

## **TUGAS STRUKTUR DATA**

Dosen Pengampu :

Adam Bachtiar S, kom, M, MT



**Disusun oleh :**

Nama : Sinta Nabila Soraya

Nim : 24241071

Kelas : B

**PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI  
FAKULTAS SAINS, TEHNIK DAN TERAPAN  
UNIVERSITAS PENDIDIKAN MANDALIKA MATARAM  
TAHUN 2025**

## PRAKTEK KE 1

The screenshot shows a Jupyter Notebook cell with the following content:

```
Array > ⚡ cobapy > ...
1 # impor library numpy
2 import numpy as np
3
4 # membuat array dengan numpy
5 nilai_siswa = np.array([85, 55, 40, 90])
6
7 # akses data pada array
8 print(nilai_siswa[3])
```

Below the code cell, the terminal output is displayed:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA - RIMA NOVA UTAMI\Python313\python.exe "c:/Users/user/OneDrive/Dokumen/modul 04/STRUKTUR DAT
90
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA - RIMA NOVA UTAMI\
```

### Baris 2

```
import numpy as np
```

Baris ini **mengimpor library** bernama numpy dan memberi alias np, sehingga Anda bisa menggunakan fungsi-fungsi NumPy dengan menulis np.nama\_fungsi().

NumPy adalah library Python yang sangat kuat untuk perhitungan numerik dan manipulasi array.

### Baris 5

```
nilai_siswa = np.array([85, 55, 40, 90])
```

Anda membuat sebuah **array NumPy satu dimensi** yang berisi data nilai-nilai siswa: 85, 55, 40, 90.

Ini berbeda dari list biasa Python. Array NumPy lebih efisien dan memiliki banyak fitur tambahan seperti operasi vektor/matriks.

### Baris 8

```
print(nilai_siswa[3])
```

Anda mencetak nilai pada indeks ke-3 dari array nilai\_siswa.

Dalam Python (dan NumPy), indeks dimulai dari **0**, sehingga:

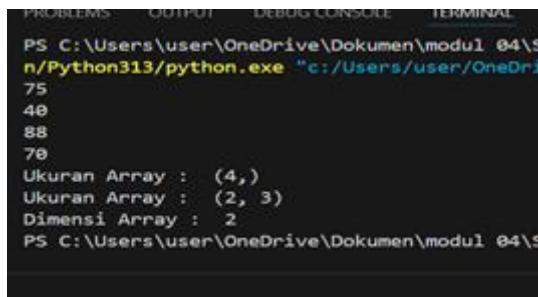
- nilai\_siswa[0] → 85
- nilai\_siswa[1] → 55
- nilai\_siswa[2] → 40
- nilai\_siswa[3] → 90 (yang dicetak)

Jadi, output dari program ini adalah:

90

## PERAKTEK KE 2

```
12
13 # membuat array dengan numpy
14 nilai_siswa_1 = np.array([75, 65, 45, 80])
15 nilai_siswa_2 = np.array([[85, 55, 40], [50, 40, 99]])
16
17 # cara akses elemen array
18 print(nilai_siswa_1[0])
19 print(nilai_siswa_2[1][1])
20
21 # mengubah nilai elemen array
22 nilai_siswa_1[0] = 88
23 nilai_siswa_2[1][1] = 70
24
25 # cek perubahannya dengan akses elemen array
26 print(nilai_siswa_1[0])
27 print(nilai_siswa_2[1][1])
28
29 # Cek ukuran dan dimensi array
30 print("Ukuran Array : ", nilai_siswa_1.shape)
31 print("Ukuran Array : ", nilai_siswa_2.shape)
32 print("Dimensi Array : ", nilai_siswa_2.ndim)
33
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\user\OneDrive\Dokumen\modul_04\Soal_Python313\python.exe "c:/Users/user/OneDrive/Dokumen/modul_04/soal.py"
75
40
88
70
Ukuran Array : (4,)
Ukuran Array : (2, 3)
Dimensi Array : 2
PS C:\Users\user\OneDrive\Dokumen\modul_04\Soal_Python313
```

### Baris 13

```
import numpy as np
```

Mengimpor library **NumPy** dengan alias np.

### Baris 14–15

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

- nilai\_siswa\_1: array **1 dimensi** dengan 4 elemen.
- nilai\_siswa\_2: array **2 dimensi (2 baris × 3 kolom)**.

### Baris 18–19: Akses elemen array

```
print(nilai_siswa_1[0])      # Output: 75
print(nilai_siswa_2[1][1])    # Output: 40


- nilai_siswa_1[0]: elemen pertama (75)
- nilai_siswa_2[1][1]: baris ke-2, kolom ke-2 → 40

```

### Baris 22–23: Ubah nilai elemen array

```
nilai_siswa_1[0] = 88
nilai_siswa_2[1][1] = 70


- Elemen pertama nilai_siswa_1 diubah dari 75 → 88
- Elemen baris ke-2 kolom ke-2 nilai_siswa_2 dari 40 → 70

```

### Baris 26–27: Cek perubahan

```
print(nilai_siswa_1[0])      # Output: 88
```

```
print(nilai_siswa_2[1][1]) # Output: 70
```

### Baris 30–32: Cek ukuran & dimensi

```
print("Ukuran Array : ", nilai_siswa_1.shape)
print("Ukuran Array : ", nilai_siswa_2.shape)
print("Dimensi Array : ", nilai_siswa_2.ndim)

• .shape: menunjukkan ukuran/tata letak array
    ○ nilai_siswa_1.shape → (4,) → array 1 dimensi dengan 4 elemen
    ○ nilai_siswa_2.shape → (2, 3) → 2 baris, 3 kolom
• .ndim: menunjukkan Jumlah dimensi
    ○ nilai_siswa_2.ndim → 2 → array 2D
```

### Ringkasan Output:

75

40

88

70

Ukuran Array : (4,)

Ukuran Array : (2, 3)

Dimensi Array : 2

### PERAKTEK KE 3

```
Array > ⚡ coba.py > ...
1   import numpy as np
2
3
4   # membuat array
5   a = np.array([1, 2, 3])
6   b = np.array([4, 5, 6])
7
8   # menggunakan operasi penjumlahan pada 2 array
9   print(a + b)      # array([5, 7, 9])
10
11  # Indexing dan Slicing pada Array
12  arr = np.array([10, 20, 30, 40])
13  print(arr[1:3])    # array([20, 30])
14
15
16  # iterasi pada array
17  for x in arr:
18      print(x)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA - R
n\Python313\python.exe "c:/Users/user/OneDrive/Dokumen/modul
[5 7 9]
[20 30]
10
20
30
40
```

### KODE PROGRAM DENGAN PENJELASAN:

Membuat dua array 1 dimensi

```
a = np.array([1, 2, 3])
```

```
b = np.array([4, 5, 6])
```

a dan b adalah array NumPy satu dimensi.

Isi array:

- a = [1, 2, 3]
- b = [4, 5, 6]

Penjumlahan dua array

```
print(a + b)      # array([5, 7, 9])
```

Ini melakukan **penjumlahan elemen per elemen** (bukan menjumlahkan semua angka).

Hitungannya:

- $1 + 4 = 5$
- $2 + 5 = 7$
- $3 + 6 = 9$

Hasil: [5, 7, 9]

Indexing dan slicing pada array

```
arr = np.array([10, 20, 30, 40])  
print(arr[1:3]) # array([20, 30])  
  
arr[1:3] artinya ambil elemen dari indeks 1 sampai sebelum 3:
```

- indeks 0 = 10
- indeks 1 = 20 ✓
- indeks 2 = 30 ✓
- indeks 3 = 40 ✗ (tidak diambil)

Hasil: [20, 30]

Iterasi (perulangan) pada array

```
for x in arr:
```

```
    print(x)
```

Ini akan mencetak **semua elemen dalam array** satu per satu:

```
10  
20  
30  
40
```

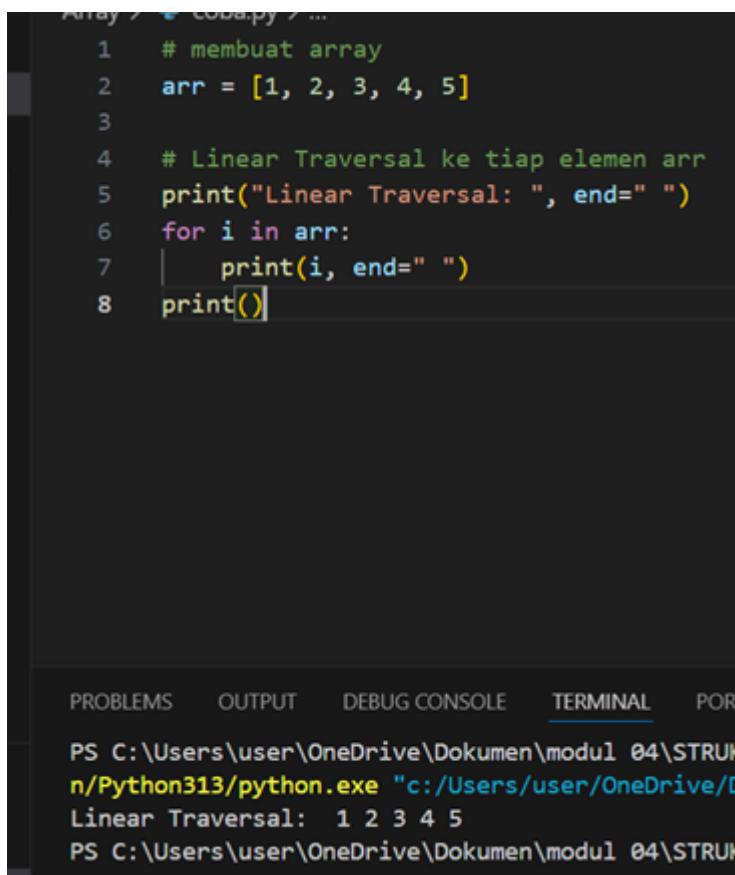
## RINGKASAN FUNGSI YANG DIPAKAI

**Fungsi / Konsep      Penjelasan**

`np.array(...)`      Membuat array dari list

a + b	Menjumlahkan elemen array per posisi
arr[1:3]	Mengambil sebagian isi array (slicing)
for x in arr:	Mengulang setiap elemen di dalam array

#### 4. PERAKTEK KE 4



```
Array > coba.py > ...
1 # membuat array
2 arr = [1, 2, 3, 4, 5]
3
4 # Linear Traversal ke tiap elemen arr
5 print("Linear Traversal: ", end=" ")
6 for i in arr:
7     print(i, end=" ")
8 print()
```

The screenshot shows a Jupyter Notebook cell with the following code:

```
1 # membuat array
2 arr = [1, 2, 3, 4, 5]
3
4 # Linear Traversal ke tiap elemen arr
5 print("Linear Traversal: ", end=" ")
6 for i in arr:
7     print(i, end=" ")
8 print()
```

Below the code cell, the terminal output is displayed:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA\Python313\python.exe "c:/Users/user/OneDrive/Dokumen/modul 04\STRUKTUR DATA\Python313\coba.py"
Linear Traversal: 1 2 3 4 5
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA\Python313\
```

1. Membuat array (dalam bentuk list biasa, bukan NumPy)

```
arr = [1, 2, 3, 4, 5]
```

arr adalah list biasa di Python (bukan array dari NumPy).

List ini berisi 5 elemen: [1, 2, 3, 4, 5]

2. Linear Traversal ke tiap elemen arr

```
print("Linear Traversal: ", end=" ")
```

Baris ini mencetak teks "Linear Traversal: " tanpa pindah baris, karena end=" " mengganti karakter akhir default \n (newline) menjadi spasi.

```
for i in arr:
```

```
    print(i, end=" ")
```

Ini adalah loop for untuk mengakses setiap elemen di dalam list arr.

- i akan bernilai 1, lalu 2, lalu 3, lalu 4, lalu 5.
- Setiap angka dicetak di baris yang sama, karena end=" " print()

Ini mencetak baris kosong untuk mengakhiri output traversal tadi, agar kursor turun ke baris baru setelah selesai.

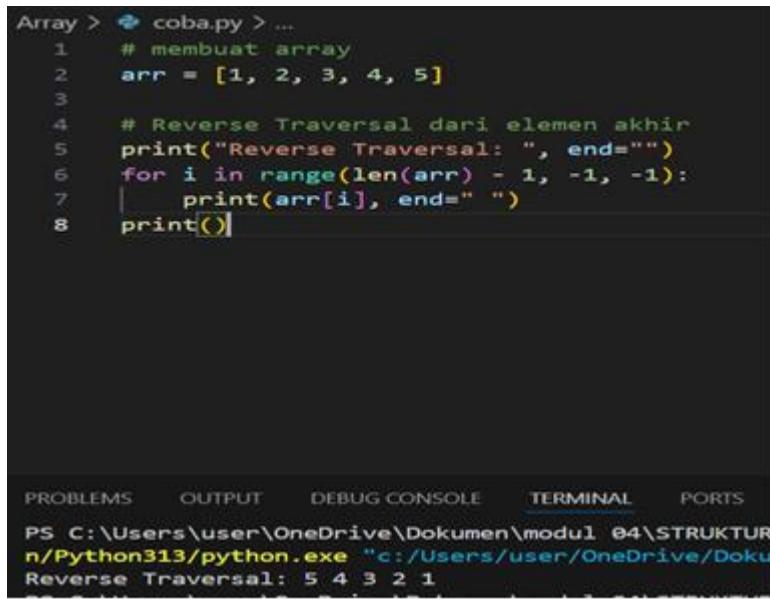
#### OUTPUT PROGRAM:

Linear Traversal: 1 2 3 4 5

#### APA ITU LINEAR TRAVERSAL?

Linear traversal adalah proses menelusuri atau mengunjungi setiap elemen dalam urutan satu per satu, dari awal sampai akhir.

## 5. PERAKTEK KE 5



```
Array > coba.py > ...
1 # membuat array
2 arr = [1, 2, 3, 4, 5]
3
4 # Reverse Traversal dari elemen akhir
5 print("Reverse Traversal: ", end="")
6 for i in range(len(arr) - 1, -1, -1):
7     print(arr[i], end=" ")
8 print()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR n\Python313\python.exe "c:/Users/user/OneDrive/Doku Reverse Traversal: 5 4 3 2 1
```

### KODE PROGRAM DAN PENJELASAN

#### 1. Membuat array (list)

```
arr = [1, 2, 3, 4, 5]
```

Kamu membuat sebuah **list** Python yang berisi angka:

```
[1, 2, 3, 4, 5]
```

#### 2. Traversal mundur (dari belakang ke depan)

```
print("Reverse Traversal: ", end="")
```

📌 Ini mencetak teks "Reverse Traversal: " tanpa pindah baris karena `end=""`.

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

```
    print(arr[i], end=" ")
```

Penjelasan bagian `range(len(arr) - 1, -1, -1)`:

- `len(arr) - 1` → posisi indeks terakhir → 4
- `-1` → batas akhir (**tidak termasuk -1**) → jadi sampai 0
- `-1` → langkah mundur

Jadi, range(4, -1, -1) menghasilkan:

4, 3, 2, 1, 0

Kemudian arr[i] mencetak elemen berdasarkan indeks itu:

- arr[4] → 5
- arr[3] → 4
- arr[2] → 3
- arr[1] → 2
- arr[0] → 1

print()

Ini untuk **pindah baris** setelah traversal selesai.

#### **OUTPUT PROGRAM:**

Reverse Traversal: 5 4 3 2 1

#### **CATATAN TAMBAHAN:**

##### **Penjelasan**

range(start,stop,step)	Membuat urutan angka dari start ke stop (tidak termasuk), dengan langkah step
len(arr)	Mengembalikan jumlah elemen dalam list
end=" "	Mencegah pindah baris setelah print, diganti dengan spasi
[Text Wrapping Break] Kalau kamu ingin versi <b>terbalik otomatis</b> tanpa for, bisa juga pakai:	
for i in reversed(arr):	
print(i, end=" ")	

#### **6. PERAKTEK KE 6**

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

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

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

C:\Users\user\OneDrive\Kuliah\modul 04\STRUKTUR DATA - RIHA NOVA  
python313/python.exe "C:/Users/user/OneDrive/Kuliah/modul 04/STRU  
near Traversal using while loop: 1 2 3 4 5  
C:\Users\user\OneDrive\Kuliah\modul 04\STRUKTUR DATA - RIHA NOVA

## KODE DAN PENJELASAN

### 1. Membuat array (list biasa)

```
arr = [1, 2, 3, 4, 5]
```

Kamu membuat list berisi 5 angka: [1, 2, 3, 4, 5]

### 2. Mendeklarasikan nilai awal

```
n = len(arr) # n akan berisi 5 (panjang list)
```

```
i = 0 # i adalah indeks awal
```

Variabel:

- n menyimpan panjang list (jumlah elemen)
- i adalah **indeks yang akan dipakai untuk menelusuri list**

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

Mencetak teks pembuka, tanpa pindah baris (karena end=" ").

### 3. Traversal menggunakan while loop

```
while i < n:
```

```
    print(arr[i], end=" ")
```

`i += 1`

Ini adalah **loop while**:

- Selama `i` kurang dari `n` (yaitu 5), program akan:
  - Cetak elemen `arr[i]`
  - Tambahkan `i` satu per satu

Urutan yang terjadi:

`i = 0 → arr[0] = 1`

`i = 1 → arr[1] = 2`

`i = 2 → arr[2] = 3`

`i = 3 → arr[3] = 4`

`i = 4 → arr[4] = 5`

Setelah `i = 5`, kondisi `i < n` menjadi salah, maka loop berhenti.

`print()`

Untuk **pindah ke baris baru** setelah traversal selesai.

#### **OUTPUT PROGRAM:**

Linear Traversal using while loop: 1 2 3 4 5

#### **PERBEDAAN DENGAN FOR LOOP**

##### **for loop**

Lebih ringkas

Cocok saat tahu jumlah pengulangan

##### **while loop**

Butuh inisialisasi dan peningkatan `i`

Cocok saat butuh kontrol lebih fleksibel

#### **7. PERAKTEK KE 7**

```
1 # membuat array
2 arr = [1, 2, 3, 4, 5]
3
4 # mendeklarasikan nilai awal
5 start = 0
6 end = len(arr) - 1
7
8 print("Reverse Traversal using while loop: ", end="")
9 # Reverse Traversal dengan while
10 while start < end:
11
12     arr[start], arr[end] = arr[end], arr[start]
13     start += 1
14     end -= 1
15 print(arr)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\user\OneDrive\Dokumen\modul\_04\STRUKTUR DATA - RIINA NOVA UTAU\Python313\python.exe "c:/Users/user/OneDrive/Dokumen/modul\_04/STRUKTUR Reverse Traversal using while loop: [5, 4, 3, 2, 1]  
PS C:\Users\user\OneDrive\Dokumen\modul\_04\STRUKTUR DATA - RIINA NOVA UTAU

## KODE PROGRAM DAN PENJELASAN

### 1. Membuat array

```
arr = [1, 2, 3, 4, 5]
```

Kamu membuat list biasa Python dengan elemen [1, 2, 3, 4, 5].

### 2. Mendeklarasikan nilai awal

```
start = 0
```

```
end = len(arr) - 1
```

Kamu menyiapkan dua indeks:

- start = 0 → indeks pertama (elemen paling kiri)
- end = 4 (karena panjang list = 5) → indeks terakhir (elemen paling kanan)

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

Mencetak teks pembuka, tanpa pindah baris (karena end=" ").

### 3. Reverse traversal menggunakan while loop

```
while start < end:
```

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

```
    start += 1
```

```
    end -= 1
```

### **Penjelasan logika:**

- Selama start < end, kamu **tukar posisi elemen kiri dan kanan**
- Lalu, start maju ke kanan dan end mundur ke kiri
- Ini disebut **in-place reverse** (membalik tanpa membuat list baru)

Langkah-langkahnya:

- Pertama: tukar arr[0] dan arr[4] → jadi [5, 2, 3, 4, 1]
- Kedua: tukar arr[1] dan arr[3] → jadi [5, 4, 3, 2, 1]
- Ketiga: start = 2, end = 2 → kondisi start < end salah → loop berhenti

```
print(arr)
```

Cetak list hasil akhir setelah dibalik: [5, 4, 3, 2, 1]

### **OUTPUT PROGRAM:**

Reverse Traversal using while loop: [5, 4, 3, 2, 1]

### **INTI LOGIKA:**

Kamu **tidak hanya menelusuri mundur**, tapi juga **membalik urutan elemen** list dengan cara:

- Menukar elemen dari ujung kiri dan ujung kanan
- Terus bergerak ke tengah

## PERAKTEK KE 8

```
1 # membuat array
2 arr = [12, 16, 20, 40, 50, 70]
3
4 # cetak arr sebelum penyisipan
5 print("Array Sebelum Insertion : ", arr)
6
7 # cetak panjang array sebelum penyisipan
8 print("Panjang Array : ", len(arr))
9
10 # menyisipkan elemen di akhir menggunakan .append()
11 arr.append(26)
12
13 # cetak arr setelah penyisipan
14 print("Array Setelah Insertion : ", arr)
15
16 # cetak panjang array setelah penyisipan
17 print("Panjang Array : ", len(arr))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL FOCUS
PS C:\Users\user\OneDrive\Documents\modul 04\STRUKTUR DATA - RIMA NOVA L
/n/Python311/python.exe "c:/Users/user/OneDrive/Documents/modul 04/STRUK
Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]
Panjang Array : 6
Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]
Panjang Array : 7
```

### KODE DAN PENJELASAN:

1. Membuat array (list)

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

Kamu membuat list arr berisi 6 elemen angka.

2. Cetak array sebelum penyisipan

```
print("Array Sebelum Insertion : ", arr)
```

Mencetak isi list sebelum elemen baru ditambahkan.

#### Output sementara:

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

3. Cetak panjang array sebelum penyisipan

```
print("Panjang Array : ", len(arr))
```

Menampilkan jumlah elemen di dalam list sebelum ditambah apa pun.

#### Output sementara:

```
Panjang Array : 6
```

4. Menyisipkan elemen di akhir menggunakan .append()

```
arr.append(26)
```

Fungsi .append() digunakan untuk **menambahkan 1 elemen di bagian akhir list**.

Setelah baris ini, arr akan menjadi:

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

5. Cetak array setelah penyisipan

```
print("Array Setelah Insertion : ", arr)
```

Menampilkan isi list setelah elemen baru (26) ditambahkan ke akhir.

**Output:**

```
Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]
```

6. Cetak panjang array setelah penyisipan

```
print("Panjang Array : ", len(arr))
```

Menampilkan jumlah elemen setelah penambahan.

**Output:**

```
Panjang Array : 7
```

**INTISARI:**

Fungsi/Perintah	Penjelasan
arr.append(x)	Menambahkan elemen x ke <b>akhir list</b>
len(arr)	Mengembalikan jumlah total elemen di dalam list
Cetak sebelum/sesudah	Berguna untuk melihat perubahan list karena operasi tertentu

**PERAKTEK KE 9**

```
1 # membuat array
2 arr = [12, 16, 20, 40, 50, 70]
3
4 # cetak arr sebelum penyisipan
5 print("Array Sebelum Insertion : ", arr)
6
7 # cetak panjang array sebelum penyisipan
8 print("Panjang Array : ", len(arr))
9
10 # menyisipkan array pada tengah elemen menggunakan .insert(pos, x)
11 arr.insert(4, 5)
12
13 # cetak arr setelah penyisipan
14 print("Array Setelah Insertion : ", arr)
15
16 # cetak panjang array setelah penyisipan
17 print("Panjang Array : ", len(arr))
```

### KODE DAN PENJELASAN:

#### 1. Membuat array (list)

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

Membuat list arr dengan 6 elemen awal.

#### 2. Cetak array sebelum penyisipan

```
print("Array Sebelum Insertion : ", arr)
```

Menampilkan isi list sebelum perubahan.

#### Output:

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

#### 3. Cetak panjang array sebelum penyisipan

```
print("Panjang Array : ", len(arr))
```

Menampilkan panjang list sebelum disisipkan elemen baru.

#### Output:

```
Panjang Array : 6
```

#### 4. Menyisipkan elemen 5 pada indeks 4 menggunakan .insert()

```
arr.insert(4, 5)
```

.insert(pos, x) menyisipkan elemen x pada indeks pos (posisi ke-4 dalam list).

- Indeks ke-4 saat ini adalah elemen 50
- Elemen baru 5 akan disisipkan di posisi ini

- Elemen di posisi 4 dan sesudahnya bergeser ke kanan

Setelah ini, arr jadi:

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

5. Cetak array setelah penyisipan

```
print("Array Setelah Insertion : ", arr)
```

Menampilkan list setelah elemen baru disisipkan.

**Output:**

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

6. Cetak panjang array setelah penyisipan

```
print("Panjang Array : ", len(arr))
```

Menampilkan panjang list setelah penambahan.

**Output:**

Panjang Array : 7

**INTISARI:**

**Fungsi/Perintah**

.insert(pos, x)

Indeks list dimulai dari 0

Elemen setelah posisi pos akan bergeser ke kanan secara otomatis

**Penjelasan**

Menyisipkan elemen x di indeks pos

Posisi ke-4 artinya elemen ke-5 dalam list

Berikut adalah penjelasan **baris per baris** dari kode Python yang Anda berikan:

**Baris**

arr = [1, 2, 3, 4, 5]

- Membuat sebuah **array/list** bernama arr dengan elemen: 1, 2, 3, 4, 5.

### Baris

```
start = 0
```

- Menginisialisasi variabel start sebagai indeks **awal** dari list, yaitu indeks pertama (0).

### Baris

```
end = len(arr) - 1
```

- Menginisialisasi variabel end sebagai indeks **akhir** dari list.
- `len(arr)` adalah panjang list (yaitu 5), sehingga `end = 5 - 1 = 4` (indeks terakhir dari array).

### Baris

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

- Mencetak teks "**Reverse Traversal using while loop:**" tanpa pindah baris (`end=" "` berarti cetak spasi, bukan newline).
- Ini hanya untuk memberi tahu bahwa proses berikutnya adalah traversal terbalik.

### Baris

```
while start < end:
```

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

```
    start += 1
```

```
    end -= 1
```

### Baris

- `while start < end:` adalah kondisi perulangan. Loop akan berjalan selama indeks start masih **lebih kecil** dari end.

### Baris

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

- Menukar elemen pada posisi start dengan end. Ini adalah cara membalik urutan elemen array **secara in-place** (langsung di dalam array, tanpa membuat array baru).

### Baris

```
start += 1
```

- Menaikkan nilai start agar mendekati ke tengah dari array.

### Baris

```
end -= 1
```

- Menurunkan nilai end agar juga mendekati tengah.

Loop ini akan terus berjalan dan menukar elemen dari luar ke dalam hingga start tidak lagi kurang dari end.

### Baris

```
print(arr)
```

- Setelah loop selesai (array sudah dibalik), baris ini mencetak isi array yang baru.

### Output

Reverse Traversal using while loop: [5, 4, 3, 2, 1]

- Elemen array arr telah **dibalik** dari [1, 2, 3, 4, 5] menjadi [5, 4, 3, 2, 1].

## PERAKTEK KE 10

```
1 #!/usr/bin/python
2 arr = [12, 16, 20, 40, 50, 70]
3
4 # cetak arr sebelum penyisipan
5 print("Array Sebelum Insertion : ", arr)
6
7 # cetak panjang array sebelum penyisipan
8 print("Panjang Array : ", len(arr))
9
10 # menyisipkan array pada tengah elemen menggunakan .insert(pos, x)
11 arr.insert(4, 5)
12
13 # cetak arr setelah penyisipan
14 print("Array Setelah Insertion : ", arr)
15
16 # cetak panjang array setelah penyisipan
17 print("Panjang Array : ", len(arr))

[IN] [OUT] [BBUP] [TERMINAL] [PLIST]

PS C:\Users\Elisan\OneDrive\Documents\modul_2> & C:/Users/Elisan/AppData/Local/Microsoft/Windows/PowerShell/2.0/tugas1.py
Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]
Panjang Array : 6
Array Setelah Insertion : [12, 16, 20, 40, 5, 50, 70]
Panjang Array : 7
```

membuat array

Komentar ini menunjukkan bahwa baris berikut akan membuat array (dalam Python disebut list).

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

Membuat sebuah list bernama arr yang berisi 6 elemen:

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

cetak arr sebelum penyisipan

Komentar bahwa baris berikut akan mencetak isi array sebelum dilakukan penyisipan.

```
print("Array Sebelum Insertion : ", arr)
```

Menampilkan isi array sebelum ditambahkan elemen:

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

cetak panjang array sebelum penyisipan

Komentar ini menjelaskan bahwa kita akan mencetak jumlah elemen dalam array sebelum penambahan.

```
print("Panjang Array : ", len(arr))
```

Menggunakan fungsi len() untuk menghitung jumlah elemen dalam array. Hasilnya adalah 6:

Panjang Array : 6

menyisipkan array pada tengah elemen menggunakan .insert(pos, x)

Komentar yang menjelaskan bahwa akan dilakukan penyisipan elemen di posisi tertentu menggunakan .insert(posisi, nilai).

```
arr.insert(4, 5)
```

Baris ini menyisipkan angka 5 ke dalam array pada indeks ke-4 (ingat: indeks dimulai dari 0).

Sebelum penyisipan:

Index: 0 1 2 3 4 5

Value: 12 16 20 40 50 70

Setelah insert(4, 5) dijalankan, angka 5 akan masuk di posisi ke-4 (sebelum angka 50), menjadi:

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

cetak arr setelah penyisipan

Komentar bahwa baris berikutnya akan mencetak isi array setelah penyisipan.

```
print("Array Setelah Insertion : ", arr)
```

Mencetak array setelah elemen 5 disisipkan:

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

cetak panjang array setelah penyisipan

Komentar bahwa kita akan menghitung ulang jumlah elemen setelah ada penyisipan.

```
print("Panjang Array : ", len(arr))
```

Mencetak panjang array setelah penambahan elemen. Karena ada satu elemen tambahan, hasilnya sekarang:

```
Panjang Array : 7
```

Kesimpulan:

- `insert(posisi, nilai)` menyisipkan elemen pada posisi tertentu tanpa menghapus elemen lain.
- Elemen-elemen setelah posisi itu akan bergeser ke kanan

`len()` digunakan untuk melihat jumlah elemen sebelum dan sesudah perubahan.

- Kalau kamu ingin, aku juga bisa tunjukkan cara menghapus elemen dari list setelah penyisipan.

## 11. PERAKTEK KE 11

```
1 # membuat array
2 a = [10, 20, 30, 40, 50]
3 print("Array Sebelum Deletion : ", a)
4
5 # menghapus elemen array pertama yang nilainya 30
6 a.remove(30)
7 print("Setelah remove(30):", a)
8
9 # menghapus elemen array pada index 1 (20)
10 popped_val = a.pop(1)
11 print("Popped element:", popped_val)
12 print("Setelah pop(1):", a)
13
14 # Menghapus elemen pertama (10)
15 del a[0]
16 print("Setelah del a[0]:", a)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\HP\OneDrive\dyni python\modul 2> & C:/Users/HP/AppData/Local/Programs/Python/3.8.5/python.exe .py  
Array Sebelum Deletion : [10, 20, 30, 40, 50]  
Setelah remove(30): [10, 20, 40, 50]  
Popped element: 30  
Setelah pop(1): [10, 40, 50]  
Setelah del a[0]: [40, 50]  
PS C:\Users\HP\OneDrive\dyni python\modul 2>

Berikut adalah penjelasan baris per baris dari kode Python yang kamu berikan:

membuat array

```
a = [10, 20, 30, 40, 50]
```

Artinya: Membuat sebuah list (array) bernama a yang berisi lima elemen: 10, 20, 30, 40, dan 50.

```
print("Array Sebelum Deletion : ", a)
```

Artinya: Menampilkan isi list a sebelum dilakukan penghapusan elemen.

- menghapus elemen array pertama yang nilainya 30

```
a.remove(30)
```

Artinya: Menghapus elemen pertama yang memiliki nilai 30 dari list. Jika ada lebih dari satu elemen dengan nilai 30, hanya yang pertama yang akan dihapus.

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

Artinya: Menampilkan isi list setelah elemen bernilai 30 dihapus.

- menghapus elemen array pada index 1 (20)

```
popped_val = a.pop(1)
```

Artinya: Menghapus elemen di indeks ke-1 (elemen ke-2) dari list, yaitu 20, dan menyimpannya ke dalam variabel popped\_val.

```
print("Popped element:", popped_val)
```

Artinya: Menampilkan elemen yang telah dihapus tadi (yaitu 20).

```
print("Setelah pop(1):", a)
```

Artinya: Menampilkan isi list setelah elemen di indeks ke-1 dihapus.

Menghapus elemen pertama (10)

```
del a[0]
```

Artinya: Menghapus elemen di indeks ke-0 (elemen pertama) dari list, yaitu 10, menggunakan kata kunci del.

```
print("Setelah del a[0]:", a)
```

Artinya: Menampilkan isi list setelah elemen pertama dihapus.

## **PERAKTEK KE 12**

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. At the top, there are tabs for 'Welcome', 'aray.py', and 'tugas1.py'. The 'tugas1.py' tab is active, displaying the following Python code:

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

Below the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the following command-line output:

```
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:/Users/elsan/Dokumen/modul 2/tugas1.py"
PS C:\Users\elsan\OneDrive\Dokumen\modul 2>
```

Berikut adalah penjelasan \*baris per baris\* dari kode Python yang kamu berikan:

```
# *Baris 1:*
```

```
python
```

```
# impor library numpy
```

- ◆ Ini adalah \*komentar\* (ditandai dengan #), artinya baris ini tidak akan dieksekusi.
- ◆ Tujuannya adalah memberi penjelasan bahwa baris berikutnya akan melakukan import library numpy.

---

#\*Baris 2:\*

```
python  
import numpy as np
```

- Ini adalah baris yang \*\*mengimpor library numpy\*\* dan memberinya alias np.
- numpy adalah library Python yang digunakan untuk komputasi numerik, terutama untuk \*mengolah array atau matriks\*.
- Dengan menulis as np, kamu bisa menggunakan np sebagai singkatan dari numpy, sehingga lebih ringkas saat memanggil fungsinya.

# \*Baris 4–7:\*

```
python  
matriks_np = np.array([[1,2,3],  
                      [4,5,6],  
                      [7,8,9]])
```

Baris ini membuat sebuah \*array dua dimensi\* (atau bisa disebut matriks) menggunakan numpy.

Fungsi np.array() digunakan untuk mengubah list (daftar) biasa menjadi array numpy.

Di dalam np.array, terdapat list 2 dimensi:

- \* Baris pertama: [1, 2, 3]
  - \* Baris kedua: [4, 5, 6]
  - \* Baris ketiga: [7, 8, 9]
- ◆ Hasilnya adalah matriks berukuran \*3x3\*.

\*Kesimpulan:

Kode ini membuat sebuah \*matriks 3x3\* dengan numpy, isinya:

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

### **PERAKTEK KE 13**

```
4 # Membuat matriks dengan numpy
5 X = np.array([
6     [12,7,3],
7     [4,5,6],
8     [7,8,9]])
9
10 Y = np.array([
11     [[5,8,1],
12     [6,7,3],
13     [4,5,9]]])
14
15 # Operasi penjumlahan dua matrik numpy
16 result = X + Y
17
18 # cetak hasil
19 print("Hasil Penjumlahan Matriks dari NumPy")
20 print(result)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA - RIMA NOVA UTAMI
1/Programs/Python/Python313/python.exe "c:/Users/user/OneDrive/Dokumen/modul_2/Array/coba.py"
Hasil Penjumlahan Matriks dari NumPy
[[17 15  4]
 [10 12  9]
 [11 13 18]]
PS C:\Users\user\OneDrive\Dokumen\modul 04\STRUKTUR DATA - RIMA NOVA UTAMI
```

Berikut adalah penjelasan baris perbaris dari kode python tersebut

#Program penjumlahan matriks yang dibuat dari list

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

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

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

```
# 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]

print("Hasil Penjumlahan Matriks dari LIST")

# cetak hasil penjumlahan secara iteratif

for r in result:

    print(r)

```

Berikut adalah penjelasan baris per baris dari kode Python untuk penjumlahan matriks yang dibuat dari list:

```

python

# Program penjumlahan matriks yang dibuat dari list

```

- Komentar yang menjelaskan tujuan program, yaitu menjumlahkan dua matriks yang direpresentasikan sebagai list di Python.

```

python

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

```

- Mendefinisikan matriks X sebagai list dua dimensi (list of lists) dengan 3 baris dan 3 kolom.

```

python

Y = [[5,8,1],
      [6,7,3],

```

[4,5,9]]

- Mendefinisikan matriks Y juga sebagai list dua dimensi dengan ukuran yang sama seperti X.

python

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

- Membuat matriks result dengan ukuran 3x3 yang diinisialisasi dengan nol sebagai tempat penyimpanan hasil penjumlahan.

python

```
# proses penjumlahan dua matriks menggunakan nested loop  
# mengulang sebanyak row (baris)  
for i in range(len(X)):
```

- Loop pertama (i) berjalan dari 0 sampai jumlah baris matriks X (3 baris). Ini mengontrol iterasi per baris.

python

```
# mengulang sebanyak column (kolom)  
for j in range(len(X[0])):
```

- Loop kedua (j) berjalan dari 0 sampai jumlah kolom matriks X (3 kolom). Ini mengontrol iterasi per kolom dalam setiap baris.

python

```
result[i][j] = X[i][j] + Y[i][j]
```

- Menjumlahkan elemen pada posisi [i][j] dari matriks X dan Y, lalu menyimpan hasilnya di posisi yang sama pada matriks result.

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

- Mencetak teks sebagai judul hasil penjumlahan matriks.

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

- Loop untuk mencetak setiap baris dari matriks result satu per satu, sehingga hasil penjumlahan ditampilkan dalam format matriks.

#### # Ringkasan

Kode ini membuat dua matriks 3x3, menjumlahkan elemen-elemen yang bersesuaian dari kedua matriks tersebut menggunakan nested loop, menyimpan hasilnya di matriks baru, dan mencetak hasilnya baris per baris.

#### **PERAKTEK 14**

```
❷ tugas1.py > ...
1   # impor library numpy
2   import numpy as np
3
4   # Membuat matriks dengan numpy
5   X = np.array([
6       [12,7,3],
7       [4,5,6],
8       [7,8,9]])
9
10  Y = np.array(
11      [[5,8,1],
12      [6,7,3],
13      [4,5,9]])
14
15  # Operasi penjumlahan dua matrik numpy
16  result = X + Y
17
18  # catatan hasil
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    **TERMINAL**    PORT

```
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:
umen/modul 2/tugas1.py"
Hasil Penjumlahan Matriks dari NumPy
[[17 15  4]
 [10 12  9]
 [11 13 18]]
```

Berikut adalah penjelasan baris per baris dari kode Python tersebut:

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

Penjelasan:

Mengimpor library NumPy dan memberinya alias np. NumPy adalah library Python yang digunakan untuk operasi matematika dan manipulasi array/matriks.

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

Penjelasan:

Membuat array 2 dimensi (matriks) bernama X menggunakan fungsi np.array.

Matriks X berisi:

12 7 3

4 5 6

7 8 9

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

Penjelasan:

Membuat matriks kedua bernama Y, juga menggunakan np.array.

Matriks Y berisi:

5 8 1

6 7 3

4 5 9

```
# Operasi penjumlahan dua matrik numpy
```

```
result = X + Y
```

Penjelasan:

Melakukan operasi penjumlahan matriks antara X dan Y. NumPy secara otomatis menjumlahkan elemen yang berada di posisi yang sama.

Contohnya:

Baris 1 kolom 1:  $12 + 5 = 17$

Baris 2 kolom 2:  $5 + 7 = 12$

Baris 3 kolom 3:  $9 + 9 = 18$

Hasilnya disimpan dalam variabel result.

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

Penjelasan:

Mencetak teks informasi, lalu mencetak isi dari matriks result, yaitu hasil penjumlahan dari X dan Y.

Output program:

Hasil Penjumlahan Matriks dari NumPy

```
[[17 15  4]
 [10 12  9]
 [11 13 18]]
```

## PERAKTEK KE 15

The screenshot shows a Jupyter Notebook cell with the following content:

```
# tugas1.py > ...
1 # impor library numpy
2 import numpy as np
3
4 # Membuat matriks dengan numpy
5 X = np.array([
6     [12,7,3],
7     [4,5,6],
8     [7,8,9]])
9
10 Y = np.array(
11     [[5,8,1],
12     [6,7,3],
13     [4,5,9]])
14
15 # Operasi pengurangan dua matrik numpy
16 result = X - Y
17
18 # catat hasil
```

Below the code cell, the Jupyter interface shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTFOLIO. The TERMINAL tab is active.

The terminal output is as follows:

```
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:\Dokumen\modul 2\tugas1.py"
Hasil Pengurangan Matriks dari NumPy
[[ 7 -1  2]
 [-2 -2  3]
 [ 3  3  0]]
PS C:\Users\elsan\OneDrive\Dokumen\modul 2>
```

Berikut adalah penjelasan \*baris per baris\* dari kode Python tersebut yang menggunakan \*NumPy\* untuk melakukan \*pengurangan dua matriks\*:

```
# Baris 1  
python  
# impor library numpy
```

> Ini adalah komentar yang menjelaskan bahwa baris berikutnya akan mengimpor library \*NumPy\*, sebuah library populer di Python untuk komputasi numerik, terutama operasi matriks dan array.

```
# Baris 2
```

```
python  
import numpy as np  
> Mengimpor library *NumPy* dan memberi alias np agar lebih ringkas saat digunakan dalam kode.
```

```
# Baris 5–8
```

```
python  
X = np.array([  
    [12,7,3],  
    [4,5,6],  
    [7,8,9]])
```

> Membuat \*matriks (array 2 dimensi)\* X menggunakan fungsi np.array. Matriks ini berukuran \*3x3\* dengan nilai-nilai sebagai berikut:

```
[12, 7, 3]  
[ 4, 5, 6]
```

[ 7, 8, 9]

# Baris 10–13

python

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

> Membuat \*matriks kedua\* Y, juga berukuran 3x3, dengan nilai:

[5, 8, 1]

[6, 7, 3]

[4, 5, 9]

# Baris 15

python

```
result = X - Y
```

> Melakukan \*pengurangan elemen-elemen dari dua matriks\* (element-wise subtraction). Setiap elemen pada posisi yang sama di X dan Y akan dikurangkan:

- $12 - 5 = 7$
- $7 - 8 = -1$
- $3 - 1 = 2$

- dan seterusnya...

Hasilnya adalah matriks result:

```
[ 7, -1, 2]
```

```
[-2, -2, 3]
```

```
[ 3, 3, 0]
```

```
# Baris 18
```

```
python
```

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

> Menampilkan teks judul agar hasil yang dicetak lebih mudah dipahami.

```
# Baris 19
```

```
python
```

```
print(result)
```

> Menampilkan hasil pengurangan matriks yang telah disimpan dalam variabel result.

```
# Kesimpulan
```

Kode ini menunjukkan \*cara menggunakan NumPy untuk membuat dua matriks dan mengurangkannya secara langsung\*. Ini jauh lebih efisien daripada menggunakan nested loop seperti pada Python standar.

### PERAKTEK KE 16

The screenshot shows a Jupyter Notebook cell with the following code:

```
# tugas1.py > ...
1 # impor library numpy
2 import numpy as np
3
4 # Membuat matriks dengan numpy
5 X = np.array([
6     [12,7,3],
7     [4,5,6],
8     [7,8,9]])
9
10 Y = np.array(
11     [[5,8,1],
12     [6,7,3],
13     [4,5,9]])
14
15 # Operasi perkalian dua matrik numpy
16 result = X * Y
17
18 # result hasil
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PC. The TERMINAL tab is selected, showing the command-line output:

```
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:\Users\elsan\OneDrive\Dokumen\modul 2\tugas1.py"
Hasil Perkalian Matriks dari NumPy
[[60 56  3]
 [24 35 18]
 [28 40 81]]
```

Berikut adalah \*penjelasan baris per baris\* dari kode Python yang menggunakan \*NumPy\* untuk melakukan \*perkalian dua matriks\* secara \*element-wise (per elemen)\*:

```
# Baris 1

python
# impor library numpy
```

> Komentar yang menjelaskan bahwa baris selanjutnya akan mengimpor library \*NumPy\*, yang digunakan untuk operasi numerik di Python.

```
# Baris 2
```

```
python
import numpy as np
```

> Mengimpor \*NumPy\* dan memberi alias np agar lebih singkat saat digunakan dalam kode.

```
# Baris 5–8
```

```
python
X = np.array([
    [12,7,3],
    [4,5,6],
    [7,8,9]])
```

> Membuat \*matriks (array 2 dimensi)\* X menggunakan np.array. Matriks ini berukuran \*3x3\* dengan elemen:

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

```
# Baris 10–13
```

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

> Membuat \*matriks kedua\* Y, juga berukuran \*3x3\*, dengan elemen:

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

# Baris 15

```
python  
result = X * Y
```

> Melakukan \*perkalian elemen-per-elemen (element-wise multiplication)\* antara matriks X dan Y. Ini \*bukan perkalian matriks biasa (dot product)\*, tetapi setiap elemen dikalikan dengan elemen pada posisi yang sama:

```
* 12 * 5 = 60  
* 7 * 8 = 56  
* 3 * 1 = 3  
* dan seterusnya...
```

Hasilnya:

[60, 56, 3]

[24, 35, 18]

[28, 40, 81]

# Baris 18

```
python
```

```
print("Hasil Perkalian Matriks dari NumPy")
```

> Menampilkan teks judul untuk memberikan konteks pada output.

# Baris 19

```
python
```

```
print(result)
```

> Menampilkan hasil perkalian elemen-per-elemen dari matriks X dan Y.

# Kesimpulan:

Kode ini memperlihatkan \*perkalian dua matriks secara element-wise\* menggunakan \* dalam NumPy. Jika kamu ingin melakukan \*perkalian matriks sesungguhnya (dot product)\*, kamu harus menggunakan:

python

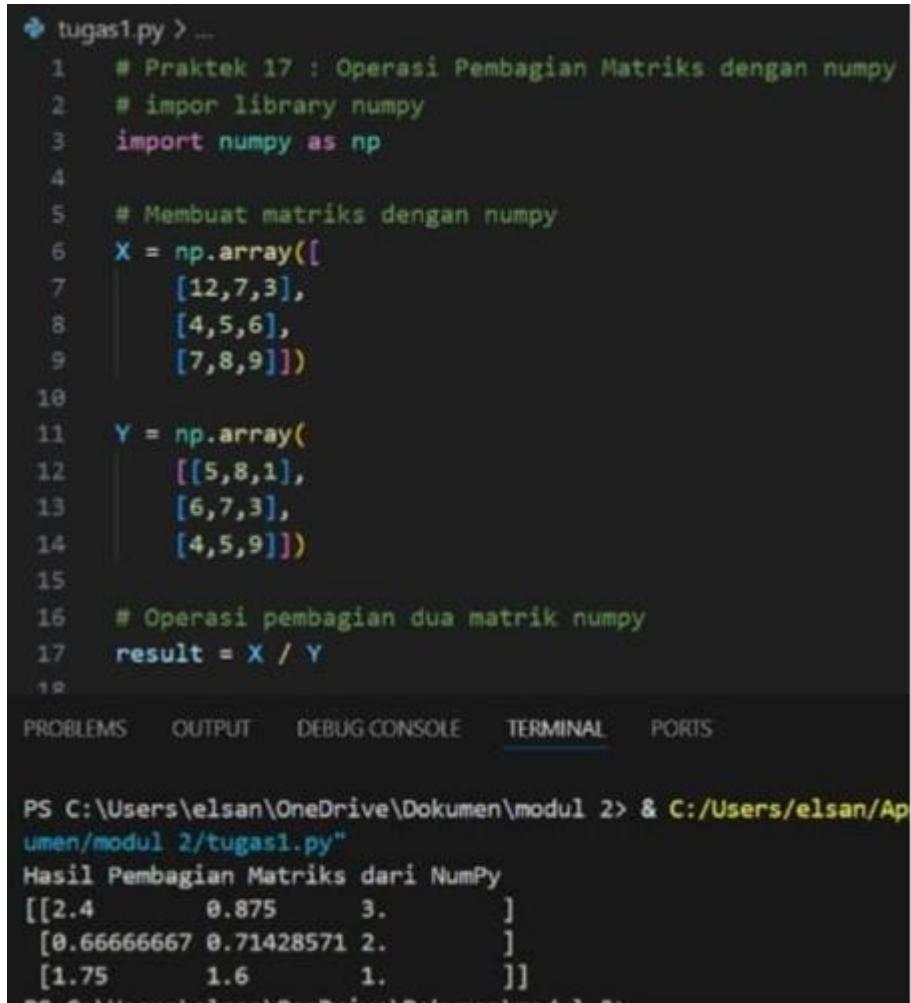
```
result = np.dot(X, Y)
```

atau

python

```
result = X @ Y
```

### PERAKTEK KE 17



```
# tugas1.py > ...
# Praktek 17 : Operasi Pembagian Matriks dengan numpy
# impor library numpy
import numpy as np

# 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]])

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

The screenshot shows a Jupyter Notebook cell with the following code:

```
# tugas1.py > ...
# Praktek 17 : Operasi Pembagian Matriks dengan numpy
# impor library numpy
import numpy as np

# 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]])

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

Below the code cell, the terminal output is displayed:

```
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:/Users/elsan/Ap
umen/modul 2/tugas1.py"
Hasil Pembagian Matriks dari NumPy
[[2.4          0.875         3.          ]
 [0.666666667 0.714285712.          ]
 [1.75          1.6           1.          ]]
```

Berikut penjelasan baris per baris dari kode Python yang kamu berikan:

```
# Praktek 17 : Operasi Pembagian Matriks dengan numpy
```

Komentar ini memberikan informasi bahwa ini adalah praktik ke-17 dan berisi contoh operasi pembagian matriks menggunakan library NumPy.

```
# impor library numpy
```

Komentar yang menjelaskan bahwa kita akan mengimpor library NumPy.

```
# import numpy as np
```

Baris ini mengimpor library NumPy dan memberi alias np, sehingga kita bisa menggunakan np untuk memanggil fungsi-fungsi dalam NumPy.

```
python
```

```
X = np.array([
    [12, 7, 3],
    [4, 5, 6],
    [7, 8, 9]
])
```

Baris ini membuat matriks 3x3 bernama X dari list Python menggunakan fungsi np.array(). Matriks X:

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

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

Membuat matriks 3x3 bernama Y yang juga berasal dari list Python. Matriks Y:

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

```
# Operasi pembagian dua matrik numpy
```

Komentar ini menjelaskan bahwa operasi selanjutnya adalah pembagian dua matriks.

```
result = X / Y
```

Baris ini melakukan pembagian elemen per elemen (element-wise division) antara matriks X dan Y. Artinya:

```
python
result[i][j] = X[i][j] / Y[i][j]
```

Contoh:

```
* result[0][0] = 12 / 5 = 2.4  
* result[0][1] = 7 / 8 = 0.875  
* dan seterusnya...
```

```
# cetak hasil
```

Komentar bahwa baris berikut akan mencetak hasil ke layar.

```
python  
print("Hasil Pembagian Matriks dari NumPy")  
print(result)
```

```
* print("Hasil Pembagian Matriks dari NumPy") mencetak judul output.  
* print(result) mencetak hasil dari pembagian matriks X dan Y dalam bentuk matriks  
3x3.
```

```
# Contoh Output:
```

Jika dijalankan, akan muncul hasil seperti ini (dibulatkan untuk tampilan):

Hasil Pembagian Matriks dari NumPy

```
[[2.4    0.875   3.     ]  
 [0.66666667 0.71428571 2.     ]  
 [1.75    1.6     1.     ]]
```

## PERAKTEK 18

The screenshot shows a Jupyter Notebook cell with the following content:

```
# tugas1.py > ...
1 # impor library numpy
2 import numpy as np
3
4 # membuat matriks
5 matriks_a = np.array([
6     [1, 2, 3],
7     [4, 5, 6],
8     [7, 8, 9]
9 ])
10
11 # cetak matriks
12 print("Matriks Sebelum Transpose")
13 print(matriks_a)
14
15 # transpose matriks_a
16 balik = matriks_a.transpose()
17
18 # cetak matriks setelah di transpose
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected.

```
umen/modul_2/tugas1.py"
Matriks Sebelum Transpose
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Matriks Setelah Transpose
[[1 4 7]
 [2 5 8]
 [3 6 9]]
PS C:\Users\elsan\OneDrive\Dokumen\modul_2>
```

Berikut adalah penjelasan baris per baris dari kode Python yang menggunakan NumPy untuk melakukan transpose (permutasi baris dan kolom) pada matriks:

```
# Baris 1
```

```
python
```

```
# impor library numpy
```

> Komentar yang menjelaskan bahwa kode akan menggunakan library NumPy.# Baris 2

```
python
```

```
import numpy as np
```

> Mengimpor library NumPy dan memberi alias np untuk mempersingkat penulisan fungsi-fungsinya.

```
# Baris 5–9
```

```
python
```

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

> Membuat matriks 2 dimensi matriks\_a menggunakan np.array. Matriks ini memiliki ukuran 3x3, dengan elemen:

```
[1, 2, 3]
```

```
[4, 5, 6]
```

```
[7, 8, 9]
```

```
# Baris 12
```

```
python
print("Matriks Sebelum Transpose")
```

> Menampilkan teks untuk memberi tahu bahwa output berikut adalah matriks sebelum dilakukan operasi transpose.

# Baris 13

```
python
print(matriks_a)
```

> Menampilkan isi dari matriks\_a.

# Baris 16

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

> Melakukan transpose, yaitu \*\*menukar baris menjadi kolom dan kolom menjadi baris.

> Hasil transpose dari matriks\_a adalah:

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

Matriks ini disimpan dalam variabel balik.

Alternatif penulisan transpose:

```
python  
balik = matriks_a.T
```

# Baris 19

```
python  
print("Matriks Setelah Transpose")
```

> Menampilkan teks penjelasan bahwa output berikut adalah matriks hasil transpose.

# Baris 20

```
python  
print(balik)
```

> Menampilkan hasil dari operasi transpose yang sudah disimpan dalam variabel balik.

# Kesimpulan:

Kode ini memperlihatkan bagaimana menggunakan NumPy untuk:

- \* Membuat matriks 2 dimensi
- \* Melihat isi matriks sebelum dan sesudah di-transpose

Transpose sangat penting dalam aljabar linear, seperti dalam operasi dot product, rotasi, atau manipulasi data tabular.