Praktek-1

```
 # impor library numpy

import numpy as np
```

Penjelasan :

Library **NumPy** diimpor dan diberi alias **np**. Ini adalah langkah standar agar bisa menggunakan fungsi-fungsi dari NumPy, seperti np.array(), dengan penulisan yang lebih singkat.

```
# membuat array dengan numpy
nilai_siswa = np.array([85, 55, 40, 90])
```

Penjelasan :

Baris ini membuat **array NumPy** bernama nilai_siswa yang berisi data nilai: 85, 55, 40, dan 90.
Berbeda dengan list Python biasa, array NumPy memiliki kelebihan dalam hal kecepatan dan operasi matematis

```
# akses data pada array
print(nilai_siswa[3])
```

Pejelasan :

Baris ini mencetak nilai elemen ke-4 dalam array nilai_siswa.
Ingat bahwa indeks dalam Python dimulai dari **0**, sehingga:

Indeks	**Nilai**
0	85
1	55
2	40
3	90

Jadi nilai_siswa[3] bernilai **90**, dan itu yang akan ditampilkan di output.

Praktek-2

```
# impor libaray numpy
import numpy as np
```

Penjelasan :

Library **NumPy** diimpor dan diberi alias **np**. Ini adalah langkah standar agar bisa menggunakan fungsi-fungsi dari NumPy, seperti np.array(), dengan penulisan yang lebih singkat.

```
# membuat array dengan numpy
nilai_siswa_1 = np.array([75, 65, 45, 80])
nilai_siswa_2 = np.array([[85, 55, 40], [50, 40, 99]])
```

Penjelasan :

- nilai_siswa_1 adalah **array 1 dimensi (1D)** berisi nilai: 75, 65, 45, 80.
- nilai_siswa_2 adalah **array 2 dimensi (2D)** berbentuk 2 baris × 3 kolom, yang menyerupai tabel:

```
# cara akses elemen array
print(nilai_siswa_1[0])
print(nilai_siswa_2[1][1])
```

Penjelasan :

Untuk membaca atau mengambil nilai dari array, kita gunakan indeks. Karena indeks dimulai dari 0:

- nilai_siswa_1[0] mengambil elemen pertama dari array satu dimensi (hasilnya 75).
- nilai_siswa_2[1][1] mengambil elemen di baris kedua dan kolom kedua dari array dua dimensi (hasilnya 40).

```
# mengubah nilai elemen array
nilai_siswa_1[0] = 88
nilai_siswa_2[1][1] = 70
```

Penjelasan :

Nilai array bisa diubah dengan menunjuk langsung ke indeksnya. Di sini:

- Nilai pertama pada nilai_siswa_1 diubah dari 75 menjadi 88.
- Nilai di baris kedua kolom kedua pada nilai_siswa_2 diubah dari 40 menjadi 70.

```
# cek perubahannya dengan akses elemen array
print(nilai_siswa_1[0])
print(nilai_siswa_2[1][1])
```

Penjelasan :

Kode ini memverifikasi perubahan sebelumnya:

- Sekarang nilai_siswa_1[0] akan mencetak 88.
- nilai_siswa_2[1][1] akan mencetak 70, sesuai nilai yang telah dimodifikasi.

```
# Cek ukuran dan dimensi array
print("Ukuran Array : ", nilai_siswa_1.shape)
print("Ukuran Array : ", nilai_siswa_2.shape)
print("Dimensi Array : ", nilai_siswa_2.ndim)
```

Penjelasan :

- shape menunjukkan ukuran array:

- Untuk nilai_siswa_1, hasilnya (4,) artinya array dengan 4 elemen.
- Untuk nilai_siswa_2, hasilnya (2, 3) artinya array dengan 2 baris dan 3 kolom.

- ndim menunjukkan jumlah dimensi array:

- nilai_siswa_2.ndim bernilai 2 karena merupakan array dua dimen

Praktek-3

```
# impor library numpy
import numpy as np
```

Penjelasan :

Library **NumPy** diimpor dan diberi alias **np**. Ini adalah langkah standar agar bisa menggunakan fungsi-fungsi dari NumPy, seperti np.array(), dengan penulisan yang lebih singkat.

```
# membuat array
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
```

Penjelasan :

Di sini Anda membuat dua buah array satu dimensi (a dan b) menggunakan np.array(). Array a berisi angka [1, 2, 3] dan b berisi [4, 5, 6]. Kedua array ini memiliki panjang (jumlah elemen) yang sama, sehingga bisa dilakukan operasi antar elemen (element-wise operation).

```
# menggunakan operasi penjumlahan pada 2 array
print(a + b)       # array([5, 7, 9])
```

Penjelsan :

Pada baris ini, dilakukan penjumlahan antar dua array. Dalam NumPy, ketika dua array dengan ukuran yang sama dijumlahkan, operasi dilakukan **element-wise**, artinya:

- 1 + 4 = 5

- 2 + 5 = 7
- 3 + 6 = 9

Hasilnya adalah array baru [5, 7, 9], yang langsung dicetak ke layar.

```
# Indexing dan Slicing pada Array
arr = np.array([10, 20, 30, 40])
print(arr[1:3])    # array([20, 30])
```

Penjelasan :

Anda membuat array baru bernama arr berisi empat elemen. Kemudian dilakukan **slicing**, yaitu mengambil sebagian elemen dari array menggunakan notasi indeks. arr[1:3] berarti mengambil elemen mulai dari indeks ke-1 sampai sebelum indeks ke-3:

- Indeks ke-1: 20
- Indeks ke-2: 30
  Jadi, hasil slicing ini adalah [20, 30]

```
# iterasi pada array
for x in arr:
    print(x)
```

Penjelasan :

Anda membuat array baru bernama arr berisi empat elemen. Kemudian dilakukan **slicing**, yaitu mengambil sebagian elemen dari array menggunakan notasi indeks. arr[1:3] berarti mengambil elemen mulai dari indeks ke-1 sampai sebelum indeks ke-3:

- Indeks ke-1: 20
- Indeks ke-2: 30
  Jadi, hasil slicing ini adalah [20, 30]

Praktek-4

```
# membuat array
arr = [1, 2, 3, 4, 5]
```

Penjelasan :

Di baris ini, Anda membuat sebuah **list** Python bernama arr, yang berisi lima elemen integer: 1, 2, 3, 4, dan 5. Meskipun dalam istilah umum disebut “array”, secara teknis ini adalah list Python, bukan array dari library NumPy. List adalah struktur data bawaan Python yang mendukung berbagai operasi seperti indexing, slicing, dan iterasi.

```
# Linear Traversal ke tiap elemen arr
print("Linear Traversal: ", end=" ")
for i in arr:
    print(i, end=" ")
print()
```

Penjelasan :

Bagian ini melakukan **linear traversal**, yaitu menelusuri setiap elemen dalam list satu per satu dari awal hingga akhir.

- print("Linear Traversal: ", end=" ") mencetak label di awal tanpa berpindah baris (karena end=" ").
- for i in arr: adalah loop for yang mengambil tiap elemen i dari list arr.
- print(i, end=" ") mencetak elemen i di baris yang sama, dipisahkan oleh spasi, bukan baris baru.
- print() di akhir baris digunakan untuk memastikan bahwa kursor pindah ke baris baru setelah traversal selesai.

Praktek-5

```
# membuat array
arr = [1, 2, 3, 4, 5]
```

Penjelasan :

Di baris ini, Anda membuat sebuah **list** Python bernama arr, yang berisi lima elemen integer: 1, 2, 3, 4, dan 5. Meskipun dalam istilah umum disebut “array”, secara teknis ini adalah list Python, bukan array dari library NumPy. List adalah struktur data bawaan Python yang mendukung berbagai operasi seperti indexing, slicing, dan iterasi.

```
# Reverse Traversal dari elemen akhir
print("Reverse Traversal: ", end="")
for i in range(len(arr) - 1, -1, -1):
    print(arr[i], end=" ")
print()
```

Penjelasan :

a. print("Reverse Traversal: ", end="")

Baris ini mencetak label awal **"Reverse Traversal:"** di layar tanpa pindah ke baris baru, karena menggunakan end="" yang mengatur karakter akhir sebagai string kosong (default-nya adalah pindah baris).

b. for i in range(len(arr) - 1, -1, -1):

Baris ini adalah inti dari **reverse traversal** (penelusuran dari belakang ke depan). Fungsi range() menghasilkan urutan indeks dari:

- len(arr) - 1 → indeks terakhir (dalam hal ini 4)
- -1 → berhenti sebelum mencapai indeks -1 (berarti hingga indeks 0)
- -1 → langkah negatif, artinya iterasi mundur (dari besar ke kecil)

Jadi, range(4, -1, -1) menghasilkan urutan indeks: 4, 3, 2, 1, 0.

c. print(arr[i], end=" ")

Baris ini mencetak elemen arr[i] satu per satu dari belakang ke depan (dari indeks 4 ke 0), dan end=" " menjaga agar semuanya tetap dalam satu baris, dipisahkan oleh spasi.

d. print()

Terakhir, baris ini digunakan untuk mengakhiri output dengan pindah ke baris baru setelah traversal selesai.

Praktek-6

```
# membuat array
arr = [1, 2, 3, 4, 5]
```

Penjelasan :

Di baris ini, Anda membuat sebuah **list** Python bernama arr, yang berisi lima elemen integer: 1, 2, 3, 4, dan 5. Meskipun dalam istilah umum disebut “array”, secara teknis ini adalah list Python, bukan array dari library NumPy. List adalah struktur data bawaan Python yang mendukung berbagai operasi seperti indexing, slicing, dan iterasi.

```
# mendeklarasikan nilai awal
n = len(arr)
i = 0
```

Penjelasan :

Pada bagian ini, dua variabel diinisialisasi:

- n = len(arr) menyimpan panjang (jumlah elemen) dari list arr, yaitu 5 jika arr = [1, 2, 3, 4, 5].
- i = 0 adalah indeks awal yang digunakan untuk memulai traversal dari elemen pertama list. Nilai ini akan digunakan sebagai penghitung dalam loop while.

```
        print("Linear Traversal using while loop: ", end=" ")
```

Penjelasan :

Baris ini mencetak teks “Linear Traversal using while loop:” sebagai label sebelum hasil traversal dicetak. Parameter end=" " digunakan agar kursor tidak pindah ke baris baru

setelah mencetak label, melainkan tetap di baris yang sama untuk mencetak elemen array selanjutnya.

```
# Linear Traversal dengan while
while i < n:
    print(arr[i], end=" ")
    i += 1
```

Penjelasan :

Bagian ini adalah inti dari penelusuran (traversal) list menggunakan perulangan while:

- while i < n: artinya selama nilai i masih kurang dari panjang array (indeks valid), loop akan terus berjalan.
- print(arr[i], end=" ") mencetak elemen ke-i dari list, diikuti oleh spasi, bukan baris baru.
- i += 1 menaikkan nilai indeks i setiap kali loop dijalankan, agar traversal bergerak maju ke elemen berikutnya.

Loop ini akan mencetak elemen list satu per satu: 1 2 3 4 5.

```
print()
```

Penjelasan :

Baris ini hanya mencetak baris baru setelah traversal selesai, agar output berikutnya (jika ada) tidak berada di baris yang sama.

Praktek-7

```
# membuat array
arr = [1, 2, 3, 4, 5]
```

Penjelasan :

Di baris ini, Anda membuat sebuah **list** Python bernama arr, yang berisi lima elemen integer: 1, 2, 3, 4, dan 5. Meskipun dalam istilah umum disebut “array”, secara teknis ini adalah list Python, bukan array dari library NumPy. List adalah struktur data bawaan Python yang mendukung berbagai operasi seperti indexing, slicing, dan iterasi.

```
# mendeklarasikan nilai awal
start = 0
end = len(arr) - 1
```

Penjelasan :

- start = 0 adalah indeks awal, yang dimulai dari elemen pertama.

 end = len(arr) - 1 menghasilkan indeks terakhir dari list (yaitu 4, karena panjang arr adalah 5).
Variabel start dan end akan digunakan untuk **menukar elemen dari dua arah (kiri dan kanan)** agar menghasilkan pembalikan (reversal) list.

```
print("Reverse Traversal using while loop: ", end=" ")
```

Penjelasan ;

Baris ini mencetak teks “Reverse Traversal using while loop:” sebagai label awal, dan menggunakan end=" " agar tidak langsung pindah ke baris baru.

```
# Reverse Traversal dengan while
while start < end:

    arr[start], arr[end] = arr[end], arr[start]
    start += 1
    end -= 1
```

Penjelasan :

Inilah inti dari proses **membalik urutan elemen list**:

- while start < end: artinya perulangan akan terus berlangsung selama indeks start masih kurang dari indeks end.
- arr[start], arr[end] = arr[end], arr[start] adalah operasi **penukaran elemen**. Elemen di posisi start ditukar dengan elemen di posisi end.
- start += 1 menaikkan indeks awal agar bergerak ke kanan.
- end -= 1 menurunkan indeks akhir agar bergerak ke kiri.

Proses ini berjalan sampai start dan end bertemu di tengah, memastikan semua elemen telah ditukar posisinya, menghasilkan list yang urutannya terbalik.

```
print(arr)
```

Penjelasan :

Baris ini mencetak isi list arr setelah proses pembalikan selesai. Dengan input awal [1, 2, 3, 4, 5], hasil akhirnya adalah:

Praktek-8

```
# membuat array
arr = [12, 16, 20, 40, 50, 70]
```

Penjelasan :

Di baris ini, Anda membuat sebuah **list Python** bernama arr yang berisi enam elemen angka: [12, 16, 20, 40, 50, 70]. List ini akan menjadi target untuk operasi penyisipan (insertion) elemen baru di bagian akhir.

```
# cetak arr sebelum penyisipan
print("Array Sebelum Insertion : ", arr)
# cetak panjang array sebelum penyisipan
print("Panjang Array : ", len(arr))
```

Penjelasan :

Baris ini mencetak isi list sebelum elemen baru disisipkan. Output yang dihasilkan adalah:

Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

Hal ini penting untuk menunjukkan keadaan awal list

```
# menyisipkan array di akhir elemen menggunakan .append()
arr.append(26)
```

Penjelasan :

Kode ini mencetak panjang (jumlah elemen) list menggunakan fungsi len(). Karena list arr memiliki 6 elemen, outputnya:

Panjang Array : 6

```
# cetak arr setelah penyisipan
print("Array Setelah Insertion : ", arr)
```

Penjelasan :

Baris ini mencetak isi list setelah elemen 26 ditambahkan. List sekarang berisi:

[12, 16, 20, 40, 50, 70, 26]

Ini menunjukkan bahwa 26 berhasil disisipkan di posisi paling akhir.

```
# cetak panjang array setelah penyisipan
print("Panjang Array : ", len(arr))
```

Penjelasan :

Baris terakhir mencetak panjang list setelah elemen baru dimasukkan. Karena elemen bertambah satu, panjangnya menjadi 7:

Panjang Array : 7

Praktek-9

```
# membuat array
arr = [12, 16, 20, 40, 50, 70]
```

Penjelasan :

Pada baris ini, sebuah array (dalam Python disebut list) bernama arr dibuat dengan enam elemen integer: 12, 16, 20, 40, 50, dan 70. Ini adalah array awal sebelum dilakukan penyisipan elemen baru.

```
# cetak arr sebelum penyisipan
print("Array Sebelum Insertion : ", arr)
```

Penjelasan :

Baris ini mencetak array sebelum dilakukan penyisipan, sehingga output yang muncul adalah:

Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

```
# cetak panjang array sebelum penyisipan
print("Panjang Array : ", len(arr))
```

Penjelasan :

Di sini, fungsi len() digunakan untuk mengetahui jumlah elemen dalam array sebelum penyisipan. Karena array awal memiliki enam elemen, maka akan dicetak:

Panjang Array : 6

```
# menyisipkan array pada tengah elemen menggunakan .insert(pos, x)
arr.insert(4, 5)
```

Penjalasan :

Metode .insert(pos, x) digunakan untuk menyisipkan elemen x ke dalam list pada posisi indeks pos. Dalam hal ini, angka 5 disisipkan ke indeks ke-4. Ingat bahwa indeks

dimulai dari 0, sehingga elemen 5 akan diletakkan sebelum angka 50, menggeser elemen setelahnya satu posisi ke kanan.

Array yang awalnya:

[12, 16, 20, 40, 50, 70]

Menjadi :

[12, 16, 20, 40, 5, 50, 70]

```
# cetak arr setelah penyisipan
print("Array Setelah Insertion : ", arr)
```

Penjelasan :

Baris ini mencetak array setelah elemen 5 disisipkan. Outputnya menjadi:

Array Setelah Insertion : [12, 16, 20, 40, 5, 50, 70]

```
# cetak panjang array setelah penyisipan
print("Panjang Array : ", len(arr))
```

Penjelasan :

Karena satu elemen telah ditambahkan, panjang array sekarang bertambah menjadi 7 elemen. Fungsi len() mencetak nilai panjang terbaru dari array:

Panjang Array : 7

Praktek-10

```
# membuat array
a = [10, 20, 30, 40, 50]
print("Array Sebelum Deletion : ", a)
```

Penjelasan :

Bagian ini membuat sebuah list bernama a yang berisi lima elemen: 10, 20, 30, 40, 50. Baris print() digunakan untuk menampilkan array sebelum ada penghapusan elemen. Output:

Array Sebelum Deletion : [10, 20, 30, 40, 50]

```
# menghapus elemen array pertama yang nilainya 30
a.remove(30)
```

```
print("Setelah remove(30):", a)
```

Pemjelasan :

Metode .remove(x) digunakan untuk menghapus elemen **pertama** yang memiliki nilai **30** dari array. Setelah penghapusan ini, array berubah menjadi:

[10, 20, 40, 50]

```
# menghapus elemen array pada index 1 (20)
popped_val = a.pop(1)
print("Popped element:", popped_val)
print("Setelah pop(1):", a)
```

Penjelasan :

Metode .pop(index) menghapus dan **mengembalikan** elemen pada indeks yang ditentukan. Dalam kasus ini, elemen di indeks 1 adalah 20, sehingga 20 dihapus dan disimpan di variabel popped_val. Hasilnya:

Popped element: 20

Setelah pop(1): [10, 40, 50]

```
# Menghapus elemen pertama (10)
del a[0]
print("Setelah del a[0]:", a)
```

Penjelasan :

Kata kunci del digunakan untuk menghapus elemen pada indeks tertentu **tanpa mengembalikannya**. Di sini, elemen pertama (nilai 10) dihapus. Setelah itu, array menjadi:

[40, 50]

Praktek-11

```
# impor library numpy
import numpy as np
```

Penjelasan :

Pada baris ini, Anda mengimpor library **NumPy** dan memberinya alias **np**. NumPy adalah library populer di Python yang digunakan untuk komputasi numerik, terutama manipulasi array atau matriks. Dengan memberikan alias np, Anda bisa menulis kode lebih singkat saat memanggil fungsi-fungsi dari NumPy.

```
# membuat matiks dengan numpy
matriks_np = np.array([[1,2,3],
                       [4,5,6],
                       [7,8,9]])
```

Penjelasan :

Di baris ini, Anda membuat sebuah **matriks 3x3** menggunakan fungsi np.array(). Fungsi ini mengubah daftar bersarang (nested list) Python menjadi array NumPy. Hasilnya adalah sebuah array dua dimensi:

[[1 2 3]

 [4 5 6]

 [7 8 9]]

Array ini menyerupai matriks dalam matematika, sehingga memudahkan untuk melakukan operasi matematika seperti perkalian, penjumlahan, invers, transpos, dan sebagainya.

Praktek-12

```
X = [[12,7,3],
    [4,5,6],
    [7,8,9]]

Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]
```

Penjelasan :

Di bagian ini, dua buah **matriks 3x3** didefinisikan menggunakan struktur **list bersarang**. Matriks X dan Y masing-masing berisi 3 baris dan 3 kolom. Setiap elemen matriks direpresentasikan sebagai angka integer dalam list.

```
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
```

Penjelasan :

Di sini, Anda membuat sebuah matriks kosong result dengan ukuran yang sama (3x3), yang akan digunakan untuk menyimpan hasil penjumlahan antara elemen-elemen yang sesuai dari X dan Y. Awalnya, semua elemen dalam result diisi dengan nol.

```
# proses penjumlahan dua matriks menggunakan nested loop
# mengulang sebanyak row (baris)
for i in range(len(X)):
   # mengulang sebanyak column (kolom)
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]
```

Penjelasan :

Penjumlahan dilakukan menggunakan **dua loop bersarang**:

- Loop luar (for i in range(len(X))) mengiterasi baris-baris matriks.
- Loop dalam (for j in range(len(X[0]))) mengiterasi kolom dalam setiap baris.
  Pada setiap iterasi, elemen yang berada di posisi [i][j] dalam X dan Y dijumlahkan dan disimpan ke posisi yang sama di matriks result.

```
print("Hasil Penjumlahan Matriks dari LIST")

# cetak hasil penjumlahan secara iteratif
for r in result:
   print(r)
```

Penjelasan :

Bagian ini mencetak teks keterangan terlebih dahulu. Kemudian, dilakukan iterasi melalui setiap baris dalam result dan mencetaknya satu per satu, sehingga hasil akhir terlihat dalam bentuk matriks.

Praktek-13

```
# impor library numpy
import numpy as np
```

Penjelasan :

Baris ini digunakan untuk mengimpor library **NumPy**, salah satu pustaka paling populer di Python untuk perhitungan numerik dan manipulasi array atau matriks. np adalah alias singkat yang sering digunakan untuk memanggil fungsi-fungsi NumPy, agar penulisan kode menjadi lebih ringkas dan efisien.

```
# Membuat matriks dengan numpy
X = np.array([
   [12,7,3],
   [4,5,6],
   [7,8,9]])

Y = np.array(
   [[5,8,1],
   [6,7,3],
   [4,5,9]])
```

Penjelasan :

Di sini, dua buah **array 2D** (yang berperan sebagai matriks) dibuat menggunakan fungsi np.array(). Matriks X dan Y memiliki ukuran 3x3 dan berisi angka-angka integer. Karena menggunakan NumPy, array ini bisa langsung digunakan dalam operasi matematika tingkat lanjut tanpa perlu loop manual.

```
 Operasi penjumlahan dua matrik numpy
result = X + Y
```

Penjelasan :

Pada baris ini, dilakukan **penjumlahan elemen-elemen** dari matriks X dan Y secara langsung menggunakan operator +. Karena X dan Y adalah array NumPy dengan ukuran yang sama, operasi ini otomatis akan menjumlahkan elemen pada posisi yang sama dari kedua matriks. Ini disebut **broadcasting**, salah satu fitur unggulan NumPy yang memudahkan komputasi array.

```
# cetak hasil
print("Hasil Penjumlahan Matriks dari NumPy")
print(result)
```

Penjelasan :

Bagian ini mencetak hasil dari penjumlahan matriks dalam bentuk array 2D ke layar. Fungsi print(result) akan menampilkan hasil penjumlahan dalam format matriks yang mudah dibaca.

Prektek-14

```
# impor library numpy
import numpy as np
```

Penjelasan :

Baris ini digunakan untuk mengimpor library NumPy dengan alias np. NumPy adalah library di Python yang digunakan untuk melakukan operasi numerik, terutama yang berkaitan dengan array dan matriks, serta berbagai fungsi matematika tingkat tinggi.

```
# Membuat matriks dengan numpy
X = np.array([
    [12,7,3],
    [4,5,6],
    [7,8,9]])
```

Penjelasan :

Di sini, dibuat sebuah matriks X berukuran 3x3 menggunakan np.array. Masing-masing baris dalam array adalah sebuah list, dan secara keseluruhan membentuk sebuah array dua dimensi (matriks). Nilai-nilai dalam X adalah angka-angka bulat.

```
Y = np.array(
    [[5,8,1],
    [6,7,3],
    [4,5,9]])
```

Penjalasan :

Sama seperti sebelumnya, Y adalah matriks lain yang juga dibuat dengan np.array dan memiliki ukuran yang sama dengan X, yaitu 3x3. Ukuran yang sama ini penting agar operasi pengurangan bisa dilakukan elemen per elemen.

```
        # Operasi pengurangan dua matrik numpy
result = X - Y
```

Penjelasan :

Baris ini melakukan operasi pengurangan antara dua matriks X dan Y. Karena keduanya memiliki ukuran yang sama, maka NumPy akan secara otomatis mengurangkan elemen yang sesuai dari masing-masing posisi dalam kedua matriks. Misalnya, elemen pertama X[0][0] = 12 dikurangi dengan Y[0][0] = 5 menghasilkan 7, dan seterusnya.

```
        # cetak hasil
print("Hasil Pengurangan Matriks dari NumPy")
print(result)
```

Penjelasan :

Bagian ini mencetak teks sebagai judul hasil pengurangan, diikuti dengan hasil matriks baru result, yang merupakan hasil dari X - Y. Output yang dihasilkan akan menunjukkan matriks 3x3 yang masing-masing elemennya adalah selisih dari elemen-elemen di X dan Y.

Praktek-15

```
# impor library numpy
import numpy as np
```

Penjelasan :

Baris ini digunakan untuk mengimpor library NumPy dengan alias np. NumPy adalah library di Python yang digunakan untuk melakukan operasi numerik, terutama yang berkaitan dengan array dan matriks, serta berbagai fungsi matematika tingkat tinggi.

```
# Membuat matriks dengan numpy
X = np.array([
    [12,7,3],
    [4,5,6],
    [7,8,9]])
```

Penjelasan :

Di sini, dibuat sebuah matriks X berukuran 3x3 menggunakan np.array. Masing-masing baris dalam array adalah sebuah list, dan secara keseluruhan membentuk sebuah array dua dimensi (matriks). Nilai-nilai dalam X adalah angka-angka bulat.

```
Y = np.array(
    [[5,8,1],
    [6,7,3],
    [4,5,9]])
```

Penjelasan :

Sama seperti sebelumnya, Y adalah matriks lain yang juga dibuat dengan np.array dan memiliki ukuran yang sama dengan X, yaitu 3x3. Ukuran yang sama ini penting agar operasi pengurangan bisa dilakukan elemen per elemen.

```
# Operasi pengurangan dua matrik numpy

result = X - Y
```

Penjelasan :

Baris ini melakukan **perkalian elemen-per-elemen** antara matriks X dan Y, bukan perkalian matriks secara matematis. Ini berarti elemen yang berada pada posisi yang sama dalam kedua matriks akan dikalikan. Misalnya, elemen pertama pada baris pertama dari X (yaitu 12) dikalikan dengan elemen pertama pada baris pertama dari Y (yaitu 5), menghasilkan 60. Proses ini dilakukan untuk seluruh elemen.

Jika ingin melakukan **perkalian matriks secara matematika (dot product)**, seharusnya digunakan np.dot(X, Y) atau X @ Y.

```
# cetak hasil

print("Hasil Pengurangan Matriks dari NumPy")
print(result)
```

Penjelasan :

Bagian ini mencetak teks sebagai judul hasil pengurangan, diikuti dengan hasil matriks baru result, yang merupakan hasil dari X * Y. Output yang dihasilkan akan menunjukkan matriks 3x3 yang masing-masing elemennya adalah selisih dari elemen-elemen di X dan Y.

Praktek-16

```
# Praktek 17 : Operasi Pembagian Matriks dengan numpy
# impor library numpy
import numpy as np
```

Penjelasan :

Baris ini mengimpor library **NumPy** dan memberi alias np agar mudah digunakan. NumPy adalah library utama untuk operasi numerik dan manipulasi array di Python, termasuk operasi matriks.

```
# Membuat matriks dengan numpy
X = np.array([
    [12,7,3],
    [4,5,6],
    [7,8,9]])
```

Penjelasan :

Kode ini membuat sebuah matriks 3x3 bernama X dengan elemen-elemen yang sudah didefinisikan secara eksplisit dalam bentuk list 2 dimensi. Fungsi np.array() mengubah list ini menjadi objek array NumPy, yang memungkinkan operasi numerik lebih efisien.

```
Y = np.array(
    [[5,8,1],
    [6,7,3],
    [4,5,9]])
```

Penjelasan :

Matriks kedua Y juga dibuat dengan ukuran yang sama (3x3), berisi nilai-nilai berbeda. Tujuannya adalah untuk melakukan operasi matematika antara dua matriks ini.

```
# Operasi pembagian dua matrik numpy
result = X / Y
```

Penjelasan :

Bagian ini melakukan **pembagian elemen-per-elemen** antara matriks X dan Y. Artinya, setiap elemen pada posisi yang sama di matriks X dibagi dengan elemen pada posisi yang sama di matriks Y. Misalnya, elemen 12 di X[0,0] dibagi dengan elemen 5 di Y[0,0], menghasilkan 2.4. Hasil operasi ini juga sebuah matriks baru berukuran 3x3.

```
# cetak hasil
print("Hasil Pembagian Matriks dari NumPy")
print(result)
```

Penjelasan :

Bagian ini menampilkan teks penjelasan dan hasil operasi pembagian matriks ke layar. Isi dari variabel result, yang merupakan hasil pembagian elemen-per-elemen antara X dan Y, akan dicetak dalam bentuk matriks.

Praktek-17

```
# impor library numpy
import numpy as np
```

Penjelaskan :

Baris ini mengimpor library **NumPy** dengan alias np. NumPy sangat penting untuk operasi matematis dengan array dan matriks di Python.

```
# membuat matriks
matriks_a = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
```

Penjelasan :

Di sini kamu membuat sebuah matriks 3x3 bernama matriks_a menggunakan fungsi np.array(). Matriks ini berisi angka dari 1 sampai 9 yang disusun dalam 3 baris dan 3 kolom.

```
# cetak matriks
print("Matriks Sebelum Transpose")
print(matriks_a)
```

Penjelasan :

Baris ini menampilkan teks penjelasan dan kemudian mencetak isi matriks matriks_a ke layar. Ini menunjukkan bentuk asli matriks sebelum dilakukan operasi transpose.

```
# transpose matriks_a
balik = matriks_a.transpose()
```

Penjelasan :

Di bagian ini dilakukan operasi **transpose** pada matriks matriks_a menggunakan fungsi .transpose(). Transpose matriks berarti baris menjadi kolom dan kolom menjadi baris. Misalnya, elemen pada posisi baris 1 kolom 2 akan berpindah ke baris 2 kolom 1.

```
# cetak matriks setelah dibalik
print("Matriks Setelah Transpose")
print(balik)
```

Penjelasan :

Baris ini mencetak teks penjelasan dan hasil dari operasi transpose, yaitu matriks baru balik yang merupakan transpose dari matriks_a. Matriks ini memperlihatkan susunan elemen yang sudah ditukar antara baris dan kolom.

Praktek-18

```
# impor library numpy
import numpy as np
```

Penjelasan :

Pada baris ini, Anda mengimpor library **NumPy**, sebuah library populer di Python yang digunakan untuk komputasi numerik dan manipulasi array. Anda mengimpor library ini dengan alias np, sehingga setiap pemanggilan fungsi dari NumPy bisa disingkat menggunakan np.

```
# membuat array 1 dimensi
arr_1d = np.array([50, 70, 89, 99, 103, 35])
```

Penjelasan :

Di sini, Anda membuat sebuah **array satu dimensi** menggunakan fungsi np.array(), dengan isi elemen berupa daftar angka [50, 70, 89, 99, 103, 35]. Array ini memiliki 6 elemen.

```
# cetak matriks sebelum reshape
print("Matriks Sebelum Reshape")
print(arr_1d)
print("Ukuran Matriks : ", arr_1d.shape)
print("\n")
```

Penjelasan :

Blok ini digunakan untuk menampilkan isi array sebelum dilakukan perubahan bentuk (reshape). Fungsi arr_1d.shape menunjukkan ukuran array, yang dalam hal ini adalah (6,) — artinya array ini memiliki 6 elemen dalam satu dimensi.

```
# mengubah matriks menjadi ordo 3 x 2
ubah = arr_1d.reshape(3, 2)
```

Penjelasan :

Pada baris ini, Anda **mengubah bentuk (reshape)** array 1 dimensi menjadi array 2 dimensi berukuran **3 baris dan 2 kolom**. Karena jumlah elemen total (6) tetap sama, proses ini berhasil tanpa kehilangan data.

```
# cetak matriks setelah reshape ke ordo 3 x 2
print("Matriks Setelah Reshape")
print(ubah)
print("Ukuran Matriks : ", ubah.shape)
```

Penjelasan :

Terakhir, Anda mencetak array yang telah diubah bentuknya menjadi matriks 2 dimensi (3x2). Fungsi ubah.shape sekarang akan menampilkan (3, 2), menunjukkan bentuk matriks baru: 3 baris dan 2 kolom.