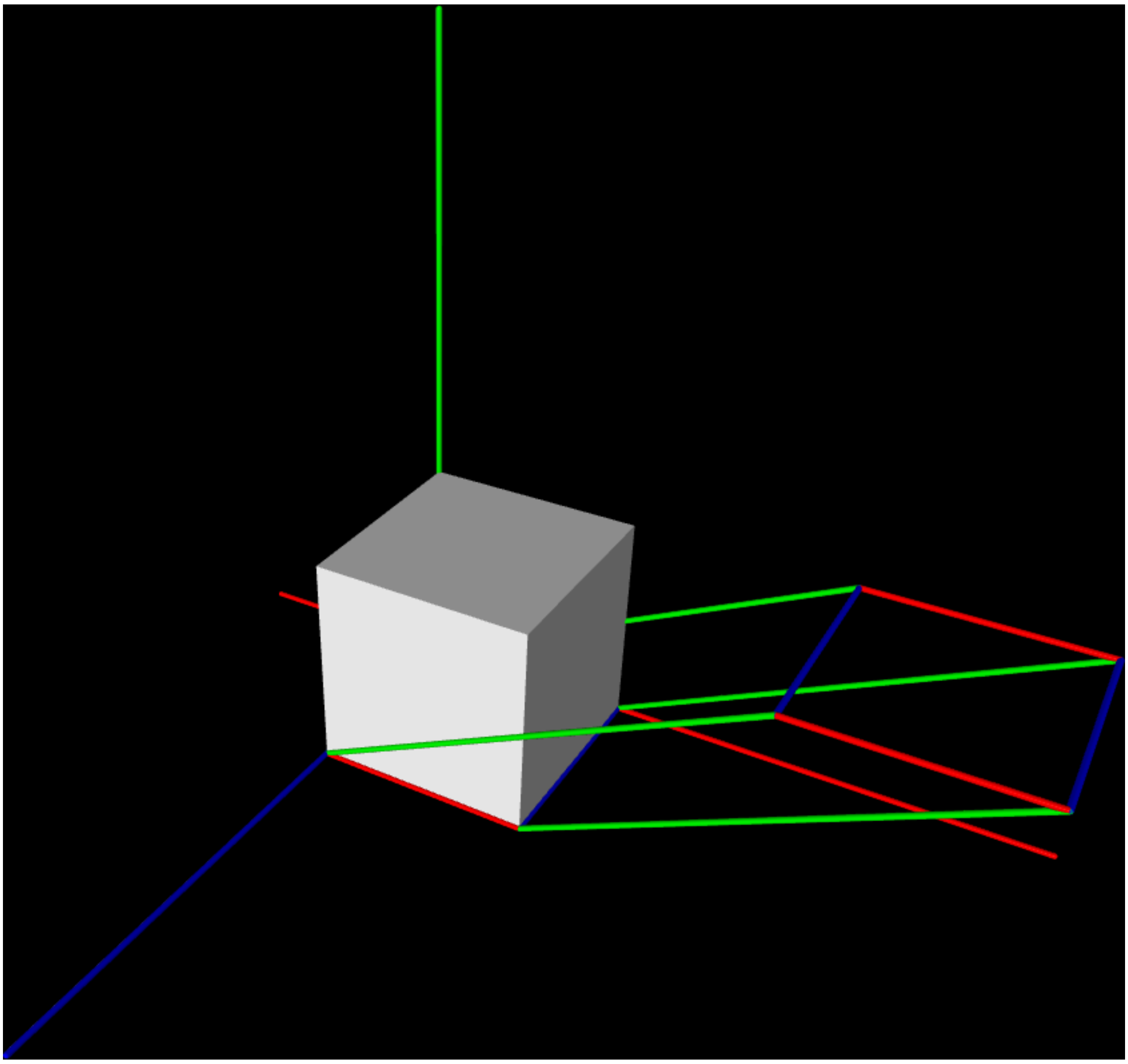
Chapter 5. Elementary Operation and Matrix Invariants

5.1. Elementary matrices and operations

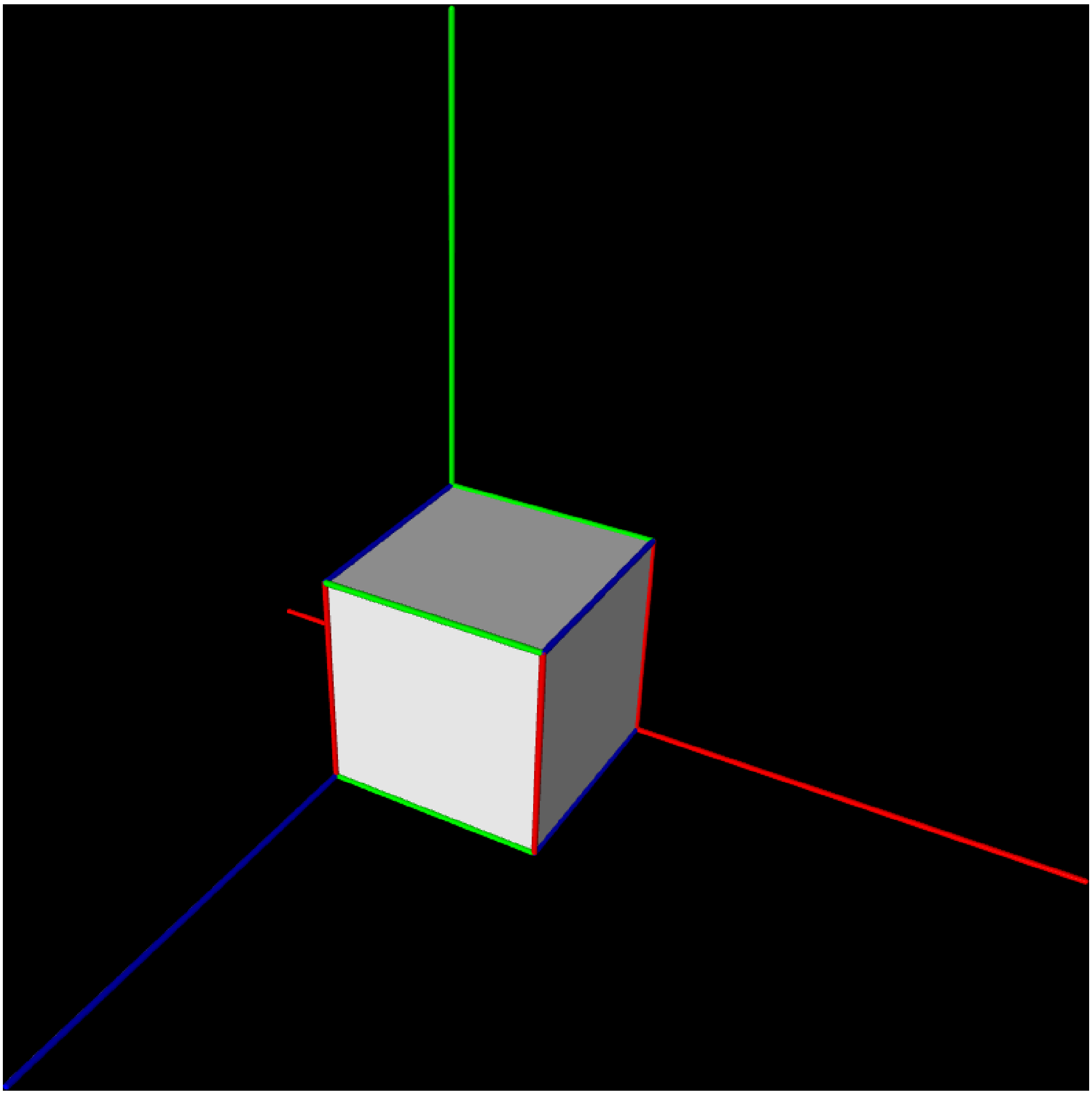
Program:elementary_vp.py

In [1]:
```
from vpython import *
import numpy as np

o = vec(0, 0, 0)
x, y, z = vec(1, 0, 0), vec(0, 1, 0), vec(0, 0, 1)
yz, zx, xy = [o, y, z, y+z], [o, z, x, z+x], [o, x, y, x+y]

def T(A, u): return vec(*np.dot(A, (u.x, u.y, u.z)))

E1 = [[1, 2, 0], [0, 1, 0], [0, 0, 1]]
E2 = [[0, 1, 0], [1, 0, 0], [0, 0, 1]]
E3 = [[2, 0, 0], [0, 1, 0], [0, 0, 1]]

def draw(E):
    scene = canvas(width=600, height=600)
    scene.camera.pos = vec(3, 4, 5)
    scene.camera.axis = -vec(3, 4, 5)
    box(pos=(x+y+z)/2)
    for axis in [x, y, z]:
        curve(pos=[-axis, 3*axis], color=axis)
    for axis, face in [(x, yz), (y, zx), (z, xy)]:
        for side in face:
            A = E
            curve(pos=[T(A, side), T(A, axis+side)], color=axis)
```

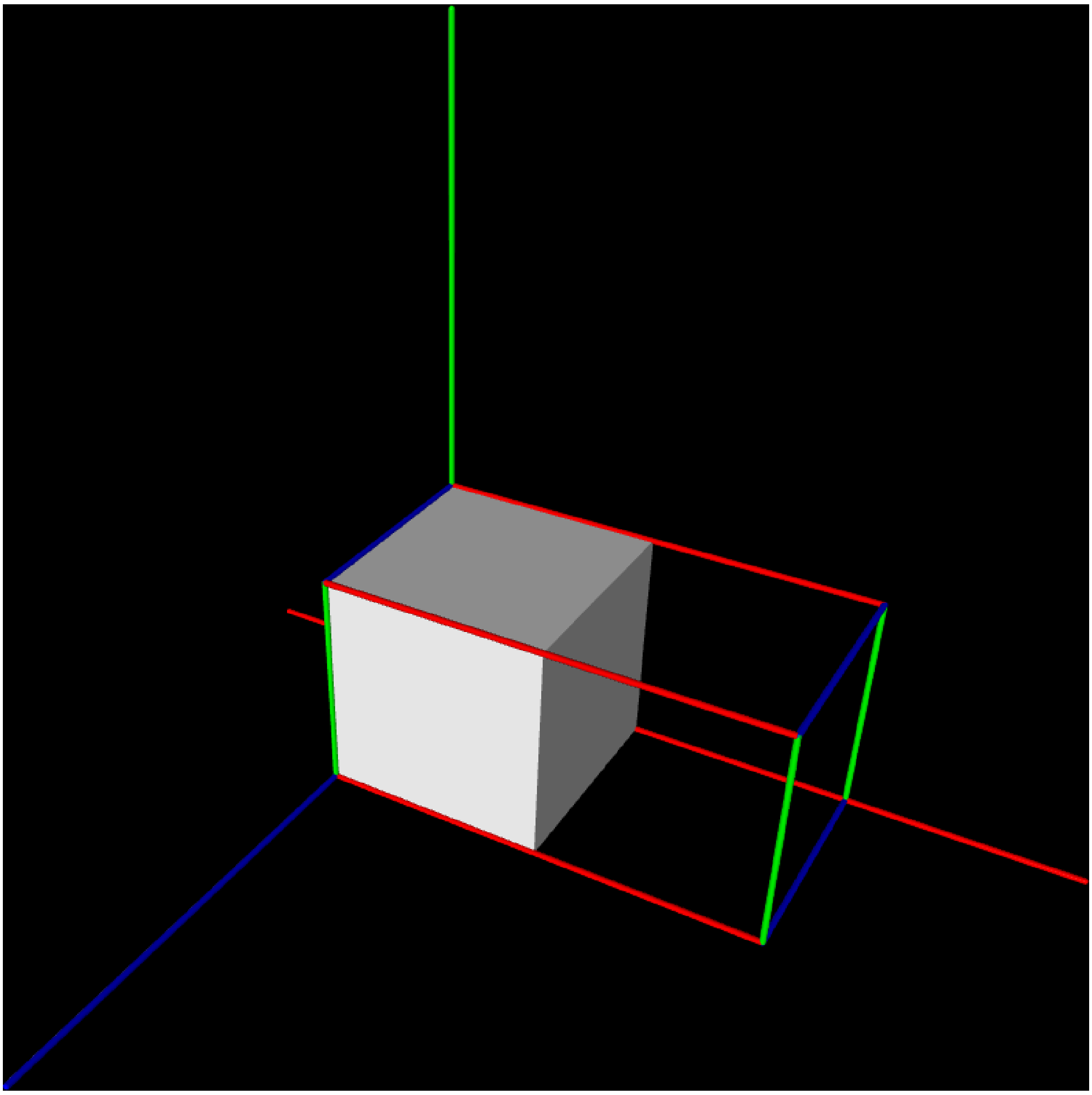
In [2]:
```
draw(E1)
```



In [3]:
```
draw(E2)
```



In [4]:
```
draw(E3)
```



---

elementary_sp.ipynb

In [1]:
```
from sympy import Matrix, var

var('x y a11 a12 a13 a21 a22 a23 a31 a32 a33')
E1 = Matrix([[1, x, 0], [0, 1, 0], [0, 0, 1]])
E2 = Matrix([[0, 1, 0], [1, 0, 0], [0, 0, 1]])
E3 = Matrix([[1, 0, 0], [0, y, 0], [0, 0, 1]])
A = Matrix([[a11, a12, a13], [a21, a22, a23], [a31, a32, a33]])
```

In [2]:
```
E1 * A
```

Out[2]:

$$\begin{bmatrix} a_{11} + a_{21}x & a_{12} + a_{22}x & a_{13} + a_{23}x \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [3]:
```
E2 * A
```

Out[3]:

$$\begin{bmatrix} a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [4]:
```
E3 * A
```

Out[4]:
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21}y & a_{22}y & a_{23}y \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [5]:
```
A * E1
```

Out[5]:
$$\begin{bmatrix} a_{11} & a_{11}x + a_{12} & a_{13} \\ a_{21} & a_{21}x + a_{22} & a_{23} \\ a_{31} & a_{31}x + a_{32} & a_{33} \end{bmatrix}$$

In [6]:
```
A * E2
```

Out[6]:
$$\begin{bmatrix} a_{12} & a_{11} & a_{13} \\ a_{22} & a_{21} & a_{23} \\ a_{32} & a_{31} & a_{33} \end{bmatrix}$$

In [7]:
```
A * E3
```

Out[7]:
$$\begin{bmatrix} a_{11} & a_{12}y & a_{13} \\ a_{21} & a_{22}y & a_{23} \\ a_{31} & a_{32}y & a_{33} \end{bmatrix}$$

In [8]:
```
B = A.copy(); B[1,:] *= x; B
```

Out[8]:
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21}x & a_{22}x & a_{23}x \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [9]:
```
B = A.copy(); B[:,2] *= x; B
```

Out[9]:
$$\begin{bmatrix} a_{11} & a_{12} & a_{13}x \\ a_{21} & a_{22} & a_{23}x \\ a_{31} & a_{32} & a_{33}x \end{bmatrix}$$

In [10]:
```
B = A.copy(); B[0,:], B[1,:] = B[1,:], B[0,:]; B
```

Out[10]:
$$\begin{bmatrix} a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [11]:
```
B = A.copy(); B[:,1], B[:,2] = B[:,2], B[:,1]; B
```

Out[11]:
$$\begin{bmatrix} a_{11} & a_{13} & a_{12} \\ a_{21} & a_{23} & a_{22} \\ a_{31} & a_{33} & a_{32} \end{bmatrix}$$

In [12]:
```
B = A.copy(); B[0,:] += y * B[1,:]; B
```

Out[12]:

$$\begin{bmatrix} a_{11} + a_{21}y & a_{12} + a_{22}y & a_{13} + a_{23}y \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In [13]:
```
B = A.copy(); B[:,1] += y * B[:,2]; B
```

Out[13]:

$$\begin{bmatrix} a_{11} & a_{12} + a_{13}y & a_{13} \\ a_{21} & a_{22} + a_{23}y & a_{23} \\ a_{31} & a_{32} + a_{33}y & a_{33} \end{bmatrix}$$

Untitled.ipynb

In [1]:
```
from numpy import array
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
B = A.copy(); B[[0, 1], :] = B[[1, 0], :]; B
```

Out[1]:
```
array([[4, 5, 6],
       [1, 2, 3],
       [7, 8, 9]])
```

In [2]:
```
B = A.copy(); B[:, [1, 2]] = B[:, [2, 1]]; B
```

Out[2]:
```
array([[1, 3, 2],
       [4, 6, 5],
       [7, 9, 8]])
```

5.2. Rank

Untitled.ipynb

In [1]:
```
from numpy import *
A = array([[1, 2, 3], [2, 3, 4], [3, 4, 5]])
linalg.matrix_rank(A)
```

Out[1]: 2

---

prob_rank.ipynb

In [1]:
```
from numpy.random import seed, choice, permutation
from sympy import Matrix

def f(P, m1, m2, n):
    if n > min(m1, m2):
        return Matrix(choice(P, (m1, m2)))
    else:
        while True:
            X, Y = choice(P, (m1, n)), choice(P, (n, m2))
            A = Matrix(X.dot(Y))
            if A.rank() == n:
                return A

m1, m2 = 3, 4
seed(2021)
```

```
for i in permutation(max(m1, m2)):
    print(f([-3, -2, -1, 1, 2, 3], m1, m2, i+1))
```

```
Matrix([[-3, 3, 2, 1], [3, 3, 3, -3], [2, -2, 3, -2]])
Matrix([[5, -2, 1, -14], [-5, 2, -9, 4], [-4, 1, -3, 11]])
Matrix([[3, -1, -2, 2], [-7, 4, 7, -3], [1, 3, 4, 4]])
Matrix([[-2, -1, 3, 1], [4, 2, -6, -2], [4, 2, -6, -2]])
```

5.3. Determinant

Program: determinant.ipynb

In [1]:
```
from functools import reduce

def P(n):
    if n == 1:
        return [([0], 1)]
    else:
        Q = []
        for p, s in P(n-1):
            Q.append((p + [n-1], s))
            for i in range(n-1):
                q = p + [n-1]
                q[i], q[-1] = q[-1], q[i]
                Q.append((q, -1*s))
        return Q

def prod(L): return reduce(lambda x, y: x*y, L)

def det(A):
    n = len(A)
    a = sum([s * prod([A[i][p[i]] for i in range(n)])
            for p, s in P(n)])
    return a

if __name__ == '__main__':
    A = [[1, 2], [2, 3]]
    B = [[1, 2], [2, 4]]
    C = [[1, 2, 3], [2, 3, 4], [3, 4, 5]]
    D = [[1, 2, 3], [2, 3, 1], [3, 1, 2]]
    print(det(A), det(B), det(C), det(D))
```

```
-1 0 0 -18
```

Untitled.ipynb

In [1]:
```
from numpy.linalg import det
A = [[1, 2], [2, 3]]
B = [[1, 2], [2, 4]]
C = [[1, 2, 3], [2, 3, 4], [3, 4, 5]]
D = [[1, 2, 3], [2, 3, 1], [3, 1, 2]]
det(A), det(B), det(C), det(D)
```

Out[1]:
```
(-1.0, 0.0, -7.401486830834414e-17, -18.000000000000004)
```

Program: error.ipynb

In [1]:
```
from numpy.linalg import det, matrix_rank
from numpy.random import seed, normal

seed(123)
n = 20
F =normal(0, 1, (n, n-1))
G = F.dot(F.T)
print(f'shape = {G.shape}')
print(f'det = {det(G)}')
print(f'rank = {matrix_rank(G)}')
```

```
shape = (20, 20)
det = 23.147833157995517
rank = 19
```

Untitled1.ipynb

In [1]:
```
from sympy import Matrix, symbols
A = Matrix([[1, 2], [2, 3]])
B = Matrix([[1, 2], [2, 4]])
C = Matrix([[1, 2, 3], [2, 3, 4], [3, 4, 5]])
D = Matrix([[1, 2, 3], [2, 3, 1], [3, 1, 2]])
A.det(), B.det(), C.det(), D.det()
```

Out[1]: (-1, 0, 0, -18)

In [2]:
```
a11, a12, a13 = symbols('a11, a12, a13')
a21, a22, a23 = symbols('a21, a22, a23')
a31, a32, a33 = symbols('a31, a32, a33')
E = Matrix([[a11,a12], [a21,a22]])
F = Matrix([[a11,a12,a13], [a21,a22,a23], [a31,a32,a33]])
E.det()
```

Out[2]: $a_{11}a_{22} - a_{12}a_{21}$

In [3]:
```
F.det()
```

Out[3]: $a_{11}a_{22}a_{33} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31}$

Program: prob_det.ipynb

In [1]:
```
from numpy.random import seed, choice, permutation
from sympy import Matrix

def f(P, m, p):
    while True:
        A = Matrix(choice(P, (m, m)))
        if p == 0:
            if A.det() == 0:
                return A
        elif A.det() != 0:
            return A

m = 3
seed(2021)
```

```
for p in permutation(2):
    print(f([-3, -2, -1, 1, 2, 3], m, p))
```

```
Matrix([[3, -2, -3], [3, 2, 1], [3, 3, 3]])
Matrix([[1, -2, 1], [-2, -2, 1], [1, -2, 1]])
```

5.4. Trace

Empty

5.5. Systems of linear equations

Untitled.ipynb

In [1]:
```
from numpy.linalg import solve
solve([[1,2,3],[2,3,1],[3,1,2]],[6,9,12])
```

Out[1]:
```
array([3.5, 0.5, 0.5])
```

Program: prob_eqn.ipynb

In [1]:
```
from numpy.random import seed, choice, shuffle
from sympy import Matrix, latex, solve, zeros
from sympy.abc import x, y, z

def f(P, m, n):
    while True:
        A = Matrix(choice(P, (3, 4)))
        if A[:, :3].rank() == m and A.rank() == n:
            break
    A, b = A[:, :3], A[:, 3]
    u = Matrix([[x], [y], [z]])
    print(f'{latex(A)}{latex(u)}={latex(b)}')
    print(solve(A*u - b, [x, y, z]))

seed(1234)
m, n = 2, 2
f(range(2, 10), m, n)
```

```
\left[\begin{matrix}3 & 2 & 3\\8 & 6 & 2\\7 & 5 & 4\end{matrix}\right]\left[\begin{matrix}x\\y\\z\end{matrix}\right]=\left[\begin{matrix}7\\2\\8\end{matrix}\right]
{x: 19 - 7*z, y: 9*z - 25}
```

$$\left[\begin{matrix}3 & 2 & 3\\8 & 6 & 2\\7 & 5 & 4\end{matrix}\right]\left[\begin{matrix}x\\y\\z\end{matrix}\right]=\left[\begin{matrix}7\\2\\8\end{matrix}\right]$$

5.6. The inverse matrix

Untitled.ipynb

In [1]:
```
from sympy import Matrix
```

```
A = Matrix([[1, 2, 3, 1, 0, 0],
            [2, 3, 1, 0, 1, 0],
            [3, 1, 2, 0, 0, 1]])
```

In [2]:
```
A[1, :] -= A[0, :] * 2; A[2, :] -= A[0, :] * 3; A
```

Out[2]:
$$\begin{bmatrix}1 & 2 & 3 & 1 & 0 & 0\\ 0 & -1 & -5 & -2 & 1 & 0\\ 0 & -5 & -7 & -3 & 0 & 1\end{bmatrix}$$

In [3]:
```
A[1, :] /= -1; A
```

Out[3]:
$$\begin{bmatrix}1 & 2 & 3 & 1 & 0 & 0\\ 0 & 1 & 5 & 2 & -1 & 0\\ 0 & -5 & -7 & -3 & 0 & 1\end{bmatrix}$$

In [4]:
```
A[0, :] -= A[1, :] * 2; A[2, :] += A[1, :] * 5; A
```

Out[4]:
$$\begin{bmatrix}1 & 0 & -7 & -3 & 2 & 0\\ 0 & 1 & 5 & 2 & -1 & 0\\ 0 & 0 & 18 & 7 & -5 & 1\end{bmatrix}$$

In [5]:
```
A[2, :] /= 18; A
```

Out[5]:
$$\begin{bmatrix}1 & 0 & -7 & -3 & 2 & 0\\ 0 & 1 & 5 & 2 & -1 & 0\\ 0 & 0 & 1 & \frac{7}{18} & -\frac{5}{18} & \frac{1}{18}\end{bmatrix}$$

In [6]:
```
A[0, :] += A[2, :] * 7; A[1, :] -= A[2, :] * 5; A
```

Out[6]:
$$\begin{bmatrix}1 & 0 & 0 & -\frac{5}{18} & \frac{1}{18} & \frac{7}{18}\\ 0 & 1 & 0 & \frac{1}{18} & \frac{7}{18} & -\frac{5}{18}\\ 0 & 0 & 1 & \frac{7}{18} & -\frac{5}{18} & \frac{1}{18}\end{bmatrix}$$

Program: inv.ipynb

In [1]:
```
from numpy import array, linalg, random

n = 5
A = random.randint(0, 10, (n, n))
K = [[j for j in range(n) if j != i] for i in range(n)]
B = array([[(-1) ** (i+j) * linalg.det(A[K[i], :][:, K[j]])
            for i in range(n)] for j in range(n)])

print(A.dot(B/linalg.det(A)))
```

```
[[ 1.00000000e+00  4.44089210e-16  7.49400542e-16 -1.83880688e-16
   4.85722573e-17]
 [-3.88578059e-16  1.00000000e+00 -5.55111512e-16 -4.30211422e-16
  -1.11022302e-16]
 [-2.56739074e-16  6.66133815e-16  1.00000000e+00 -2.08166817e-16
  -2.77555756e-17]
 [-4.99600361e-16  1.11022302e-15  7.49400542e-16  1.00000000e+00
  -1.73472348e-16]
 [-4.44089210e-16  7.77156117e-16  4.44089210e-16 -6.24500451e-16
   1.00000000e+00]]
```

In [ ]: