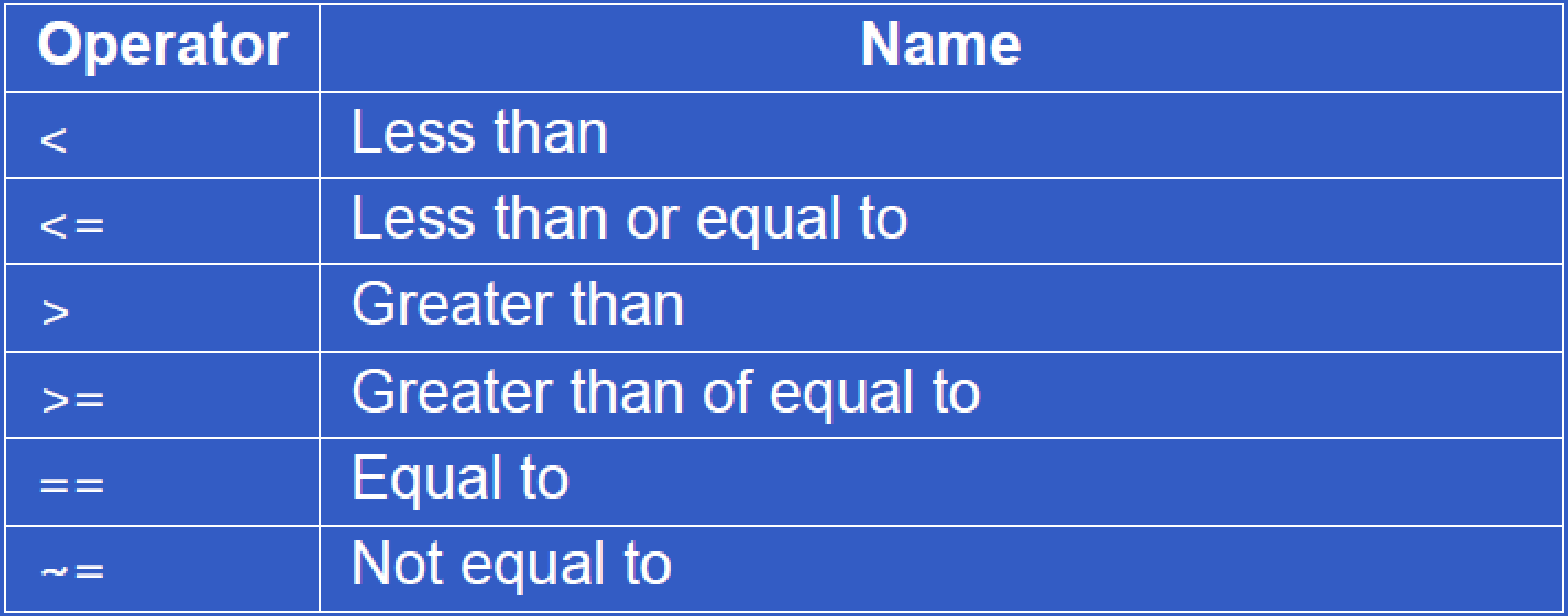
```
function [p, pmax, pmin, pn] = improd(f, g)
%IMPROD Compute the product of two images.
% [P, PMAX, PMIN, PN] = IMPROD(F, G) outputs the
element-by-element % product of two input images, F and G,
 the product maximum and % minimum values, and a
normalized product array with values in the % range [0, 1].
The input images must be of the same size. They % can be
 of class uint8, unit16, or double. The outputs are of % class double.

fd = double(f);
gd = double(g);
p = fd.*gd;
pmax = max(p(:));
 pmin = min(p(:));
pn = mat2gray(p);
```

Acerca de max



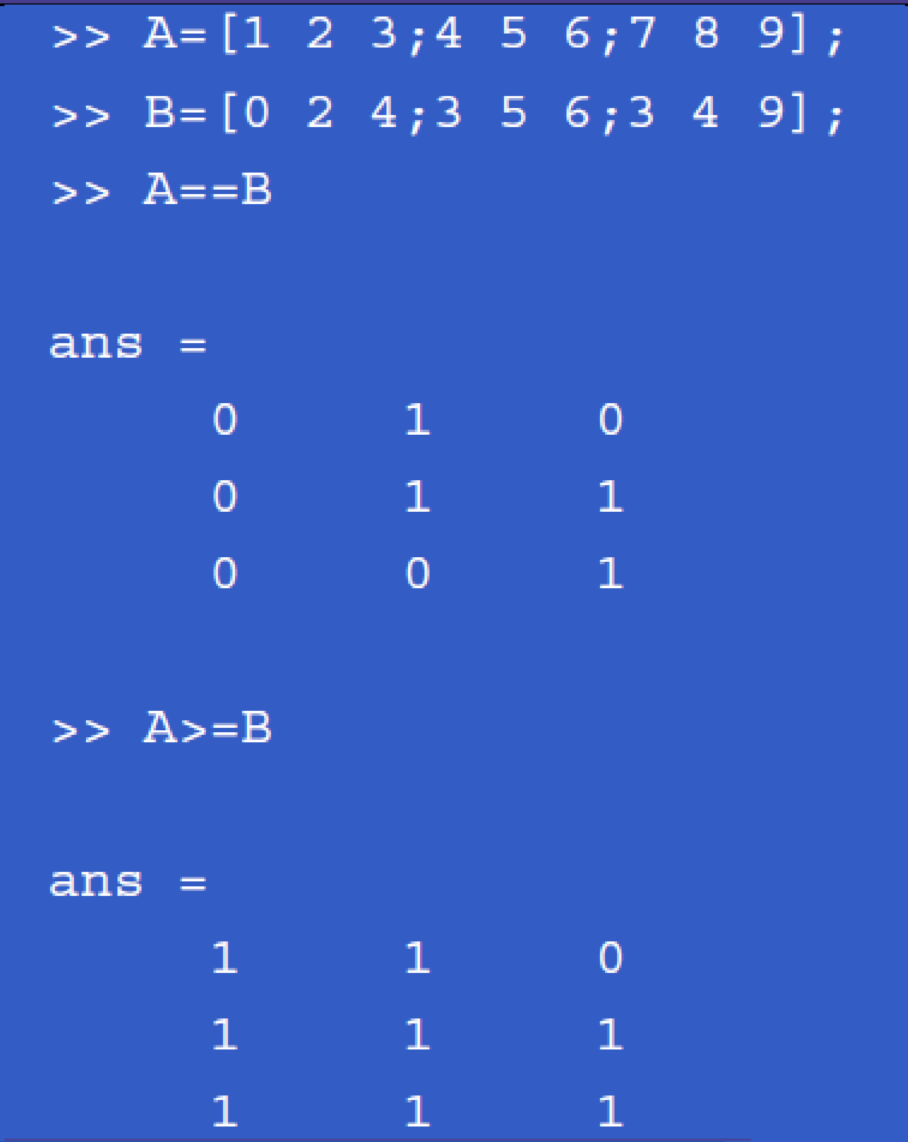
`C=max(A)`	If `A` is a vector, `max(A)` returns its largest element; if `A` is a matrix, then `max(A)` treats the columns of `A` as vectors and returns a row vector containing the maximum element from each column.
`C=max(A,B)`	Returns an array the same size as `A` and `B` with the largest elements taken from A or B.
`C=max(A,[ ],dim)`	Returns the largest elements along the dimension of `A` specified by `dim`.
`[C,I]=max(...)`	Finds the indices of the maximum values of A, and returns them in output vector `I`. If there are several identical maximum values, the index of the first one found is returned. The dots indicate the syntax used on the right of any of the previous three forms.

Operadores relacionales



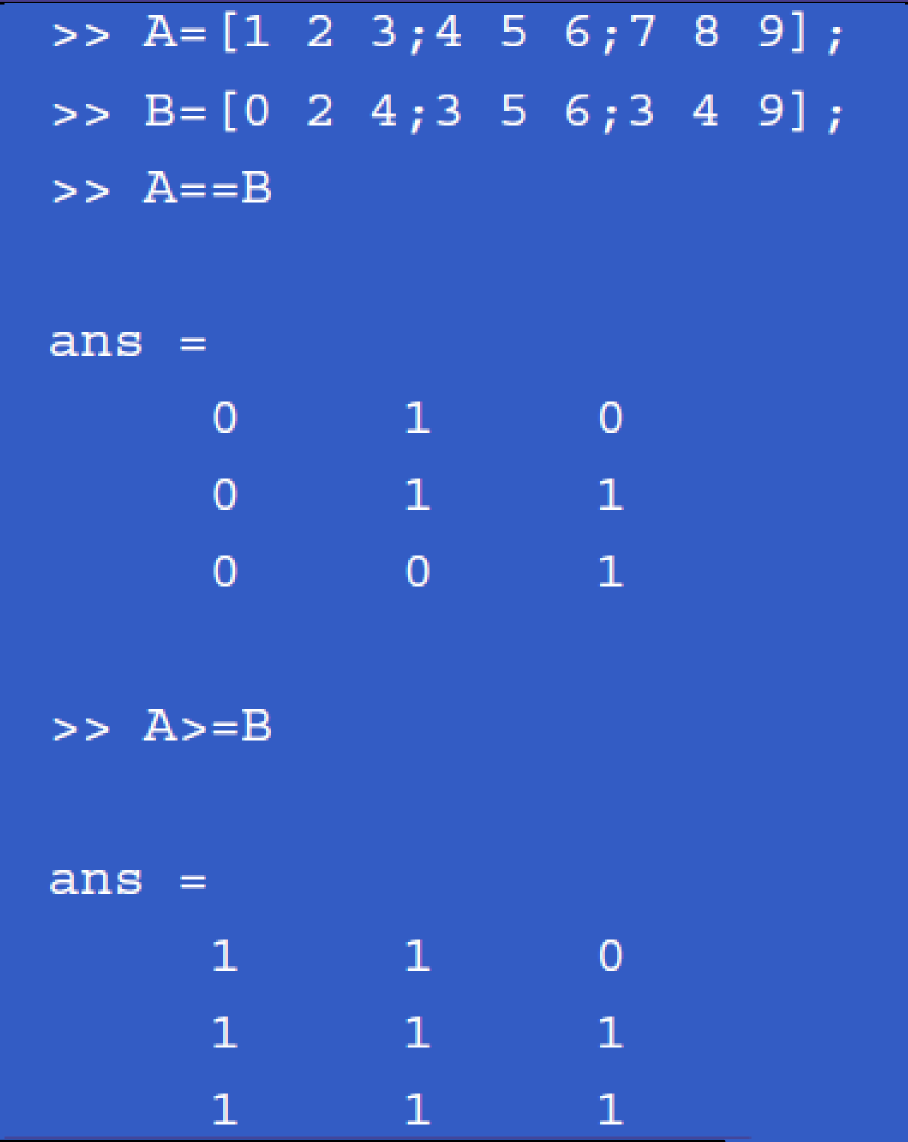
Operator	Name
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than of equal to
==	Equal to
~=	Not equal to

Operadores relacionales



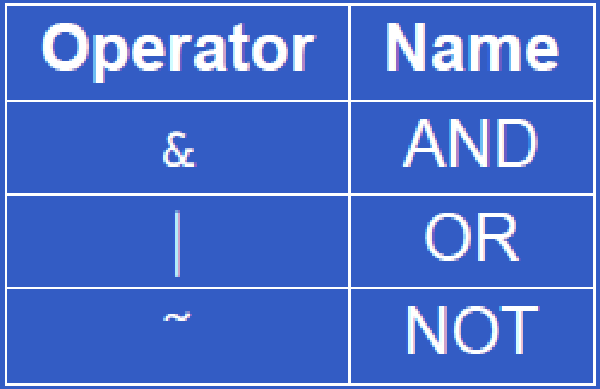
```
>> A=[1 2 3;4 5 6;7 8 9];
>> B=[0 2 4;3 5 6;3 4 9];
>> A==B

ans =
     0     1     0
     0     1     1
     0     0     1


>> A>=B


ans =
     1     1     0
     1     1     1
     1     1     1
```

Operadores relacionales



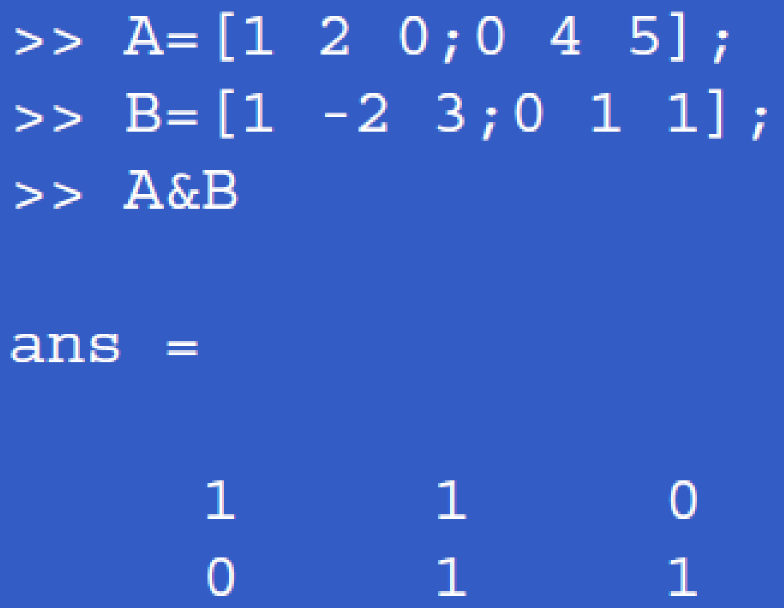
```
>> A=[1 2 3;4 5 6;7 8 9];
>> B=[0 2 4;3 5 6;3 4 9];
>> A==B

ans =
     0     1     0
     0     1     1
     0     0     1


>> A>=B

ans =
     1     1     0
     1     1     1
     1     1     1
```

Operadores logicos



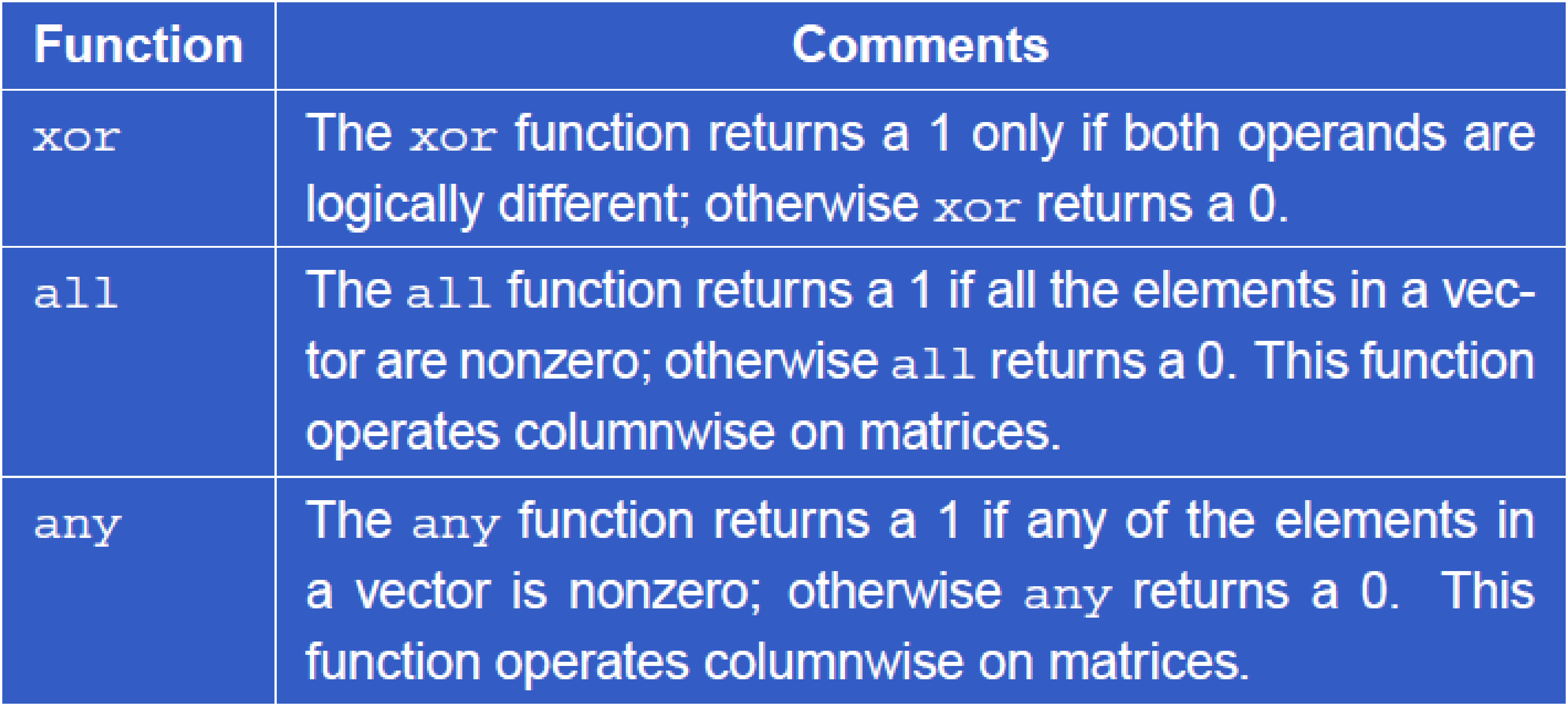
Operator	Name
&	AND
|	OR
~	NOT

Operadores logicos



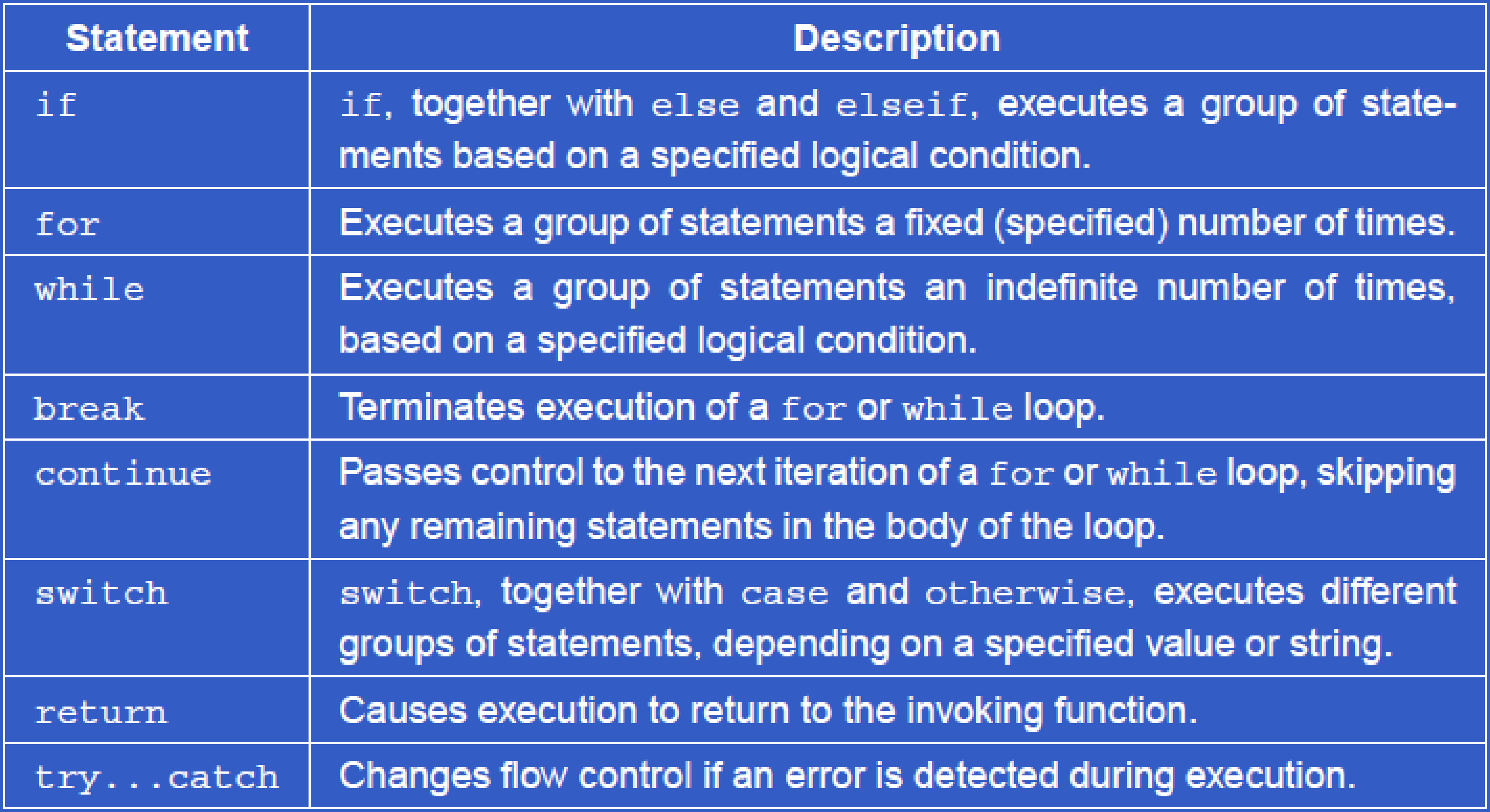
```
>> A=[1 2 0;0 4 5];
>> B=[1 -2 3;0 1 1];
>> A&B

ans =

     1     1     0
     0     1     1
```

Funciones lógicas



Function	Comments
`xor`	The `xor` function returns a 1 only if both operands are logically different; otherwise `xor` returns a 0.
`all`	The `all` function returns a 1 if all the elements in a vector are nonzero; otherwise `all` returns a 0. This function operates columnwise on matrices.
`any`	The `any` function returns a 1 if any of the elements in a vector is nonzero; otherwise `any` returns a 0. This function operates columnwise on matrices.

Control de Flujo



Statement	Description
`if`	`if`, together with `else` and `elseif`, executes a group of statements based on a specified logical condition.
`for`	Executes a group of statements a fixed (specified) number of times.
`while`	Executes a group of statements an indefinite number of times, based on a specified logical condition.
`break`	Terminates execution of a `for` or `while` loop.
`continue`	Passes control to the next iteration of a `for` or `while` loop, skipping any remaining statements in the body of the loop.
`switch`	`switch`, together with `case` and `otherwise`, executes different groups of statements, depending on a specified value or string.
`return`	Causes execution to return to the invoking function.
`try...catch`	Changes flow control if an error is detected during execution.

If, else, y elseif

```
if expression
    statements
end

if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

If, else, y elseif

```
function av=average(A)
%AVERAGE Computes the average value of an array.
%  AV=AVERAGE(A) computes the average value of
%  input array, A, which must be a 1-D or 2-D
%  array.

% Check the validity of the input. (Keep in mind
% that a 1-D array is a special case of a 2-D
% array.)
if ndims(A)>2
    error(’The dimensions of the input cannot exceed 2.’)
end

%Compute the average
av=sum(A(:))/length(A(:));
%or av=sum(A(:))/numel(A);
```

for

```
for index=start:increment:end
      statements
end
```

for

```
count=0;
for k=0:0.1:1
    count=count+1;
end
```

for

```
function s=subim(f,m,n,rx,cy)
%SUBIM Extracts a subimage, s, from a given image, f.
%  The subimage is of size m-by-n, and the coordinates
%  of its top, left corner are (rx,cy).

s=zeros(m,n);
rowhigh=rx+m-1;
colhigh=cy+n-1;
xcount=0;
for r=rx:rowhigh
    xcount=xcount+1;
    ycount=0;
    for c=cy:colhigh
        ycount=ycount+1;
        s(xcount,ycount)=f(r,c);
    end
end
```

while

```
while expression
      statements
end
```

switch

```
switch newclass
    case ’uint8’
        g=im2uint8(f);
    case ’uint16’
        g=im2uint16(f);
    case ’double’
        g=im2double(f);
    otherwise
        error(’Unknown or improper image class.’)
end
```