PRAKTEK-1

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

Penjelasan:

- Mengimpor library **NumPy** dan memberi alias **`np`** untuk mempersingkat penulisan.
- NumPy adalah library Python yang digunakan untuk komputasi numerik dan manipulasi array multidimensi dengan performa tinggi.

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

Penjelasan:

- Membuat array NumPy dengan nama `nilai_siswa` yang berisi empat nilai: 85, 55, 40, dan 90.
- Berbeda dengan list biasa, array NumPy memungkinkan operasi matematika lebih efisien dan lebih cepat.

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

Penjelasan:

- Mengakses elemen ke-4 dari array (`nilai_siswa[3]` karena indeks dimulai dari 0).
- Elemen ke-4 dari `[85, 55, 40, 90]` adalah **90**, sehingga yang dicetak adalah `90`.

PRAKTEK-2

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

Penjelasan:

- Mengimpor library **NumPy** dengan alias **`np`**.
- NumPy digunakan untuk membuat dan memanipulasi array secara efisien.

```
# membuat array dengan numpy
nilai_siswa_1 = np.array([75, 65, 45, 80])
```

Penjelasan:

- Membuat array 1 dimensi bernama `nilai_siswa_1` berisi 4 nilai: 75, 65, 45, dan 80.

```
nilai_siswa_2 = np.array([[85, 55, 40], [50, 40, 99]])
```

Penjelasan:

- Membuat array 2 dimensi bernama `nilai_siswa_2`, berupa matriks 2 baris × 3 kolom:

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

Penjelasan:

- Mengakses elemen pertama dari `nilai_siswa_1` (indeks 0), yaitu **75**.

```
print(nilai_siswa_2[1][1])
```

Penjelasan:

- Mengakses baris ke-2 (indeks 1), kolom ke-2 (indeks 1) dari `nilai_siswa_2`, yaitu nilai **40**.

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

Penjelasan:

- Mengubah nilai pertama `nilai_siswa_1` dari **75** menjadi **88**.

```
nilai_siswa_2[1][1] = 70
```

Penjelasan:

- Mengubah nilai baris ke-2 kolom ke-2 dari `nilai_siswa_2` dari **40** menjadi **70**.

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

Penjelasan:

- Menampilkan nilai baru dari `nilai_siswa_1[0]`, yaitu **88**.

```
print(nilai_siswa_2[1][1])
```

Penjelasan:

- Menampilkan nilai baru dari nilai_siswa_2[1][1], yaitu **70**.

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

Penjelasan:

- Menampilkan **shape** (ukuran) array `nilai_siswa_1`, hasilnya `(4,)` artinya 1 dimensi dengan 4 elemen.

```
print("Ukuran Array : ", nilai_siswa_2.shape)
```

Penjelasan:

- Menampilkan ukuran array `nilai_siswa_2`, hasilnya `(2, 3)` artinya 2 baris dan 3 kolom.

```
print("Dimensi Array : ", nilai_siswa_2.ndim)
```

Penjelasan:

- Menampilkan jumlah **dimensi** dari array `nilai_siswa_2`, hasilnya `2` karena bentuknya 2D (baris × kolom).

PRAKTEK-3

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

Penjelasan:

- Mengimpor library **NumPy** dan memberikan alias **`np`** agar mudah dipanggil.

- NumPy digunakan untuk mengelola array dan melakukan operasi matematis.

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

Penjelasan:

- Membuat dua buah array NumPy 1 dimensi:
- `a = [1, 2, 3]`
- `b = [4, 5, 6]`

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

Penjelasan:

- Menjumlahkan array `a` dan `b` secara **elemen per elemen**:
- `1+4 = 5`, `2+5 = 7`, `3+6 = 9`
- Hasil: `[5, 7, 9]`

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

Penjelasan:

- Membuat array baru bernama `arr` dengan elemen `[10, 20, 30, 40]`.

```
print(arr[1:3])    # array([20, 30])
```

Penjelasan:

- **Slicing**: Mengambil elemen dari indeks ke-1 sampai sebelum ke-3:
- `arr[1] = 20`, `arr[2] = 30`
- Hasil: `[20, 30]`

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

Penjelasan:

- Melakukan **loop** pada array `arr`, mencetak setiap elemen satu per satu:
- `x = 10`, lalu cetak 10
- `x = 20`, lalu cetak 20
- `x = 30`, lalu cetak 30
- `x = 40`, lalu cetak 40

PRAKTEK-4

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

Pejelasan:

- Membuat sebuah list (array) bernama `arr` yang berisi elemen `[1, 2, 3, 4, 5]`.

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

Penjelasan:

- Mencetak teks **"Linear Traversal:"** tanpa pindah baris karena `end=" "` membuat output tetap di baris yang sama.

```
for i in arr:
    print(i, end=" ")
```

Penjelasan:

- Melakukan **looping** (perulangan) terhadap setiap elemen di array `arr`.
- Setiap elemen `i` akan dicetak, dan `end=" "` menjaga agar elemen-elemen dicetak **berdampingan** di baris yang sama.

PRAKTEK-5

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

penjelasan :

- Membuat list bernama `arr` berisi lima elemen: `[1, 2, 3, 4, 5]`.

```
# Reverse Traversal dari elemen akhir
print("Reverse Traversal: ", end="")
```

penjelasan :

- Mencetak teks **"Reverse Traversal:"** tanpa pindah baris karena `end=""` (agar hasil traversal dicetak di baris yang sama).

```
for i in range(len(arr) - 1, -1, -1):
    print(arr[i], end=" ")
```

penjelasan :

- Perulangan dilakukan secara mundur menggunakan `range(start, stop, step)`:
- `start = len(arr) - 1` → mulai dari indeks terakhir (`4` untuk elemen `5`).
- `stop = -1` → berakhir sebelum indeks `-1`, artinya sampai indeks `0`.
- `step = -1` → melangkah mundur satu per satu.
- `arr[i]` mencetak elemen berdasarkan indeks mundur.
- `end=" "` menjaga agar hasil cetakan tetap dalam satu baris, dipisahkan spasi.

PRAKTEK-7

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

penjelasan :

- •Membuat sebuah array (list) bernama `arr` yang berisi 5 elemen: `[1, 2, 3, 4, 5]`.

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

penjelasan :

- `n` menyimpan panjang (jumlah elemen) array `arr`, yaitu `5`.
- `i` adalah variabel penghitung (counter) yang diinisialisasi dengan `0` sebagai indeks awal array.

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

penjelasan :

- Mencetak teks **"Linear Traversal using while loop:"** tanpa pindah baris karena `end=" "`.

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

penjelasan :

- Melakukan **perulangan `while`** selama `i` kurang dari `n` (yaitu selama indeks masih dalam jangkauan array).
- `arr[i]` mencetak elemen pada indeks ke-`i`.
- `end=" "` membuat hasil cetak berada dalam satu baris dan dipisahkan dengan spasi.
- `i += 1` menaikkan indeks agar berpindah ke elemen berikutnya.

PRAKTEK- 8

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

penjelasan :

- Membuat list (array) bernama `arr` dengan isi `[1, 2, 3, 4, 5]`

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

penjelasan :

- `start = 0` → inisialisasi indeks awal (elemen pertama).
- `end = len(arr) - 1 = 4` → inisialisasi indeks akhir (elemen terakhir).

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

penjelasan :

- Mencetak teks **"Reverse Traversal using while loop:"** tanpa pindah ke baris baru (`end=" "`).

```
while start < end:
```

penjelasan :

- Memulai perulangan selama indeks `start` masih **kurang dari** indeks `end`.

```
arr[start], arr[end] = arr[end], arr[start]
```

penjelasan :

- Menukar elemen di indeks `start` dan `end`.

- Misalnya: `arr[0]` tukar dengan `arr[4]`.

```
start += 1
    end -= 1
```

penjelasan :

- Setelah penukaran:
- `start` digeser ke kanan (`+1`)
- `end` digeser ke kiri (`-1`)
- Ini dilakukan agar penukaran berlanjut ke elemen berikutnya dari luar ke dalam.

PRAKTEK- 9

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

penjelasan :

- Membuat sebuah list bernama `arr` dengan elemen awal: `[12, 16, 20, 40, 50, 70]`.

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

penjelasan :

- Mencetak isi array **sebelum elemen baru disisipkan**:
- Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

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

penjelasan :

- Mencetak jumlah elemen array sebelum penyisipan:
- Panjang Array : 6

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

penjelasan :

- Menyisipkan elemen `26` **di akhir array** menggunakan metode `.append()`.

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

penjelasan :

- Mencetak isi array setelah penyisipan:
- Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]

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

penjelasan :

- Mencetak jumlah elemen array setelah penyisipan:
- Panjang Array : 7

PRAKTEK-10

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

penjelasan :

- Membuat sebuah list bernama `arr` dengan elemen awal: `[12, 16, 20, 40, 50, 70]`.

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

penjelasan :

- Mencetak isi array **sebelum elemen baru disisipkan**:
- Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

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

penjelasan :

- Mencetak jumlah elemen array sebelum penyisipan:
- Panjang Array : 6

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

penjelasan :

- Menyisipkan angka `5` pada **indeks ke-4**.
- Posisi ini berada **di antara elemen `50` dan `40`** karena indeks dimulai dari 0:

- `[12, 16, 20, 40, 50, 70]`
  → setelah insert di indeks 4:
  `[12, 16, 20, 40, 5, 50, 70]`

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

penjelasan :

- Mencetak isi array setelah penyisipan:
- Array Setelah Insertion : `[12, 16, 20, 40, 50, 70, 26]`

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

penjelasan :

- Mencetak jumlah elemen array setelah penyisipan:
- Panjang Array : 7

PRAKTEK- 11

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

penjelasan:

- Membuat list `a` berisi lima elemen: `[10, 20, 30, 40, 50]`
- Mencetak array sebelum ada penghapusan.

```
# menghapus elemen array pertama yang nilainya 30
a.remove(30)
print("Setelah remove(30):", a)
```

penjelasan:

- `a.remove(30)` akan **menghapus elemen pertama** yang **bernilai 30**.
- Hanya menghapus berdasarkan **nilai**, bukan posisi.
- Setelah ini, array 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:

- `a.pop(1)` menghapus dan **mengembalikan** nilai pada indeks ke-1, yaitu `20`.
- Nilai yang dihapus disimpan ke variabel `popped_val`.
- Setelah ini, array menjadi: `[10, 40, 50]`
- `popped_val` berisi `20`, dan dicetak juga.

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

penjelasan:

- `del a[0]` menghapus elemen di indeks ke-0 (yaitu `10`) **tanpa mengembalikan nilai**.
- Setelah ini, array menjadi: `[40, 50]`

Praktek 12

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

penjelasan:

- Mengimpor library **NumPy** dengan alias `np`, yang umum digunakan untuk operasi array, matriks, dan perhitungan numerik di Python.

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

Penjelasan:

- Membuat sebuah **array 2 dimensi** (yaitu matriks 3x3) menggunakan `np.array()`.
- Struktur data yang dihasilkan adalah **NumPy array**, bukan list biasa.

- Isi dari `matriks_np`:

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- Setiap **baris** adalah satu list di dalam list besar:
  - Baris 1: `[1, 2, 3]`
  - Baris 2: `[4, 5, 6]`
  - Baris 3: `[7, 8, 9]`

PRAKTEK-13

```
# Program penjumlahan matriks yang dibuat dari list
```

Penjelasan:

- Komentar sebagai judul program.

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

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

Penjelasan:

- `X` dan `Y` adalah dua buah **matriks 3x3** dalam bentuk **nested list**.
- Akan dilakukan penjumlahan elemen yang **berkorespondensi** dari `X` dan `Y`.

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

Penjelasan:

- Matriks hasil (`result`) diinisialisasi sebagai matriks 3x3 yang berisi nilai nol. Akan diisi hasil penjumlahan `X +`

```
# 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:

- `len(X)` = 3 → jumlah baris.
- `len(X[0])` = 3 → jumlah kolom.
- Loop luar (`for i`) untuk setiap **baris**.
- Loop dalam (`for j`) untuk setiap **kolom** di dalam baris tersebut.
- Isi dari `result[i][j]` akan menjadi hasil penjumlahan `X[i][j] + Y[i][j]`.

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

penjelasan:

- Menampilkan judul hasil penjumlahan.

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

penjelasan:

- Mencetak **setiap baris** dari matriks hasil (`result`) satu per satu.

PRAKTEK-14

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

penjelasan:

- Mengimpor library **NumPy** dengan alias `np`. NumPy digunakan untuk operasi numerik yang efisien, termasuk array dan matriks.

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

Penjelasan:

- Membuat matriks `X` berukuran **3x3** menggunakan `np.array`.
- Isi `X`:

```
[[12  7  3]
 [ 4  5  6]
 [ 7  8  9]]
```

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

Penjelasan:

- Melakukan **penjumlahan elemen per elemen** antara `X` dan `Y`.
- Ini disebut **broadcasting**, di mana operasi dilakukan langsung pada seluruh elemen yang sesuai.
- Contoh:
  - `result[0][0] = 12 + 5 = 17`
  - `result[1][2] = 6 + 3 = 9`
  - dan seterusnya.

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

Penjelasan:

- Menampilkan hasil penjumlahan matriks.

PRAKTEK-15

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

Penjelasan:

- Mengimpor library **NumPy**, yang digunakan untuk operasi array dan matriks di Python.

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

Penjelasan:

- Membuat matriks x berukuran **3 baris x 3 kolom**.
- Elemen-elemen dalam x:

```
[[12  7  3]
 [ 4  5  6]
 [ 7  8  9]]
```

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

Penjelasan:

- Membuat matriks `Y` juga berukuran **3x3**.
- Elemen-elemen dalam `Y`:

```
[[5 8 1]
 [6 7 3]
 [4 5 9]]
```

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

Penjelasan:

- Melakukan **pengurangan antar elemen** pada posisi yang sama:
    - `result[0][0] = 12 - 5 = 7`
    - `result[1][2] = 6 - 3 = 3`
    - `result[2][2] = 9 - 9 = 0`
    - dan seterusnya.
- Operasi ini dilakukan **otomatis dan efisien** oleh NumPy.

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

Penjelasan:

- Menampilkan hasil akhir dari operasi pengurangan.

PRAKTEK-16

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

Penjelasan:

• Mengimpor library **NumPy** yang sangat berguna untuk manipulasi array dan operasi matematika seperti matriks.

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

Penjelasan:

- Membuat matriks `X` berukuran **3 baris × 3 kolom** menggunakan `np.array`.

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

Penjelasan:

• Membuat matriks `Y`, juga 3×3.

```
# Operasi perkalian dua matrik numpy
result = X * Y
```

Penjelasan;

- Ini melakukan **perkalian elemen per elemen** (bukan perkalian matriks biasa!).
- Disebut juga **perkalian Hadamard**.
- Contoh hasil:
    - `result[0][0] = 12 * 5 = 60`
    - `result[0][1] = 7 * 8 = 56`
    - `result[2][2] = 9 * 9 = 81`
- **Bukan**: `dot product` atau `perkalian matriks linear algebra`.

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

Penjelasan:

• Menampilkan hasil dari perkalian elemen per elemen antara `X` dan `Y`.

PRAKTEK-17

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

Penjelasan:

• Mengimpor library **NumPy**, digunakan untuk perhitungan numerik dan manipulasi array/matriks secara efisien.

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

Penjelasan:

• Membuat matriks `X` berukuran 3 baris × 3 kolom dengan elemen-elemen:

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

Penjelasan:

• Membuat matriks `Y` berukuran sama (3×3) dengan elemen-elemen:

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

Penjelasan:

- Melakukan **pembagian elemen per elemen** (bukan pembagian matriks secara aljabar linear).
- Contoh:
  - `result[0][0] = 12 / 5 = 2.4`
  - `result[1][2] = 6 / 3 = 2.0`
  - `result[2][2] = 9 / 9 = 1.0`

• Jika elemen di `Y` ada yang **nol**, maka akan menghasilkan peringatan `RuntimeWarning` (karena pembagian dengan nol tidak didefinisikan).

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

Penjelasan:

• Mencetak hasil pembagian antara matriks `X` dan `Y`

PRAKTEK-18

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

penjelasan:

• Mengimpor library **NumPy**, yang digunakan untuk operasi array dan matriks secara efisien di Python.

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

Penjelasan:

- Membuat sebuah **matriks 3x3** menggunakan `np.array`.

- Isi matriks `matriks_a`:

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

Penjelasan:

- Menampilkan isi matriks sebelum dilakukan operasi transpose.

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

Penjelasan:

- Melakukan **transpose** terhadap `matriks_a`.
- Transpose artinya **baris menjadi kolom dan kolom menjadi baris**.
- Hasil transpose disimpan dalam variabel `balik`.
- Bentuk hasil transpose:

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

Penjelasan:

- Menampilkan hasil matriks setelah di-transpose.