练习1：图像向量空间与“零均值图像”子空间

场景

把一张 8×8 的灰度小图拉平成向量 $\mathbf{x} \in \mathbb{R}^{64}$。
令整空间 $\mathcal{V} = \mathbb{R}^{64}$。定义集合

$$\mathcal{U} = \{\mathbf{x} \in \mathbb{R}^{64} \mid \mathbf{1}^\top \mathbf{x} = 0\},$$

即所有 **像素和为 0（零均值）** 的图像向量。

任务

1. **证明**：$\mathcal{U}$ 是 $\mathcal{V}$ 的一个子空间。
2. 用 **Python** 随机生成几张“图像”向量，数值上**验证**子空间的封闭性（加法、数乘）。
3. 随机取三张零均值图像 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{U}$，判断它们是否**线性无关。**
4. 给定任意图像向量 $\mathbf{y} \in \mathbb{R}^{64}$，写出其在 $\mathcal{U}$ 上的**正交投影**（也就是去均值），并用代码实现。

提示 1：证明子空间

- 子空间必须满足三个条件：
  1. **非空**（至少包含零向量）；
  2. **加法封闭**；
  3. **数乘封闭。**
- 可以直接代入 $\mathbf{1}^\top \mathbf{x} = 0$ 来检验。

提示 2：Python 验证封闭性

- 用 `numpy` 生成随机向量，再减去均值得到“零均值向量”。
- 用 `.sum()` 检查是不是接近 0（浮点误差允许 `1e-15` 量级）。
- 分别验证：
  - 两个零均值向量相加，结果是否还是零均值；
  - 一个零均值向量乘以常数，结果是否还是零均值。

提示 3：线性无关性

- 将多个零均值向量拼成矩阵 `A = [x1, x2, x3]`。
- 用 `np.linalg.matrix_rank(A)` 计算秩。
- 判断标准：

- $\text{rank}(A) = 3 \Rightarrow$ 三个向量线性无关；
- $\text{rank}(A) < 3 \Rightarrow$ 三个向量线性相关。

提示 4：正交投影（去均值）

- 思考：$\mathcal{U}$ 是所有“零均值向量”的集合。
- 对任意 $\mathbf{y}$，要找它在 $\mathcal{U}$ 中的投影，就是“减去均值部分”。
- 公式：

$$\text{Proj}_{\mathcal{U}}(\mathbf{y}) = \mathbf{y} - \frac{\sum_i y_i}{64} \cdot \mathbf{1}$$

- 编程思路：`projU_y = y - y.mean()`。
- 应用场景：这是**去 DC 分量**，在信号处理、图像分析里很常见，比如人脸识别前的图像标准化。

任务1:

1. 验证 $\mathcal{U}$ 非空=

取 $\mathcal{V}$ 中的**零向量** $\mathbf{0} = [0, 0, \ldots, 0]^\top$（共64个0，对应全黑的8×8图像），计算其内积：

$$\mathbf{1}^\top \mathbf{0} = 1 \times 0 + 1 \times 0 + \cdots + 1 \times 0 = 0$$

零向量 $\mathbf{0}$ 满足 $\mathcal{U}$ 的定义（$\mathbf{1}^\top \mathbf{0} = 0$），因此 $\mathbf{0} \in \mathcal{U}$，即 $\mathcal{U}$ 非空。

2. 验证 $\mathcal{U}$ 对向量加法封闭=

任取 $\mathbf{u}, \mathbf{v} \in \mathcal{U}$，根据 $\mathcal{U}$ 的定义，有：

$$\mathbf{1}^\top \mathbf{u} = 0, \quad \mathbf{1}^\top \mathbf{v} = 0$$

需证明 $\mathbf{u} + \mathbf{v} \in \mathcal{U}$，即证明 $\mathbf{1}^\top (\mathbf{u} + \mathbf{v}) = 0$。

根据**内积的线性性质（加法分配律）**：

$$\mathbf{1}^\top (\mathbf{u} + \mathbf{v}) = \mathbf{1}^\top \mathbf{u} + \mathbf{1}^\top \mathbf{v}$$

将 $\mathbf{1}^\top \mathbf{u} = 0$ 和 $\mathbf{1}^\top \mathbf{v} = 0$ 代入，得：

$$\mathbf{1}^\top (\mathbf{u} + \mathbf{v}) = 0 + 0 = 0$$

因此 $\mathbf{u} + \mathbf{v}$ 满足 $\mathcal{U}$ 的定义，即 $\mathbf{u} + \mathbf{v} \in \mathcal{U}$，加法封闭性成立。

3. 验证 $\mathcal{U}$ 对数量乘法封闭=

任取 $\mathbf{u} \in \mathcal{U}$，任取实数 k（数量因子），根据 $\mathcal{U}$ 的定义，有：

$$\mathbf{1}^\top \mathbf{u} = 0$$

需证明 $k\mathbf{u} \in \mathcal{U}$，即证明 $\mathbf{1}^\top(k\mathbf{u}) = 0$。

根据**内积的线性性质（数乘结合律）**：

$$\mathbf{1}^\top(k\mathbf{u}) = k \cdot (\mathbf{1}^\top\mathbf{u})$$

将 $\mathbf{1}^\top\mathbf{u} = 0$ 代入，得：

$$\mathbf{1}^\top(k\mathbf{u}) = k \times 0 = 0$$

因此 $k\mathbf{u}$ 满足 $\mathcal{U}$ 的定义，即 $k\mathbf{u} \in \mathcal{U}$，数乘封闭性成立。

$\mathcal{U}$ 满足向量子空间的所有判定条件（非空、加法封闭、数乘封闭），故 $\mathcal{U}$ 是 $\mathcal{V} = \mathbb{R}^{64}$ 的一个子空间（即“零均值图像”子空间）。

任务2:

In [19]:
```python
import numpy as np
def generate_zero_mean_vector():
    img = np.random.rand(8, 8)
    vec = img.flatten()
    mean = vec.mean()
    zero_mean_vec = vec - mean

    return zero_mean_vec

def is_zero_mean(vec, tol=1e-15):
    return abs(vec.sum()) < tol


vec1 = generate_zero_mean_vector()
vec2 = generate_zero_mean_vector()


print(f"向量1是否为零均值: {is_zero_mean(vec1)} (和为: {vec1.sum():.2e})")
print(f"向量2是否为零均值: {is_zero_mean(vec2)} (和为: {vec2.sum():.2e})")
print()

sum_vec = vec1 + vec2
print(f"两向量之和是否为零均值: {is_zero_mean(sum_vec)} (和为: {sum_vec.sum():.2e

scalar = np.random.rand()
scaled_vec = scalar * vec1
print(f"向量1乘以常数{scalar:.4f}后是否为零均值: {is_zero_mean(scaled_vec)} (和为
```

```
向量1是否为零均值: True (和为: -1.11e-16)
向量2是否为零均值: True (和为: -4.44e-16)

两向量之和是否为零均值: True (和为: -7.77e-16)
向量1乘以常数0.2702后是否为零均值: True (和为: 8.33e-17)
```

任务3:

In [120…
```python
def generate_zero_mean_vector():
    img = np.random.rand(8, 8)
```

```
    vec = img.flatten()
    mean = vec.mean()
    zero_mean_vec = vec - mean
    return zero_mean_vec

def is_zero_mean(vec, tol=1e-15):
    return abs(vec.sum()) < tol

x1 = generate_zero_mean_vector()
x2 = generate_zero_mean_vector()
x3 = generate_zero_mean_vector()

print(f"x1是否为零均值: {is_zero_mean(x1)} (和为: {x1.sum():.2e})")
print(f"x2是否为零均值: {is_zero_mean(x2)} (和为: {x2.sum():.2e})")
print(f"x3是否为零均值: {is_zero_mean(x3)} (和为: {x3.sum():.2e})")
print()
A = np.column_stack((x1, x2, x3))
print(f"矩阵A的形状: {A.shape}")


rank = np.linalg.matrix_rank(A)
print(f"矩阵A的秩: {rank}")

if rank == 3:
    print("结论: 三个零均值向量线性无关")
else:
    print(f"结论: 三个零均值向量线性相关（秩为{rank} < 3）")
```

```
x1是否为零均值: True (和为: 0.00e+00)
x2是否为零均值: True (和为: 6.66e-16)
x3是否为零均值: True (和为: 7.77e-16)

矩阵A的形状: (64, 3)
矩阵A的秩: 3
结论: 三个零均值向量线性无关
```

任务4:

In [ ]:

```
def project_to_u(y):
    mean = y.mean()
    proj = y - mean
    return proj
```

练习2：多项式空间与子空间

场景说明

次数不超过 2 的所有实系数多项式构成向量空间：

$$\mathcal{V} = \{\, a_0 + a_1 x + a_2 x^2 \mid a_i \in \mathbb{R} \,\}.$$

考虑子集：

$$\mathcal{U} = \{\, p(x) \in \mathcal{V} \mid p(1) = 0 \,\}.$$

任务：

1. 证明 $\mathcal{V}$ 是向量空间。
2. 证明（或证否）$\mathcal{U}$ 是 $\mathcal{V}$ 的子空间。
3. 在 Python 中构造若干多项式，验证它们是否属于 $\mathcal{U}$（即 $p(1)=0$）。
4. 将这些多项式写成系数向量，组成矩阵，计算秩以判断线性独立性。

任务1:

要证明 $\mathcal{V}$ 是向量空间，我们需要验证它满足向量空间定义的 8 条公理。
已知 $\mathcal{V}$ 是次数不超过 2 的实系数多项式集合，即
[$\mathcal{V}$ = { a_0 + a_1 x + a_2 x^2 \mid a_0, a_1, a_2 \in $\mathbb{R}$ }.]

设 $p(x)=a_0+a_1x+a_2x^2$，$q(x)=b_0+b_1x+b_2x^2$，$r(x)=c_0+c_1x+c_2x^2$ 是 $\mathcal{V}$ 中任意多项式，$k,m\in\mathbb{R}$。

---

1. 加法交换律
[p(x) + q(x) = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 = q(x) + p(x).]

2. 加法结合律
[(p(x) + q(x)) + r(x) = p(x) + (q(x) + r(x)).] 因为实数的加法满足结合律，所以对应系数相加也满足结合律。

3. 零向量存在
零多项式 $0=0+0x+0x^2$ 属于 $\mathcal{V}$，且
[p(x) + 0 = p(x).]

4. 负向量存在
对任意 $p(x)$，定义 $-p(x)=-a_0-a_1x-a_2x^2\in\mathcal{V}$，满足
[p(x) + (-p(x)) = 0.]

5. 数乘与单位元
[1 \cdot p(x) = p(x).]

6. 数乘结合律
[k(m p(x)) = (km) p(x).] 因为实系数多项式的数乘满足该性质。

7. 数乘对向量加法的分配律
[k(p(x) + q(x)) = k p(x) + k q(x).]

8. 数乘对实数加法的分配律
[(k + m) p(x) = k p(x) + m p(x).]

由于 $\mathcal{V}$ 中多项式加法与数乘的结果仍是次数不超过 2 的多项式（封闭性成立），并且上述 8 条公理均满足，因此 $\mathcal{V}$ 是实数域 $\mathbb{R}$ 上的向量空间。

任务2:

要判断 $\mathcal{U}$ 是否是 $\mathcal{V}$ 的子空间，我们需要验证 $\mathcal{U}$ 是否满足子空间的三个条件：

1. 零向量检验
零多项式 $0 = 0 + 0x + 0x^2$ 满足 $0(1) = 0$，因此 $0 \in \mathcal{U}$。

2. 加法封闭性检验
设 $p(x), q(x) \in \mathcal{U}$，即 $p(1) = 0$，$q(1) = 0$。
考虑 $r(x) = p(x) + q(x)$，则
[r(1) = p(1) + q(1) = 0 + 0 = 0,]
所以 $r(x) \in \mathcal{U}$，加法封闭。

3. 数乘封闭性检验
设 $p(x) \in \mathcal{U}$，$k \in \mathbb{R}$，即 $p(1) = 0$。
考虑 $s(x) = k \cdot p(x)$，则
[s(1)= $k \cdot p(1)$ = k· 0 = 0,]
所以 $s(x) \in \mathcal{U}$，数乘封闭。

由于 $\mathcal{U}$ 满足子空间的三个条件，因此 $\mathcal{U}$ 是 $\mathcal{V}$ 的子空间。

任务3:

In [1]:
```python
# === Python: 构造与检验 ===
import numpy as np

# 定义几个多项式（np.poly1d 的系数从高次到低次）
p1 = np.poly1d([1, -1])      # p1(x) = x - 1
p2 = np.poly1d([1, 0, -1])   # p2(x) = x^2 - 1
p3 = np.poly1d([2, -2, 0])   # p3(x) = 2x^2 - 2x

# 验证多项式是否属于𝒰（p(1)=0）
def is_in_U(p):
    return p(1) == 0

print("验证结果:")
print(f"p1(x) = {p1} 属于𝒰? {is_in_U(p1)}")
print(f"p2(x) = {p2} 属于𝒰? {is_in_U(p2)}")
print(f"p3(x) = {p3} 属于𝒰? {is_in_U(p3)}")

# 将多项式写成"系数向量"（统一到2次：a2, a1, a0）
def get_coeff_vector(p):
    coeffs = p.coeffs
    # 补全到3个系数（2次多项式）
    while len(coeffs) < 3:
        coeffs = np.append(coeffs, 0)
    return coeffs

coeff_p1 = get_coeff_vector(p1)  # [0, 1, -1]
coeff_p2 = get_coeff_vector(p2)  # [1, 0, -1]
coeff_p3 = get_coeff_vector(p3)  # [2, -2, 0]

print("\n系数向量:")
print(f"p1的系数向量: {coeff_p1}")
```

```
print(f"p2的系数向量: {coeff_p2}")
print(f"p3的系数向量: {coeff_p3}")

# 构造系数矩阵并计算秩
matrix = np.array([coeff_p1, coeff_p2, coeff_p3])
rank = np.linalg.matrix_rank(matrix)

print(f"\n系数矩阵的秩: {rank}")
```

```
验证结果:
p1(x) =
1 x - 1 属于𝒰? True
p2(x) =    2
1 x - 1 属于𝒰? True
p3(x) =    2
2 x - 2 x 属于𝒰? True

系数向量:
p1的系数向量: [ 1 -1  0]
p2的系数向量: [ 1  0 -1]
p3的系数向量: [ 2 -2  0]

系数矩阵的秩: 2
```

任务4:

In [2]:

```
import numpy as np

# 定义多项式（np.poly1d的系数从高次到低次）
p1 = np.poly1d([1, -1])      # p1(x) = x - 1
p2 = np.poly1d([1, 0, -1])   # p2(x) = x^2 - 1
p3 = np.poly1d([2, -2, 0])   # p3(x) = 2x^2 - 2x

# 将多项式转换为系数向量（统一到2次：x², x, 常数项）
def get_coeff_vector(p):
    coeffs = p.coeffs
    # 补全到3个系数（2次多项式）
    while len(coeffs) < 3:
        coeffs = np.insert(coeffs, 0, 0)  # 在开头补0
    return coeffs

# 获取系数向量
coeff_p1 = get_coeff_vector(p1)  # [0, 1, -1]
coeff_p2 = get_coeff_vector(p2)  # [1, 0, -1]
coeff_p3 = get_coeff_vector(p3)  # [2, -2, 0]

# 构造系数矩阵（每行是一个多项式的系数向量）
matrix = np.array([coeff_p1, coeff_p2, coeff_p3])

# 计算矩阵的秩
rank = np.linalg.matrix_rank(matrix)

# 输出结果
print("多项式系数向量:")
print(f"p1(x) = {p1}: {coeff_p1}")
print(f"p2(x) = {p2}: {coeff_p2}")
print(f"p3(x) = {p3}: {coeff_p3}")

print("\n系数矩阵:")
```

```
print(matrix)

print(f"\n矩阵的秩: {rank}")

# 判断线性独立性
if rank == 3:
    print("这三个多项式线性无关")
elif rank == 2:
    print("这三个多项式线性相关（秩为2），其中有两个是线性无关的")
else:
    print("这三个多项式线性相关（秩为1），其中有一个是线性无关的")
```

```
多项式系数向量:
p1(x) =
1 x - 1: [ 0  1 -1]
p2(x) =    2
1 x - 1: [ 1  0 -1]
p3(x) =    2
2 x - 2 x: [ 2 -2  0]

系数矩阵:
[[ 0  1 -1]
 [ 1  0 -1]
 [ 2 -2  0]]

矩阵的秩: 2
这三个多项式线性相关（秩为2），其中有两个是线性无关的
```

练习3：偶函数与奇函数子空间（选做）

场景说明

在函数空间 $\mathbb{R}[x]$（所有实系数多项式的集合）中，我们定义：

- **偶函数子空间**：

$$\mathcal{U}_{\text{even}} = \{p(x) \in \mathbb{R}[x] \mid p(-x) = p(x)\}$$

  例如 $x^2, x^4 + 3, 2x^6 - 5$ 都在其中。

- **奇函数子空间**：

$$\mathcal{U}_{\text{odd}} = \{p(x) \in \mathbb{R}[x] \mid p(-x) = -p(x)\}$$

  例如 $x, 3x^3 - 2x, 5x^5$ 都在其中。

任务

1. **证明子空间**：说明 $\mathcal{U}_{\text{even}}$ 和 $\mathcal{U}_{\text{odd}}$ 都是 $\mathbb{R}[x]$ 的子空间。
2. **验证性质**：任意多项式 $p(x)$ 可以唯一写成

$$p(x) = p_{\text{even}}(x) + p_{\text{odd}}(x),$$

其中 $p_{\text{even}} \in \mathcal{U}_{\text{even}}, p_{\text{odd}} \in \mathcal{U}_{\text{odd}}$。
👉 即函数空间 = 偶子空间 $\oplus$ 奇子空间。

3. **Python 实践**：
   - 定义几个多项式，用 `p(-x)` 检查它是偶函数还是奇函数；
   - 写一个函数，把任意多项式分解为偶部分和奇部分；
   - 验证结果。

思考提示

任务 1：证明子空间

- 子空间必须满足三个条件：
  1. **包含零向量**（零多项式必须在其中）；
  2. **加法封闭**（两个函数相加仍然在其中）；
  3. **数乘封闭**（任意数乘结果仍然在其中）。
- 偶函数子空间：
  - 零函数 $0(x)$ 满足 $0(-x) = 0(x)$。
  - 若 $p(-x) = p(x)$、$q(-x) = q(x)$，则
    $$(p+q)(-x) = p(-x) + q(-x) = p(x) + q(x) = (p+q)(x)。$$
  - 若 $p(-x) = p(x)$，则
    $$(\lambda p)(-x) = \lambda p(-x) = \lambda p(x) = (\lambda p)(x)。$$

👉 逐条验证三个条件，即可得出这两个集合都是子空间。

任务 2：分解唯一性

- 定义分解公式（为什么可以这样写？）：
$$p_{\text{even}}(x) = \frac{p(x) + p(-x)}{2}, \qquad p_{\text{odd}}(x) = \frac{p(x) - p(-x)}{2}.$$
- 验证偶性：
$$p_{\text{even}}(-x) = \frac{p(-x)+p(x)}{2} = p_{\text{even}}(x)。$$
- 验证奇性：
$$p_{\text{odd}}(-x) = \frac{p(-x)-p(x)}{2} = -p_{\text{odd}}(x)。$$
- 验证还原：

$$p_{\text{even}}(x) + p_{\text{odd}}(x) = p(x)。$$

- 思考唯一性：
  - 如果 $p = u_{\text{even}} + u_{\text{odd}} = v_{\text{even}} + v_{\text{odd}}$，
    那么 $(u_{\text{even}} - v_{\text{even}}) = (v_{\text{odd}} - u_{\text{odd}})$。
  - 左边是偶函数，右边是奇函数，唯一可能是零函数。
  - 所以分解是唯一的。

任务3

1. **多项式表示**
   - 使用 `numpy.poly1d` 定义多项式，例如
     `p = np.poly1d([1, -2, 3])  # p(x) = x^2 - 2x + 3`
   - 系数顺序是“最高次到常数项”。
2. **检查偶/奇性质**
   - 取一组对称点：`x = np.linspace(-3, 3, 7)`。
   - 检查是否为偶函数：比较 `p(-x)` 与 `p(x)`。
   - 检查是否为奇函数：比较 `p(-x)` 与 `-p(x)`。
   - 用 `np.allclose(a, b)` 判断两个数组是否近似相等，避免浮点误差。
     - **解释**：`np.allclose(a, b)` 返回 `True` 表示 $a_i \approx b_i$ 对所有元素成立。
3. **分解为偶部分和奇部分**
   - 利用公式：

$$p_{\text{even}}(x) = \frac{p(x) + p(-x)}{2}, \qquad p_{\text{odd}}(x) = \frac{p(x) - p(-x)}{2}.$$

   - 可以写一个函数，输入 `poly`，输出 `p_even` 和 `p_odd`。
   - 方法一：用公式直接计算；
   - 方法二：根据幂次的奇偶性拆分系数。
4. **验证分解**
   - 检查 $p(x) = p_{\text{even}}(x) + p_{\text{odd}}(x)$ 是否成立；
   - 检查 $p_{\text{even}}(-x) = p_{\text{even}}(x)$ 是否成立；
   - 检查 $p_{\text{odd}}(-x) = -p_{\text{odd}}(x)$ 是否成立。

👉 这样可以直观验证“任意多项式 = 偶部分 + 奇部分”的结论。

练习4：文档特征向量与线性独立性

场景说明

在自然语言处理 (NLP) 中，我们常常用向量来表示文档或句子。
假设我们用一个非常简化的方式：用 **3 维向量** 表示“某文档里出现的词频”。

例如：

- 维度 1：出现了多少次“篮球” 🏀
- 维度 2：出现了多少次“蔡徐坤” 🎤
- 维度 3：出现了多少次“音乐” 🎵

于是，每篇文档都可以看成 $\mathbb{R}^3$ 里的一个向量。

任务

1. 随机生成 3 个文档向量（每个向量有 3 个分量）。
2. 把这 3 个向量拼成矩阵 A。
3. 使用 `numpy.linalg.matrix_rank` 判断它们是否线性独立。
4. 如果它们线性相关，请找出一条关系式（即某个向量能由其他向量表示）。

Python 验证

In [3]:
```python
import numpy as np

# 三个文档向量（词频统计）
doc1 = np.array([3, 5, 2])   # 篮球3, 蔡徐坤5, 音乐2
doc2 = np.array([6, 10, 4])  # 篮球6, 蔡徐坤10, 音乐4
doc3 = np.array([1, 0, 7])   # 篮球1, 蔡徐坤0, 音乐7

# 构造矩阵 A（每一列是一个文档向量）
A = np.column_stack([doc1, doc2, doc3])

# 判断线性独立性
rank = np.linalg.matrix_rank(A)

print("矩阵 A:")
print(A)
print(f"矩阵的秩 = {rank}")

if rank == 3:
    print("这三个文档向量线性独立")
else:
    print("这三个文档向量线性相关")

    # 找出线性关系
    # 解线性方程组: a*doc1 + b*doc2 + c*doc3 = 0
    # 找非零解
    coeff = np.linalg.svd(A)[2][-1]  # 最小奇异值对应的右奇异向量
    print("线性关系系数:", coeff)

    # 验证关系
    print("验证:", np.dot(coeff, A.T))
```

```
矩阵 A:
[[ 3  6  1]
 [ 5 10  0]
 [ 2  4  7]]
矩阵的秩 = 2
这三个文档向量线性相关
线性关系系数: [ 8.94427191e-01 -4.47213595e-01 -2.77555756e-17]
验证: [1.30451205e-15 2.66453526e-15 6.93889390e-16]
```

思考提示

- 注意观察 `doc1` 和 `doc2`，它们是否存在关系？
- 如果存在比例关系，那么这两个文档向量没有"新信息"，说明它们线性相关。
- 在实际 NLP 应用里，这意味着：有些文档虽然看起来不同，但在向量表示上可能**冗余**。

练习5：图像向量与线性独立性

场景说明

在图像处理中，每张灰度图片都可以展平成一个向量。
假设我们有三张 4×4 的小图片，把它们拉平成 $\mathbb{R}^{16}$ 里的向量。

- 如果三张图片线性无关 → 它们包含不同的"信息维度"；
- 如果三张图片线性相关 → 至少有一张可以由另外两张线性组合得到，说明信息冗余。

任务

1. 生成三张 4×4 的随机灰度图像（用 `numpy.random`）。
2. 把它们 reshape 成 16 维向量，并组成矩阵 A。
3. 用 `numpy.linalg.matrix_rank` 判断是否线性独立。
4. 尝试构造一个线性相关的例子（例如把一张图设为另一张图的线性组合），再用 rank 验证。

Python 验证

In [5]:
```python
import numpy as np

# 设置随机数种子，保证结果可复现
rng = np.random.default_rng(42)

# 1. 生成三张 4x4 的随机灰度图像
```

```
img1 = rng.integers(0, 256, size=(4,4))
img2 = rng.integers(0, 256, size=(4,4))
img3 = rng.integers(0, 256, size=(4,4))

# 2. 展平成向量
v1 = img1.flatten()
v2 = img2.flatten()
v3 = img3.flatten()

# 3. 拼接成矩阵 A（每列是一个图像向量）
A = np.column_stack((v1, v2, v3))

# 4. 计算矩阵的秩，判断线性独立性
rank_random = np.linalg.matrix_rank(A)
print(f"随机生成的三张图像的秩: {rank_random}")
if rank_random == 3:
    print("这三张随机图像是线性独立的")
else:
    print("这三张随机图像是线性相关的")

# 5. 人为制造线性相关：让 img3 = 2*img1 + 3*img2
img3_dependent = 2 * img1 + 3 * img2
v3_dependent = img3_dependent.flatten()

# 6. 拼接成新的矩阵 B
B = np.column_stack((v1, v2, v3_dependent))

# 7. 验证线性相关性
rank_dependent = np.linalg.matrix_rank(B)
print(f"\n线性相关构造的三张图像的秩: {rank_dependent}")
if rank_dependent < 3:
    print("验证成功：这三张图像是线性相关的")
else:
    print("验证失败：这三张图像仍然是线性独立的")
```

```
随机生成的三张图像的秩: 3
这三张随机图像是线性独立的

线性相关构造的三张图像的秩: 2
验证成功：这三张图像是线性相关的
```