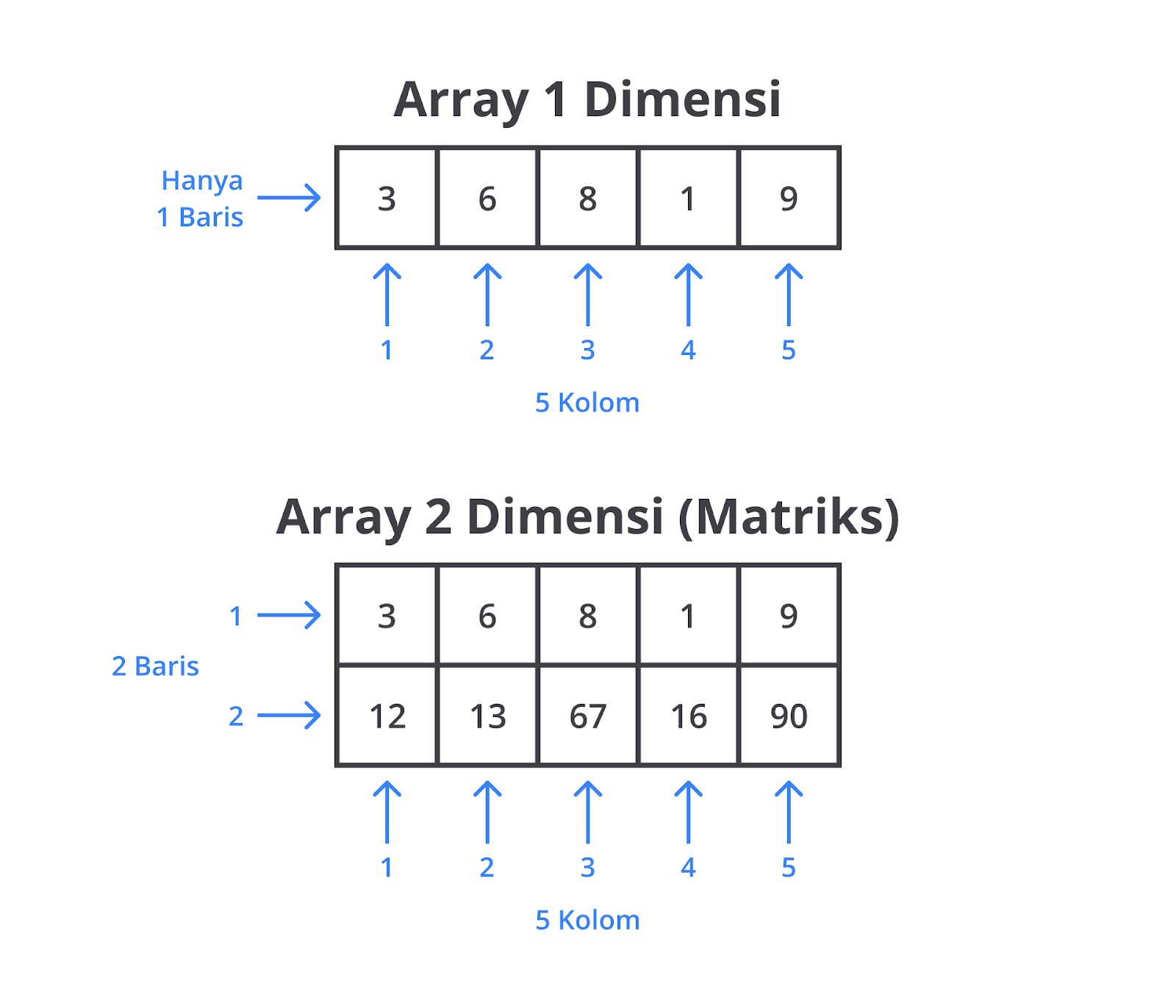
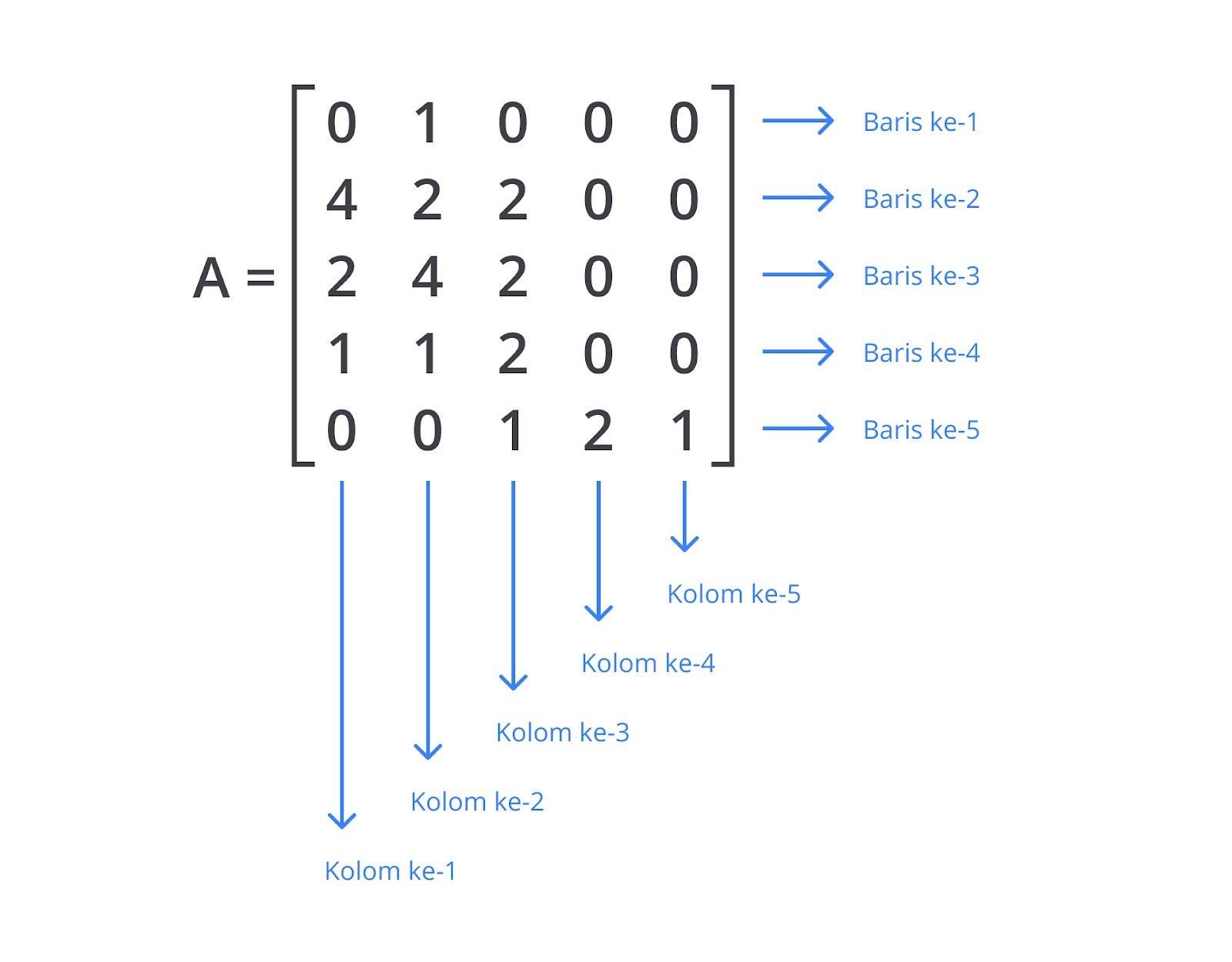
**Fundamental Matriks**

Pada materi sebelumnya, kita telah mempelajari cara menyimpan data yang sama menggunakan array dalam Python dengan list. Untuk pengingat, array merupakan salah satu jenis struktur data linear dan terdiri dari kumpulan elemen bertipe data sama dengan indeks yang berurutan atau linear.

Sebenarnya, array adalah jenis struktur data 1 dimensi yang menyimpan semua datanya secara linear. Pada materi ini, kita akan mempelajari jenis array 2 dimensi, yakni matriks.



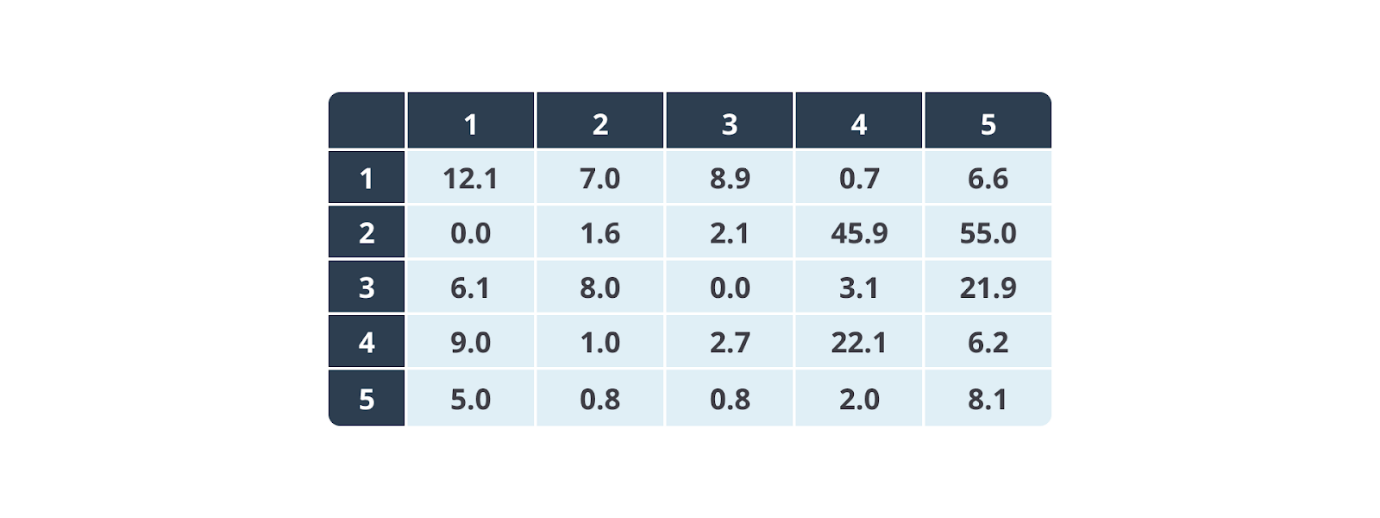
Matriks pada matematika merupakan himpunan yang terdiri dari bilangan atau elemen berdasarkan baris dan kolom. Dalam matematika, struktur matriks sebagai berikut.



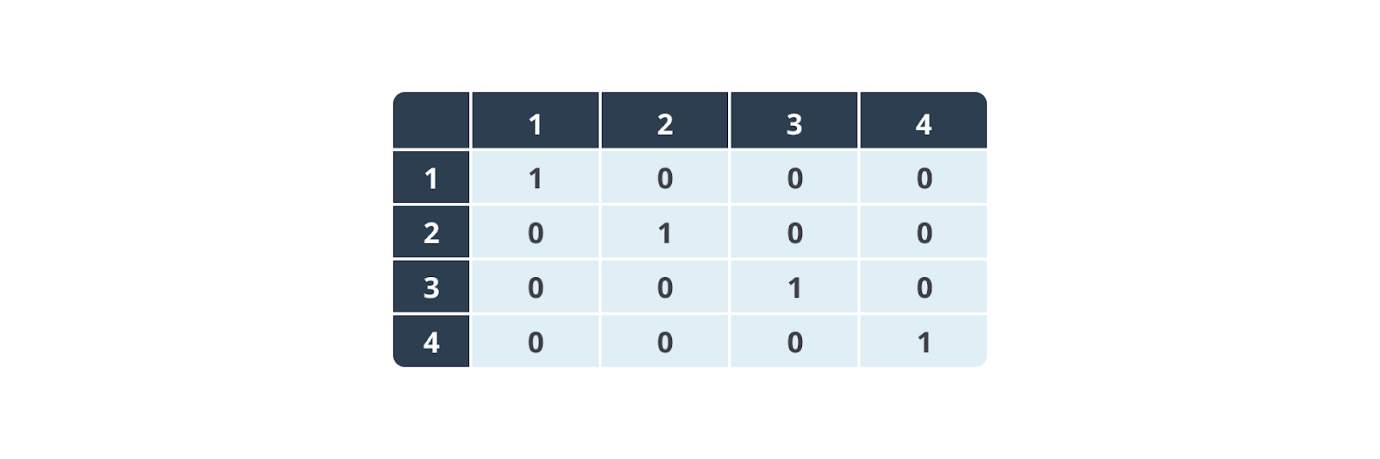
Dalam matematika, penamaan baris dan kolom dibuat secara berurutan, baris ke-1 dimulai dari atas hingga ke bawah dan kolom ke-1 dimulai dari kiri ke kanan.

Contoh matriks dalam matematika beragam jenisnya, beberapa di antaranya sebagai berikut.

1. **Matriks Pengukuran**  
   Matriks pengukuran adalah jenis matriks dengan indeks (i, j) yang merepresentasikan suatu titik koordinat. Setiap elemen dari matriks ini merepresentasikan hasil pengukuran pada suatu titik koordinat tertentu dan termasuk bilangan real atau tipe data float.



1. **Matriks Satuan**  
   Selanjutnya adalah matriks satuan dengan elemen bernilai hanya 0 atau 1. Setiap elemen matriks ini bertipe data integer.

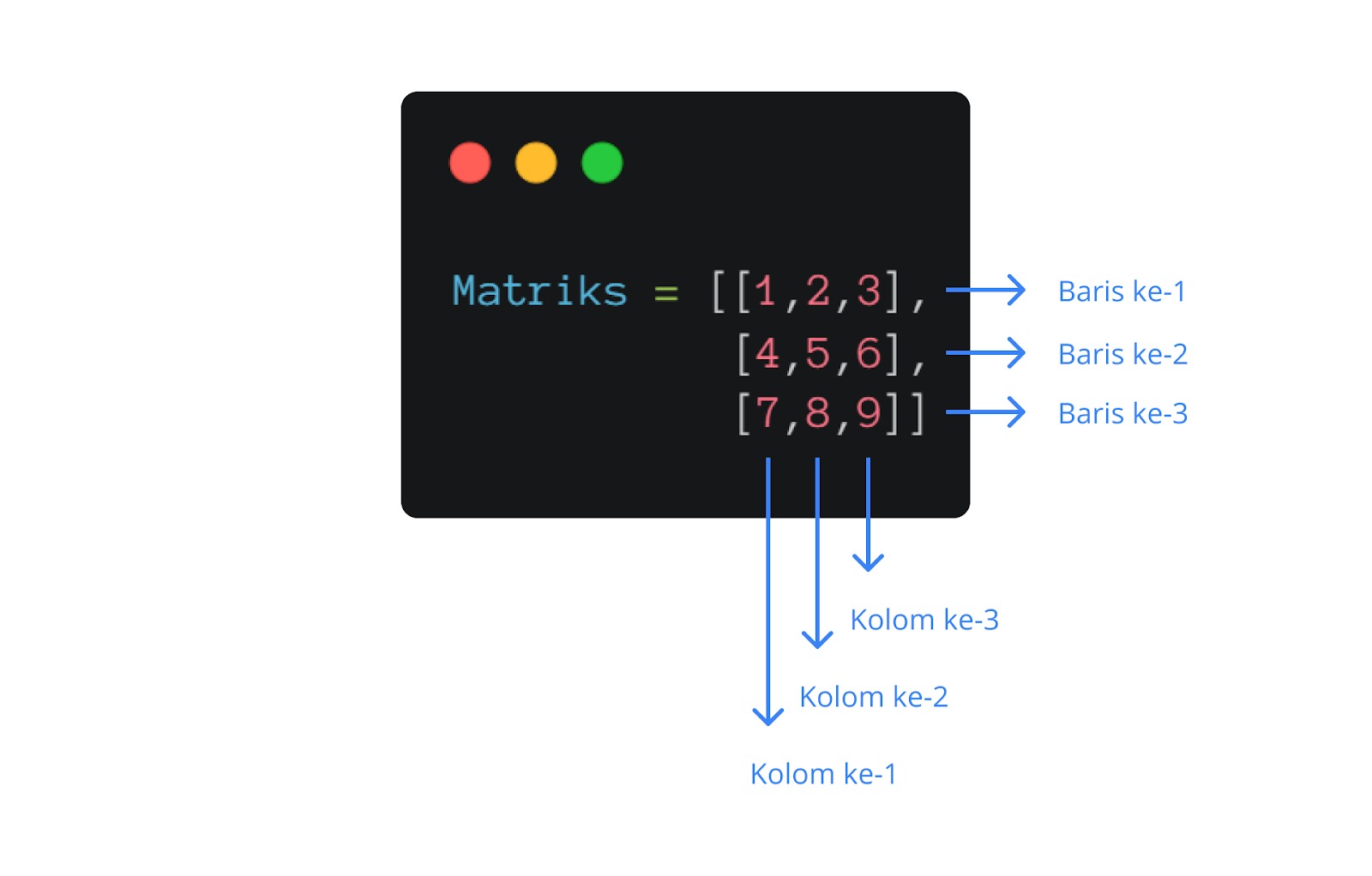


Gambar di atas merupakan matriks satuan dengan indeks baris adalah 1 sampai dengan 4 dan indeks kolom 1 sampai dengan 4.

Masih banyak penggunaan matriks dalam matematika yang dapat kita pelajari, seperti matriks nama hari yang merepresentasikan nama hari ke-1 hingga hari ke-7 dan elemen matriksnya bertipe string, matriks value, serta masih banyak lagi.

Anda pun dapat melakukan berbagai operasi pada matriks, seperti penjumlahan, perkalian, menentukan determinan, *transpose*, dan sebagainya. Bahkan matriks juga dapat dipakai untuk persoalan algoritmik, yakni untuk menyimpan informasi yang cirinya ditentukan oleh dua dimensi atau baris dan kolom, seperti *cell* pada *spreadsheet*.

Sementara itu dalam pemrograman, matriks adalah kumpulan data yang diatur dalam **bentuk tabel dua dimensi**dengan setiap elemennya terdefinisi berdasarkan baris dan kolom. Matriks dalam pemrograman diimplementasikan menggunakan array dua dimensi. Bahkan dalam Python, matriks dapat diimplementasikan menggunakan *nested list* atau list di dalam list.



Dalam gambar di atas, kita mendeklarasikan variabel "matriks" dan menginisialisasikannya dengan **nested list** atau list di dalam list. Perhatikan lebih baik,  pada gambar tersebut terdapat dua kurung siku "[]" dan kurung siku pertama adalah list yang membungkus tiga list di dalamnya, yakni "[1, 2, 3], [4, 5, 6], [7, 8, 9]".

Matriks dalam pemrograman dapat kita simpulkan sebagai berikut.

1. Matriks adalah kumpulan data yang diatur dalam **bentuk tabel dua dimensi**dengan setiap elemennya terdefinisi berdasarkan baris dan kolom.
2. Setiap elemen matriks dapat diakses melalui metode indexing jika kedua indeks telah diketahui.
3. Elemen matriks dideklarasikan memiliki tipe homogen yang artinya elemen tersebut harus mempunyai tipe data yang sama. Jika elemen tersebut adalah bilangan real, seluruh elemen lainnya pun adalah bilangan real. Walaupun dalam list Python Anda dapat membuat matriks dengan tipe data berbeda, alangkah lebih baik jika tetap mengikuti aturan ini.

matriks = [[1, 2, 3],

[4, 5, 6],

[7, 8, 9]]

Pada kode di atas, kita mendeklarasikan variabel "matriks" dan menginisialisasikan dengan nested list. Output yang dihasilkan adalah matriks "[[1, 2, 3], [4, 5, 6], [7, 8, 9]]".

Namun, perlu diingat bahwa mendeklarasikan matriks menggunakan list sangat memakan banyak memori. Masih ingatkah Anda dengan materi array yang menyatakan bahwa setiap nilai atau elemen dalam list akan disimpan pada masing-masing tempat memori? Hal ini berlaku juga pada matriks.

Jika kita membuat matriks dengan 100 kolom dan 100 baris akan menghasilkan 10.000 elemen integer dalam matriks tersebut dan menggunakan 10.000 penyimpanan untuk menampung semua elemen integer.

Oleh karena itu, menggunakan nested list sebagai matriks dianggap cara yang cukup praktis, tetapi tidak efektif dalam mengelola penyimpanan memori. Jika Anda ingin membuat matriks dengan jumlah elemen yang besar, programmer Python biasanya menggunakan librarytambahan seperti NumPy untuk melakukan tugasnya. Library merupakan sekumpulan kode yang telah dibuat oleh developer atau programmer dan disediakan kepada pengguna lain agar dapat digunakan ulang dalam pengembangan program atau perangkat lunak.

Library NumPy sering dipakai oleh programmer Python untuk melakukan tugas-tugas dalam ranah *science* dan *engineering*karena dianggap memiliki penggunaan penyimpanan memori yang efisien.

import numpy

matriks = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8 ,9]])

Pada kode di atas, kita mengimpor library "numpy" terlebih dahulu untuk mengambil fungsi-fungsi atau kode yang berada pada library tersebut. Selanjutnya, kita mendeklarasikan variabel "matriks" dan menginisialisasikannya dengan numpy array.

Ingat bahwa matriks adalah array dua dimensi sehingga pada library NumPy kita menggunakan fungsi ".array()" untuk membuat matriks atau array dua dimensi. Pada kode di atas, kita menggunakan fungsi ".array()" dengan nilai di dalamnya adalah nested list.

Jika Anda bertanya, apakah library NumPy bisa dipakai juga untuk array? Jawabannya adalah iya, Anda bisa menggunakan library ini untuk membuat array juga. Programmer sering kali menggunakan library NumPy jika perlu membuat array atau matriks dalam Python.

Dengan menggunakan matriks yang sama, yakni "[[1, 2, 3], [4, 5, 6], [7, 8, 9]]" menghasilkan jumlah memori yang berbeda terhadap masing-masing metode. Jika kita menggunakan NumPy, memori yang digunakan untuk keseluruhan elemen adalah 72. Namun, jika kita menggunakan list, memori yang digunakan untuk keseluruhan elemen adalah 240.

Inilah alasan banyak programmer Python cenderung memilih NumPy untuk memproses matriks bahkan hingga ke tahap machine learning.

**Implementasi Matriks pada Python**

Sebelumnya kita telah belajar cara mengimplementasikan matriks yang merupakan array dua dimensi menggunakan nested list. Ingat bahwa setiap elemen matriks dideklarasikan memiliki tipe homogen, yang artinya elemen tersebut harus memiliki tipe data yang sama. Jika Anda mendeklarasikan suatu elemen bertipe data float, elemen lainnya pun adalah float.

Sekarang, kita pelajari cara mengimplementasikan matriks pada Python lebih dalam. Kita akan mulai dengan cara mendeklarasikan matriks hingga mengakses setiap elemen matriks dengan metode indexing.

**Deklarasi Matriks**

Ada dua cara untuk mendeklarasikan matriks menggunakan Python sebagai berikut.

1. **Deklarasi sekaligus inisialisasi nilai matriks**.  
   Cara pertama adalah mendeklarasikan matriks sekaligus menginisialisasikan nilainya dengan ukuran N baris dan M Kolom (NxM). Cara ini dilakukan jika kita telah mengetahui nilai yang perlu diberikan.

Berikut adalah struktur untuk mendeklarasikan matriks dengan menginisialisasikan nilainya sekaligus.



Gambar di atas merupakan struktur jika kita ingin mendeklarasikan matriks dengan ukuran nxm. Ingat bahwa tipe elemen tersebut akan bergantung pada nilai yang diberikan. Jika nilai yang diberikan adalah bilangan bulat, tipe elemen adalah integer. Jika nilai yang diberikan adalah bilangan real, tipe elemen adalah float.

1. **Deklarasi dengan nilai default**.  
   Cara kedua adalah mendeklarasikan matriks dengan nilai default. Sebagaimana materi array, nilai default ditentukan oleh kesepakatan bersama sesuai kebutuhan dengan nilainya di luar rentang yang ditentukan. Misalnya, tim Anda menentukan nilai dalam list harus berkisar dari 1 hingga 10. Kita bisa menyepakati nilai "0" sebagai nilai default karena di luar jangkauan yang disepakati (1-10). Cara kedua ini melibatkan list comprehension yang sama seperti pada materi array. Struktur dari deklarasi dengan nilai default sebagai berikut.



Terlihat pada struktur tersebut, cara kedua ini menggunakan dua metode sekaligus, yakni **nested list** dan **nested for**. Kita menggunakan nested for atau for bersarang untuk membuat baris dan kolom. Perhatikan baik-baik, perulangan dalam atau perulangan kedua diapit oleh tanda siku "[]" yang artinya hasil dari perulangan kedua adalah baris pada matriks, sedangkan perulangan pertama atau perulangan luar menghasilkan kolom pada matriks.  
  
Nilai dari **<default-val>**ditentukan kesepakatan bersama, misalnya jika range yang disepakati adalah 1 hingga 10, kita bisa memilih 0 untuk nilai default-nya. Ada pula **<n>**sebagai jumlah baris matriks yang ingin dibuat dan **<m>** sebagai jumlah kolom matriks yang diinginkan.

matriks = [[0 for j in range(4)] for i in range(3)]

Pada kode di atas, kita mendeklarasikan variabel matriks dan menginisialisasikannya dengan nested list dan nested for serta nilai default-nya adalah 0. Matriks yang dibuat pada program di atas adalah matriks value dengan setiap elemennya bertipe integer serta memiliki ukuran baris = 3 dan ukuran kolom = 4.

**Akses Elemen Matriks**

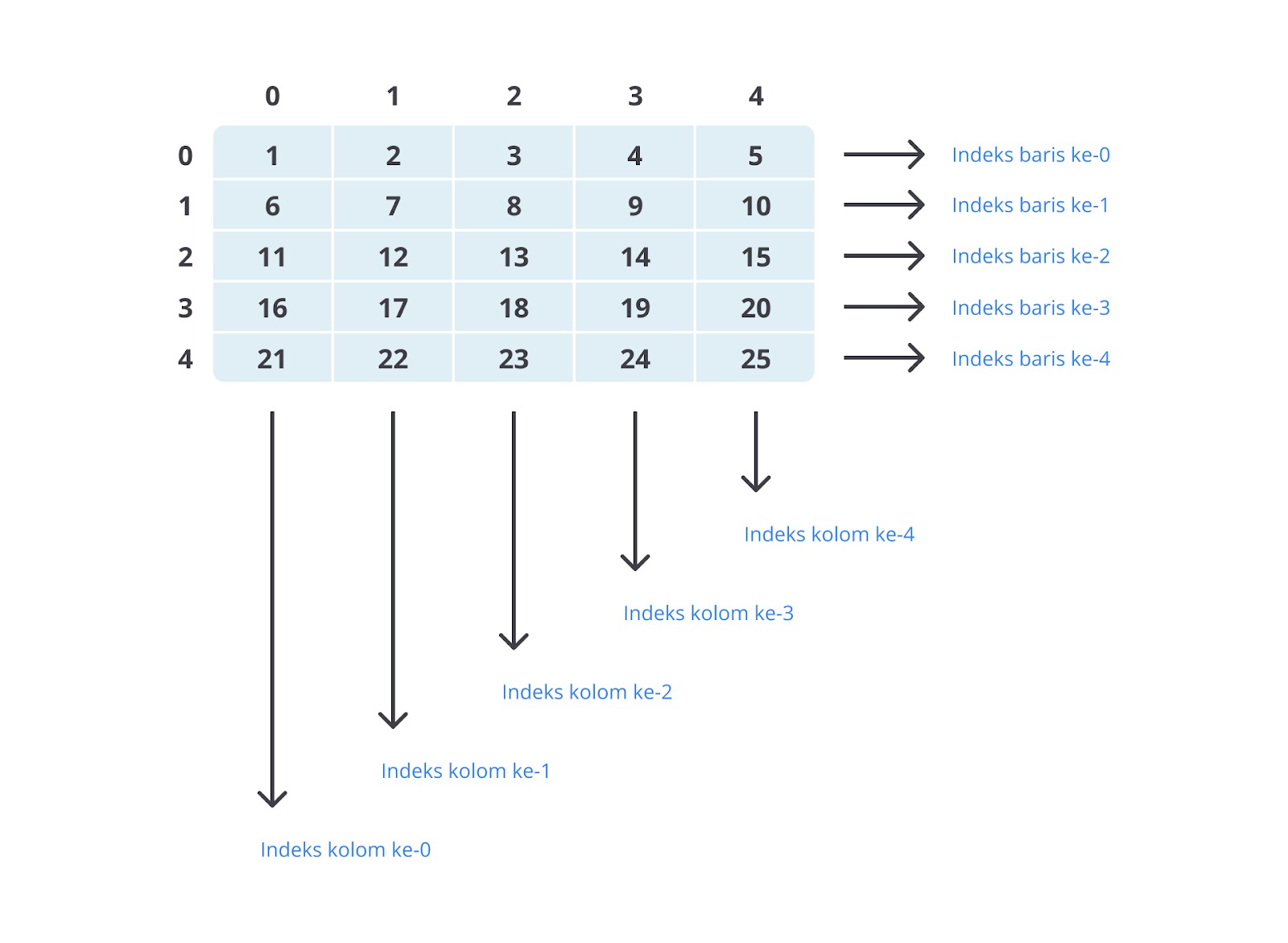
Sekarang, mari pelajari cara mengakses elemen pada matriks. Ingat bahwa matriks merupakan tabel data yang terdiri dari baris dan kolom. Jadi, jika Anda ingin mengakses elemen dari matriks, perlu mengetahui indeks dari baris dan kolom.

Kita akan mengakses elemen matriks menggunakan metode indexing. Ini artinya Anda perlu mengetahui indeks dari baris dan kolom yang ingin diakses. Berikut adalah struktur untuk mengakses elemen matriks dengan metode indexing.



Berdasarkan struktur di atas, <namamatriks> merupakan variabel yang berisi nilai matriks, <nbrs> merupakan nomor indeks baris yang ingin diakses, dan <nkol> nomor indeks kolom yang ingin diakses.

Perhatikan gambar di bawah ini untuk memahami maksud dari indeks baris dan kolom.



Asumsikan Anda memiliki matriks dengan ukuran 5 baris dan 5 kolom yang setiap elemennya berisi angka dari 1 hingga 25 seperti gambar di atas. Indeks baris ke-0 dimulai dari 1 hingga 5, indeks baris ke-1 dimulai dari 6 hingga 10, dan seterusnya. Indeks kolom ke-0 dimulai dari "1, 6, 11, 16, 21", begitu pun indeks kolom ke-1 dimulai dari "2, 7, 12, 17, 22", dan seterusnya.

Jika kita ingin mengakses nilai 12 pada matriks di atas, nilai tersebut berada pada indeks baris ke-2 dan indeks kolom ke-1. Dalam pemrograman, nilai tersebut bisa diasumsikan dengan "[2][1]" dengan nilai 2 adalah indeks atau nomor baris dan nilai 1 adalah indeks atau nomor kolom.

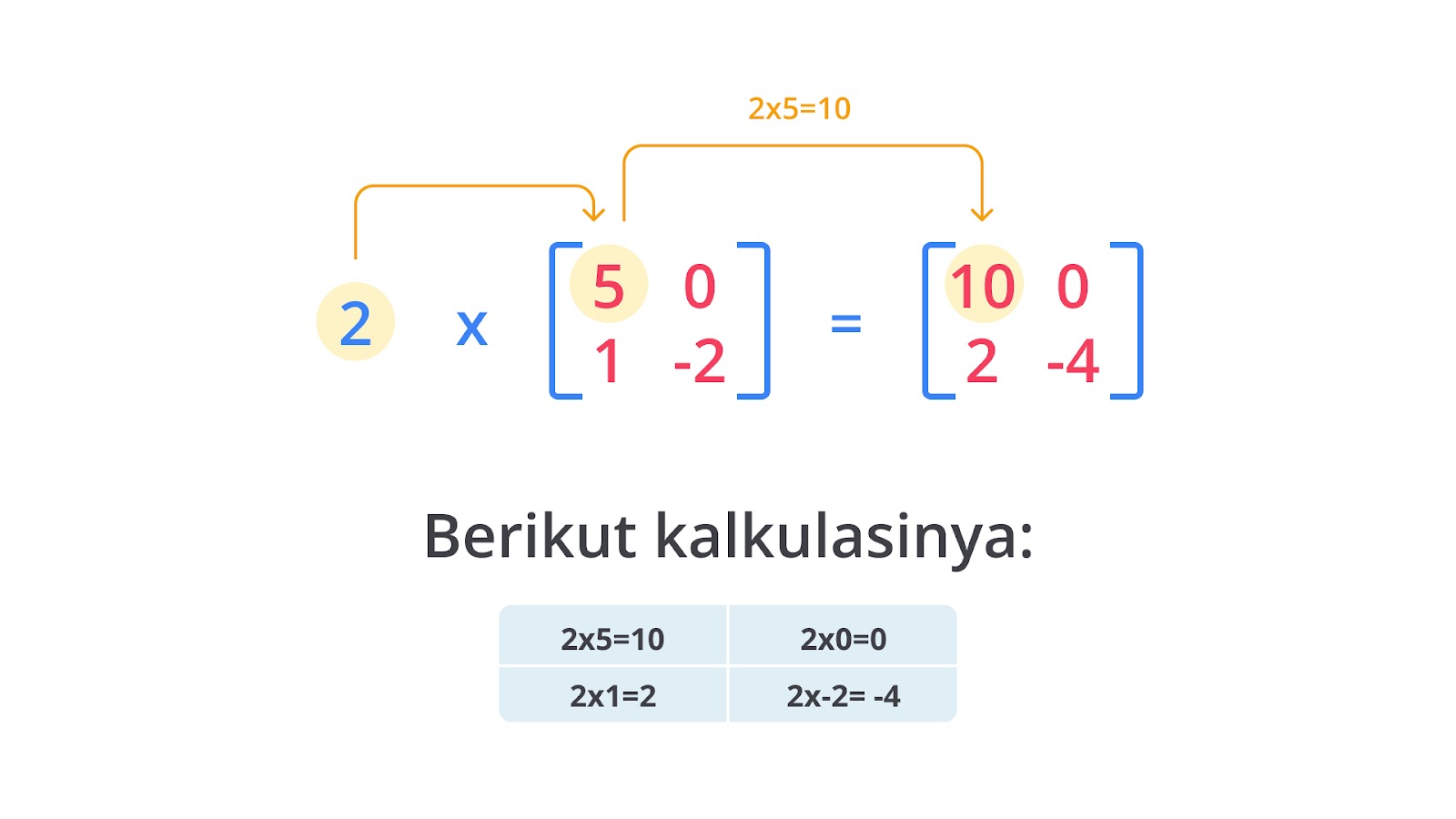
**Operasi Matriks pada Python**

Dalam matematika ataupun pemrograman, operasi matriks dapat melibatkan dua matriks sekaligus atau pun satu matriks saja. Beberapa operasi tersebut di antaranya sebagai berikut.

1. Operasi 1 matriks.
   * Menghitung total semua elemen matriks.
   * Mengalikan elemen matriks dengan konstanta.
   * Transpose matriks.
   * Inverse matriks.
   * Menentukan determinan, dan sebagainya.
2. Operasi 2 matriks:
   * Menambahkan dua matriks.
   * Mengalikan dua matriks.
   * Pembagian dua matriks, dan sebagainya.

Dari berbagai operasi matriks yang tersedia, mari kita pelajari salah satu di antaranya, yakni **mengalikan elemen matriks dengan konstanta**.

Dalam matematika, mengalikan elemen matriks dengan konstanta dapat diilustrasikan seperti gambar berikut. Asumsikan ukuran matriksnya adalah 2x2 atau 2 baris dan 2 kolom.



Pada ilustrasi tersebut, kita mengalikan matriks "[5, 0], [1, -2]" dengan konstanta "2". Nilai konstanta tersebut kemudian dikalikan dengan setiap elemen matriks. Kalkulasinya dapat kita urai seperti berikut.

1. Pertama, konstanta "2" akan dikalikan dengan elemen 5 yang menghasilkan nilai 10 (**2 X 5 = 10).**
2. Kedua, konstanta "2" akan dikalikan dengan elemen 0 yang menghasilkan nilai 0 **(2 X 0 = 0).**
3. Ketiga, konstanta "2" akan dikalikan dengan elemen 1 yang menghasilkan nilai 2 (**2 X 1 = 2**).
4. Terakhir, konstanta "2" akan dikalikan dengan elemen -2 yang menghasilkan nilai -4 **(2x-2 = -4).**

# Membuat matriks 2x2

var\_mat = [[5, 0],

[1, -2]]

def\_mat = [[0 for j in range(2)] for i in range(2)]

for i in range(len(var\_mat)):

for j in range(len(var\_mat[0])):

def\_mat[i][j] = var\_mat[i][j]\*2

print(def\_mat)

Output:

[[10, 0], [2, -4]]

Perhatikan penjelasan berikut untuk memahami contoh di atas.

1. Pertama, kita mendeklarasikan variabel "var\_mat" dan menginisialisasikannya dengan matriks yang diinginkan. Di sini, matriks yang digunakan berukuran 2x2 atau 2 baris dan 2 kolom.
2. Kedua, kita perlu mendeklarasikan variabel "def\_mat" sebagai matriks default baru dengan nilai (0). Matriks dengan nilai default ini harus berukuran sama dengan matriks yang asli.
3. Ketiga, kita akan melakukan perulangan berdasarkan dua kondisi. Kondisi pertama adalah perulangan berdasarkan banyaknya list di dalam matriks ([5, 0], [1, -2]) yang merepresentasikan baris. Kondisi kedua adalah perulangan berdasarkan banyaknya elemen pada setiap list (5,0 dan 1,-2).  
     
   Perulangan pertama (i) akan mengulang sebanyak dua kali karena variabel "var\_mat" memiliki dua list di dalamnya. Perulangan kedua (j) akan mengulang sebanyak dua kali karena setiap elemen list yang diambil memiliki dua nilai. Jadi, elemen list pertama adalah "5 dan 0" serta elemen list kedua adalah "1 dan -2".  
     
   Pada setiap perulangannya, nilai i = 0, 1; j =0, 1. Ingat, ini disebabkan kita mengambil panjang dari variabel "var\_mat" yang memang memiliki dua list dan panjang elemen dari setiap list-nya. Fungsi "range()" selalu memulai bilangan dari nol.
4. Selanjutnya, kita akan meng-update matriks default dengan nilai yang dihasilkan berdasarkan perulangan dan mengalikannya dengan konstanta 2.  
     
   Jadi,**elemen def\_mat[0][0] yang bernilai 0**akan diubah dengan elemen **var\_mat[0][0] yang bernilai 5** lalu dikalikan dengan 2 dan hasilnya adalah 10. **Elemen def\_mat[0][0] yang awalnya bernilai 0 berubah menjadi 10.**
5. Perulangan ini terus terjadi hingga semua kondisi terpenuhi dan menyebabkan semua elemen variabel "def\_mat" berubah sesuai perkalian dengan konstanta 2.

*Phew,*materi kali ini sangat erat dengan perhitungan matematika. Cara yang kita pelajari di atas terbilang cukup rumit. Nah, sebenarnya ada cara yang lebih efektif, yaitu dengan menggunakan library NumPy. Perhatikan kode di bawah ini.

import numpy as np

var\_mat = np.array([[5, 0],

[1, -2]])

result = var\_mat \* 2

print(result)

Pada contoh di atas, kita melakukan operasi perkalian matriks dengan konstanta yang sama seperti sebelumnya. Bedanya, kali ini kita menggunakan library NumPy untuk mempermudah pengkodean. Bisa Anda lihat bahwa dengan menggunakan NumPy kita tidak perlu lagi menggunakan **nested for**dan cukup mengalikan matriks dengan konstanta yang diinginkan. Dalam hal ini ditunjukkan pada kode berikut.