Cuerpos y Anillos

En los cuerpos todos los elementos distintos de 0 tienen inverso

In [1]:
```
RR # Cuerpo de los números reales.
```

Out[1]:
```
Real Field with 53 bits of precision
```

In [2]:
```
QQ # Cuerpo de los números racionales (lo usaremos siempre que sea posible porque es exacto)
```

Out[2]:
```
Rational Field
```

In [3]:
```
GF(7) # Cuerpo finito de 7 elementos
```

Out[3]:
```
Finite Field of size 7
```

In [4]:
```
# Este es un ejemplo para ver los inversos de los elementos de GF(7)
# En esta asignatura no vamos a programar, aunque es posible que te mostremos algún pequeño programa de e
jemplo
for x in GF(7):
    if x!=0:
        print(x,x^-1)
```

```
1 1
2 4
3 5
4 2
5 3
6 6
```

Los anillos son similares a los cuerpos, pero no todos los elementos no nulos tienen que tener inverso

In [5]:
```
ZZ # Anillo de los números enteros
```

Out[5]:
```
Integer Ring
```

In [6]:
```
Zmod(6) # Anillo de los enteros módulo 6
```

Out[6]:
```
Ring of integers modulo 6
```

Para definir los polinomios debemos indicar donde están los coeficientes y cómo queremos llamar a las variables

In [7]:
```
P1.<x,y,z> = QQ[]
P1
```

Out[7]:
```
Multivariate Polynomial Ring in x, y, z over Rational Field
```

In [8]:
```
(x+2*y/3)^3
```

Out[8]:
```
x^3 + 2*x^2*y + 4/3*x*y^2 + 8/27*y^3
```

También podemos usar subíndices para utilizar múltiples variables e incluso letras griegas

In [9]:
```
P2.<x1,x2> = QQ[]
P2
```

Out[9]:
```
Multivariate Polynomial Ring in x1, x2 over Rational Field
```

In [10]:
```
(x1+x2)^3
```

Out[10]:
```
x1^3 + 3*x1^2*x2 + 3*x1*x2^2 + x2^3
```

In [11]:
```
show((x1+x2)^3) # De esta forma lo vemos en un formato más matemático, tal y como lo veremos en LaTeX
```

$$x_1^3 + 3x_1^2x_2 + 3x_1x_2^2 + x_2^3$$

In [12]:
```
P3.<alpha,beta1,beta2,z> = QQ[]
```

In [13]:
```
p = (alpha+2*beta1-beta2-z/3)^2
p
```

Out[13]:
```
alpha^2 + 4*alpha*beta1 + 4*beta1^2 - 2*alpha*beta2 - 4*beta1*beta2 + beta2^2 - 2/3*alpha*z - 4/3*beta1*z
+ 2/3*beta2*z + 1/9*z^2
```

In [14]:
```
show(p)
```

$$\alpha^2 + 4\alpha\beta_1 + 4\beta_1^2 - 2\alpha\beta_2 - 4\beta_1\beta_2 + \beta_2^2 - \frac{2}{3}\alpha z - \frac{4}{3}\beta_1 z + \frac{2}{3}\beta_2 z + \frac{1}{9}z^2$$

En el caso de los polinomios es posible asignar un valor a alguna de las variables (o a todas)

In [15]:
```
q = p(alpha = beta1, z = -3*beta2)
show(q)
```

$$9\beta_1^2$$

Variables y Listas

Todos los objetos de sage se pueden asignar a variables

In [16]:
```
a = 1/2+x-y
b = x+y
c = a*b
c
```

Out[16]:
```
x^2 - y^2 + 1/2*x + 1/2*y
```

También podemos poner objetos en listas utilizando los símbolos [ ] y separando los elementos entre comas.

In [17]:
```
[1,-1/2,3.55,x] # Las listas pueden tener elementos de distinto tipo
```

Out[17]:
```
[1, -1/2, 3.55000000000000, x]
```

Incluso pueden contener otras listas

In [18]:
```
[2/3,0,[1,-1,1],x,P1]
```

Out[18]:
```
[2/3,
 0,
 [1, -1, 1],
 x,
 Multivariate Polynomial Ring in x, y, z over Rational Field]
```

Las listas se pueden asignar a variables y también podemos acceder a sus elementos utilizando índices (que empiezan en 0)

In [19]:
```
L = [-1,2/3,3]
L[0]
```

Out[19]:
```
-1
```

Se pueden crear listas utilizando for

In [20]:
```
L = [(x+y)^i for i in range(4)]
show(L)
```

$$\left[1, x + y, x^2 + 2xy + y^2, x^3 + 3x^2y + 3xy^2 + y^3\right]$$

Matrices

Para definir una matriz debemos indicar donde están los coeficientes (un cuerpo o un anillo) y los elementos que la componen por filas

In [21]:
```
M = matrix(QQ,[[1,1/2,-1],[0,1,2]])
show(M)
```

$$\begin{pmatrix} 1 & \frac{1}{2} & -1 \\ 0 & 1 & 2 \end{pmatrix}$$

También podemos indicar el tamaño de la matriz y poner todos los elementos en una lista

In [22]:
```
M0 = matrix(QQ,2,3,[1,2,3,4,5,6])
show(M0)
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Si ponemos el tamaño, pero no los valores se entenderá que son 0

In [23]:
```
show(matrix(QQ,4,5))
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Es una buena idea añadir espacios en blanco al introducir las matrices para detectar errores más fácilmente. Estos espacios no tienen ningún efecto a la hora de introducir la matriz

In [24]:
```
a = pi/3
M1 = matrix(RR,[[1,     0,      0],
                [0,cos(a),-sin(a)],
                [0,sin(a), cos(a)]])
show(M1)
```

$$\begin{pmatrix} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.000000000000000 & 0.500000000000000 & -0.866025403784439 \\ 0.000000000000000 & 0.866025403784439 & 0.500000000000000 \end{pmatrix}$$

Los elementos se pueden acceder indicando la fila y la columna, teniendo en cuenta que los índices empiezan a contar desde 0

In [25]:
```
show(M1[2,1])
```

$$0.866025403784439$$

Se pueden hacer operaciones con matrices siempre que los tamaños sean los adecuados

In [26]:
```
show(M1^6)
```

$$\begin{pmatrix} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.000000000000000 & 1.00000000000000 & 2.22044604925031 \times 10^{-16} \\ 0.000000000000000 & -2.22044604925031 \times 10^{-16} & 1.00000000000000 \end{pmatrix}$$

Si esta matriz se hubiese calculado de forma exacta, debería ser la matriz identidad pero los números reales introducen pequeños errores de redondeo que evitaremos siempre que sea posible usando los números racionales.

Se puede particionar (subdividir en bloques) una matriz indicando la lista de filas y las columnas donde queremos poner la partición. Cuando sólo hay una posición podemos poner directamente el número sin necesidad de ponerlo en una lista. Algunos ejemplos de particiones serían:

In [27]:
```
M1.subdivide(1,[1,2])
show(M1)
M1.subdivide([],2)
show(M1)
M1.subdivide(1,1)
show(M1)
```

$$\left(\begin{array}{c|cc} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ \hline 0.000000000000000 & 0.500000000000000 & -0.866025403784439 \\ 0.000000000000000 & 0.866025403784439 & 0.500000000000000 \end{array}\right)$$

$$\left(\begin{array}{cc|c} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.000000000000000 & 0.500000000000000 & -0.866025403784439 \\ \hline 0.000000000000000 & 0.866025403784439 & 0.500000000000000 \end{array}\right)$$

$$\left(\begin{array}{c|cc} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ \hline 0.000000000000000 & 0.500000000000000 & -0.866025403784439 \\ 0.000000000000000 & 0.866025403784439 & 0.500000000000000 \end{array}\right)$$

En una matriz subdividida en bloques, podemos tomar cada uno de los bloques indicando su fila y columna considerada como bloque y numerando desde 0. Así por ejemplo

In [28]:
```
show(M1)
show(M1.subdivision(0,0))
show(M1.subdivision(0,1))
show(M1.subdivision(1,0))
show(M1.subdivision(1,1))
```

$$\left(\begin{array}{c|cc} 1.00000000000000 & 0.000000000000000 & 0.000000000000000 \\ \hline 0.000000000000000 & 0.500000000000000 & -0.866025403784439 \\ 0.000000000000000 & 0.866025403784439 & 0.500000000000000 \end{array}\right)$$

$$\left(1.00000000000000\right)$$

$$\left(0.000000000000000 \quad 0.000000000000000\right)$$

$$\begin{pmatrix} 0.000000000000000 \\ 0.000000000000000 \end{pmatrix}$$

$$\begin{pmatrix} 0.500000000000000 & -0.866025403784439 \\ 0.866025403784439 & 0.500000000000000 \end{pmatrix}$$

También podemos extraer submatrices sin necesidad de que estén subdivididas en bloques. Para ello podemos indicar las filas y las columnas que queremos tomar de las matrices mediante listas.

In [29]:
```
M = matrix(ZZ,[[ 0, 1, 2, 3, 4],
               [ 5, 6, 7, 8, 9],
               [10,11,12,13,14],
               [15,16,17,18,19]])
show(M)
A = M[[0,1,3],[2,4]] # filas 0,1 y 3, columnas 2 y 4
show(A)
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 4 \\ 7 & 9 \\ 17 & 19 \end{pmatrix}$$

Lo más habitual será querer tomar las filas o las columnas en un rango entre dos valores, por ejemplo las filas 2,3,4 y 5. Para indicar esto escribiremos 2:6 (el segundo índice es el límite superior que no se alcanza por tanto 2:6 indica el rango [2,3,4,5]). Cuando no se indique el número significará desde el principio (si es a la izquierda) o hasta el final (si es a la derecha). Algunos ejemplos podrían ser:

In [30]:
```
show(M[1:3,2:4]) # filas 1 y 2, columnas 2 y 3
show(M[1:,:4]) # filas desde la 1 en adelante y columnas 0,1,2,3
show(M[1:,:]) # filas desde la 1 en adelante y todas las columnas (desde el principio hasta el final)
show(M[:,:3]) # todas las filas y las columnas 0,1,2
```

$$\begin{pmatrix} 7 & 8 \\ 12 & 13 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 13 \\ 15 & 16 & 17 & 18 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 2 \\ 5 & 6 & 7 \\ 10 & 11 & 12 \\ 15 & 16 & 17 \end{pmatrix}$$

Es incluso posible utilizar esta notación para hacer asignaciones de valores en matrices.

In [31]:
```
M[1:3,2:4] = matrix(ZZ,[[-1,-2],[-3,-4]])
show(M)
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & -1 & -2 & 9 \\ 10 & 11 & -3 & -4 & 14 \\ 15 & 16 & 17 & 18 & 19 \end{pmatrix}$$

In [32]:
```
N = matrix(ZZ,6,8) # matriz de 6 filas y 8 columnas rellena de ceros
N[:4,:5] = M
show(N)
N.subdivide(4,5)
show(N)
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\ 5 & 6 & -1 & -2 & 9 & 0 & 0 & 0 \\ 10 & 11 & -3 & -4 & 14 & 0 & 0 & 0 \\ 15 & 16 & 17 & 18 & 19 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\left(\begin{array}{rrrrr|rrr} 0 & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\ 5 & 6 & -1 & -2 & 9 & 0 & 0 & 0 \\ 10 & 11 & -3 & -4 & 14 & 0 & 0 & 0 \\ 15 & 16 & 17 & 18 & 19 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Una forma de definir matrices que usaremos mucho es construirla juntando otras matrices. Eso se llama construir una matriz por bloques.

In [33]:
```
A = matrix(GF(5),[[1,3],
                  [2,4]])
B = matrix(GF(5),[[1,1,2],
                  [3,1,0]])
C = matrix(GF(5),[[1,4]])
D = matrix(GF(5),[[0,1,0]])
show(block_matrix([[A,B],[C,D]]))
show(block_matrix([[A],[C]]))
show(block_matrix([[A,B]]))
show(block_matrix([[A,0],[C,D]])) # 0 representa la matriz 0 si el tamaño puede ser deducido del contexto
show(block_matrix([[B],[1]])) # 1 representa la matriz identidad si el tamaño puede ser deducido del cont
exto
```

$$\left(\begin{array}{cc|ccc} 1 & 3 & 1 & 1 & 2 \\ 2 & 4 & 3 & 1 & 0 \\ \hline 1 & 4 & 0 & 1 & 0 \end{array}\right)$$

$$\left(\begin{array}{cc} 1 & 3 \\ 2 & 4 \\ \hline 1 & 4 \end{array}\right)$$

$$\left(\begin{array}{cc|ccc} 1 & 3 & 1 & 1 & 2 \\ 2 & 4 & 3 & 1 & 0 \end{array}\right)$$

$$\left(\begin{array}{cc|ccc} 1 & 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ \hline 1 & 4 & 0 & 1 & 0 \end{array}\right)$$

$$\left(\begin{array}{ccc} 1 & 1 & 2 \\ 3 & 1 & 0 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right)$$

También podemos usar la matriz traspuesta que se calcula poniendo la matriz seguida de .T

In [34]:
```
show(B.T)
show(block_matrix([[B.T,1]]))
show(D*B.T)
```

$$\begin{pmatrix} 1 & 3 \\ 1 & 1 \\ 2 & 0 \end{pmatrix}$$

$$\left(\begin{array}{cc|ccc} 1 & 3 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 \end{array}\right)$$

$$\begin{pmatrix} 1 & 1 \end{pmatrix}$$

Reducción de matrices

Se denominan operaciones elementales por filas a las siguientes:

1. Sumar a una fila otra multiplicada por una constante

`A.add_multiple_of_row(2,4,1/3)` : Modifica la matriz A sumando a la fila 2 la fila 4 multiplicada por 1/3.

1. Multiplicar todos los elementos de una fila por un elemento invertible (si estamos en un cuerpo cualquier elemento distinto de 0 es invertible, en un anillo no siempre es así)

`A.rescale_row(0,-2/3)` : Modifica la matriz A multiplicando la fila 0 por -2/3.

1. Intercambiar dos filas

`A.swap_rows(1,3)` : Modifica la matriz A intercambiando las filas 1 y 3.

El algoritmo de reducción de matrices también llamado reducción de Gauss el un procedimiento que habréis usado para resolver sistemas de ecuaciones pero tiene múltiples aplicaciones que iremos viendo a lo largo del curso. Se realiza mediante el comando `R = A.echelon_form()` (asignamos a R la matriz A reducida por filas, A queda inalterada).

Cuando la matriz está definida sobre un cuerpo obtendremos la matriz reducida por filas, en el caso de anillos podemos obtener una matriz diferente.

In [35]:
```
M = matrix(QQ,[[12, 1, 0],
               [15, 0, 1]])
N = matrix(ZZ,[[12, 1, 0],
               [15, 0, 1]])
show(M.echelon_form()) # Matriz reducida por filas en el cuerpo de los números racionales.
show(N.echelon_form()) # Matriz reducida considerada como matriz en el anillo de números enteros.
                       # Es distinta porque 3 y 5 no son invertibles como números enteros.
```

$$\begin{pmatrix} 1 & 0 & \frac{1}{15} \\ 0 & 1 & -\frac{4}{5} \end{pmatrix}$$

$$\begin{pmatrix} 3 & 4 & -3 \\ 0 & 5 & -4 \end{pmatrix}$$

Vamos por ejemplo a resolver el sistema de ecuaciones AX=B siendo A y B las matrices que nos dan a continuación en función de los parámetros a y b.

In [36]:
```
P.<a,b> = QQ[]
A = matrix(P,[[0,0,-1,       0,      1],
              [1,0, 1,       1,     -2],
              [2,2, 1,-2*b + 2,2*a - 3],
              [3,2, 1,-2*b + 3,2*a - 4],
              [3,1, 0,       2,  a - 3]])
B = matrix(P,[[-3],
              [5],
              [a - b + 9],
              [a + 10],
              [7]])
AB = block_matrix([[A,B]]) # Construimos la matriz ampliada
show(AB)
```

$$\left(\begin{array}{rrrrr|r} 0 & 0 & -1 & 0 & 1 & -3 \\ 1 & 0 & 1 & 1 & -2 & 5 \\ 2 & 2 & 1 & -2b+2 & 2a-3 & a-b+9 \\ 3 & 2 & 1 & -2b+3 & 2a-4 & a+10 \\ 3 & 1 & 0 & 2 & a-3 & 7 \end{array}\right)$$

In [37]:
```
R = AB.echelon_form()  # Reducimos
show(R)
```

$$\left(\begin{array}{rrrrr|r} 1 & 0 & 0 & 1 & -1 & 2 \\ 0 & 1 & 0 & -b & a & \frac{1}{2}a-\frac{1}{2}b+1 \\ 0 & 0 & 1 & 0 & -1 & 3 \\ 0 & 0 & 0 & 0 & 0 & b-1 \\ 0 & 0 & 0 & b-1 & 0 & -\frac{1}{2}a+\frac{1}{2}b \end{array}\right)$$

De la penúltima ecuación deducimos que si b no es 1 el sistema es incompatible y de la última ecuación deducimos que si a no es igual a b el sistema también es incompatible. Seguimos en el caso en que a = b = 1. Para ello sustitimos los valores.

In [38]:
```
C = R(a=1,b=1)
C.subdivide([],5)
show(C)
```

$$\left(\begin{array}{rrrrr|r} 1 & 0 & 0 & 1 & -1 & 2 \\ 0 & 1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

In [39]:
```
Q.<alpha1,alpha2> = QQ[] # Añadimos dos nuevos parámetros
N = C.change_ring(Q) # matrix(Q,5,6)
#N[:,:] = C
N[3,3] = 1 # Añadimos los pivotes que faltan
N[4,4] = 1
N[3,5] = alpha1 # En la parte derecha ponemos los parámetros que hemos tomado
N[4,5] = alpha2
N.subdivide(3,5)
show(N)
show(N.echelon_form())
```

$$\left(\begin{array}{rrrrr|r} 1 & 0 & 0 & 1 & -1 & 2 \\ 0 & 1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 3 \\ \hline 0 & 0 & 0 & 1 & 0 & \alpha_1 \\ 0 & 0 & 0 & 0 & 1 & \alpha_2 \end{array}\right)$$

$$\left(\begin{array}{rrrrr|r} 1 & 0 & 0 & 0 & 0 & -\alpha_1 + \alpha_2 + 2 \\ 0 & 1 & 0 & 0 & 0 & \alpha_1 - \alpha_2 + 1 \\ 0 & 0 & 1 & 0 & 0 & \alpha_2 + 3 \\ \hline 0 & 0 & 0 & 1 & 0 & \alpha_1 \\ 0 & 0 & 0 & 0 & 1 & \alpha_2 \end{array}\right)$$

La columna derecha nos da las soluciones del sistema en función de los parámetros.