**TUGAS STRUKTUR DATA SMESTER II**

**Dosen pengampu:**

**ADAM BACHTIAR, S.kom.,M.MT**

****

OLEH:

RIDO JUANA (24241036)

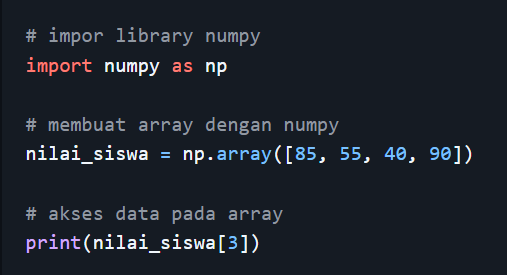
**PENDIDIKAN TEKNOLOGI INFORMASI**

**FAKULTAS TEHNIK**

**UNIVERSITAS PENDIDIKAN MANDALIKA**

**2025**

PERAKTEK 1



HASIL

90

PENJELASAN

Baris 1# impor library numpy

Komentar ini menjelaskan bahwa baris berikutnya akan digunakan untuk mengimpor library numpy, yang merupakan library Python untuk operasi numerik dan array.

Baris 2 = mengimpor library numpy dan memberi alias np. Dengan cara ini, kamu bisa menggunakan np sebagai singkatan untuk memanggil fungsi-fungsi dari numpy.

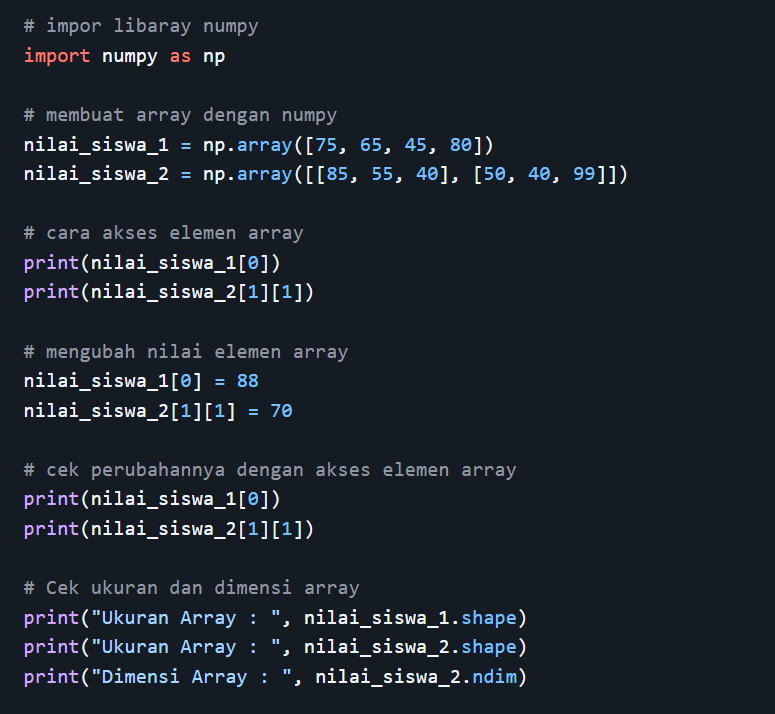
Baris 3 =Komentar ini menjelaskan bahwa baris selanjutnya akan digunakan untuk membuat array menggunakan numpy.

Baris 4 = membuat array numpy berisi empat nilai: 85, 55, 40, dan 90, lalu menyimpannya dalam variabel nilai\_siswa.

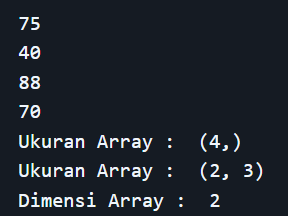
Baris 5 = Komentar ini memberi tahu bahwa baris berikut akan mengakses (mengambil) salah satu data dari array.

Baris 6 = mencetak nilai pada indeks ke-3 dari array nilai\_siswa. Karena indeks dimulai dari 0, maka nilai\_siswa[3] adalah 90.

PERAKTEK 2



HAISL



PENJELASAN

Baris 1 = Menampilkan jumlah dimensi dari nilai\_siswa\_2. Output: 2 karena array tersebut adalah 2 dimensi (baris dan kolom).

Mengimpor library NumPy dan memberi alias np. Ini membuat kita bisa menulis np.array daripada numpy.array. NumPy adalah library Python untuk operasi matematika dan manipulasi array.

Baris 2 = Membuat array 1 dimensi (vektor) dengan 4 nilai: 75, 65, 45, 80. Disimpan dalam variabel nilai\_siswa\_1

Baris 3 = Membuat array 2 dimensi (matriks) berisi 2 baris dan 3 kolom. Disimpan di variabel nilai\_siswa\_2.

Baris 4 = Mencetak elemen pertama dari nilai\_siswa\_1. Indeks 0 menunjukkan posisi pertama dalam array. Output: 75.

Baris 5 = Mengakses dan mencetak nilai pada baris ke-2 dan kolom ke-2 dari nilai\_siswa\_2. (ingat: indeks mulai dari 0). Output: 40.

Baris 6 = Mengubah nilai elemen pertama (indeks 0) pada nilai\_siswa\_1 dari 75 menjadi 88.

Baris 7 = Mengubah nilai baris ke-2, kolom ke-2 (indeks [1][1]) pada nilai\_siswa\_2 dari 40 menjadi 70.

Baris 8 = Mencetak ulang elemen pertama nilai\_siswa\_1 setelah diubah. Output: 88.

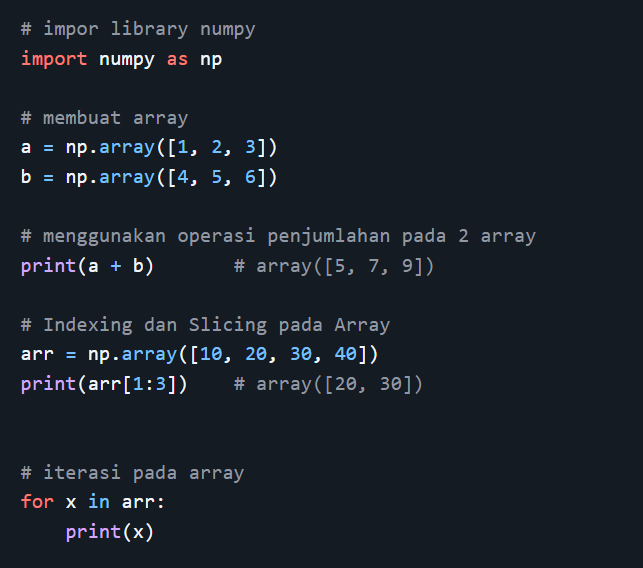
Baris 9 = Mencetak ulang nilai pada indeks [1][1] dari nilai\_siswa\_2 setelah diubah. Output: 70.

Baris 10 = Menampilkan ukuran atau bentuk array nilai\_siswa\_1. Output: (4,) yang artinya array 1 dimensi dengan 4 elemen.

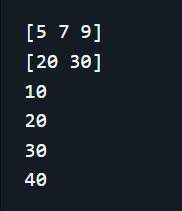
Baris 11 = Menampilkan ukuran array nilai\_siswa\_2. Output: (2, 3) artinya 2 baris dan 3 kolom.

Baris 12 = Menampilkan jumlah dimensi dari nilai\_siswa\_2. Output: 2 karena array tersebut adalah 2 dimensi (baris dan kolom).

PRAKTEK 3



HASIL



PENJELASAN

Baris 1 = Mengimpor library NumPy dan memberi alias np, agar lebih singkat saat digunakan.

Baris 2 = Membuat array 1 dimensi dan menyimpannya ke variabel a.

Baris 3 = Membuat array 1 dimensi dan menyimpannya ke variabel b.

Baris 4 = Menambahkan dua array secara elemen per elemen (penjumlahan vektor).

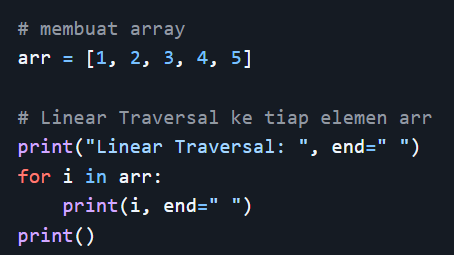
Baris 5 = Membuat array 1 dimensi dan menyimpannya ke variabel arr.

Baris 6 = Mengambil bagian dari array dari indeks 1 hingga sebelum indeks 3 (slicing).

Baris 7 = Memulai perulangan untuk setiap elemen dalam array arr.

Baris 8 = Mencetak setiap elemen dalam array satu per satu selama iterasi berlangsung.

PRAKTEK 4



HASIL



PENJELASAN

Baris 1 = Membuat list (array dalam konteks Python biasa) yang berisi lima elemen: 1, 2, 3, 4, 5. Disimpan dalam variabel arr.

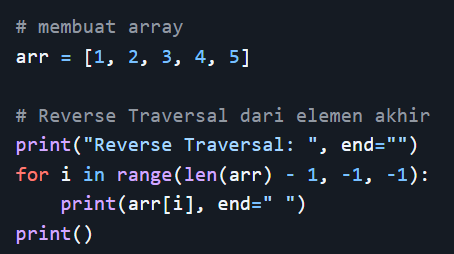
Baris 2 = Mencetak teks "Linear Traversal: " ke layar **tanpa pindah baris**, karena end=" " memberi spasi di akhir, bukan baris baru.

Baris 3 = Memulai perulangan (looping) **linear traversal**, yaitu mengakses setiap elemen dalam arr satu per satu.

Baris 4 = Mencetak elemen saat ini (i) **di baris yang sama**, dengan spasi setelahnya. Ini membuat semua angka tercetak berurutan dalam satu baris.

Baris 5 = Mencetak baris kosong untuk **pindah ke baris baru** setelah loop selesai, agar hasil lebih rapi.

PRAKTEK 5



HASIL



PENJELASAN

Baris 1 = Membuat list (array) bernama arr yang berisi lima elemen: 1, 2, 3, 4, 5.

Baris 2 = Mencetak teks "Reverse Traversal: " tanpa pindah baris karena end="" mencegah baris baru setelah teks.

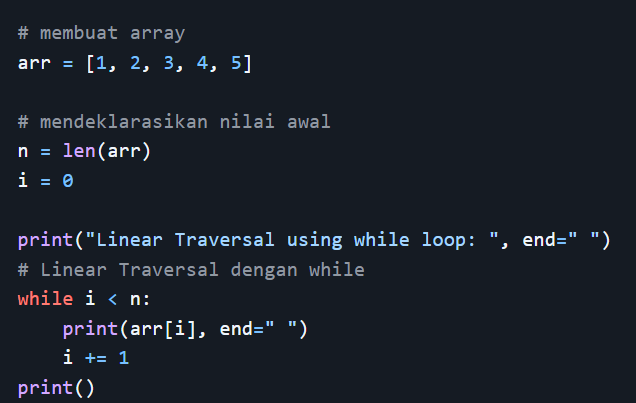
Baris 3 = Melakukan perulangan **secara terbalik** dari indeks terakhir ke indeks pertama.

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

Baris 4 = Mencetak elemen pada indeks i dari arr, diikuti spasi, **tanpa pindah baris**.

Baris 5 = Pindah ke baris baru setelah selesai mencetak semua elemen secara terbalik.

PRAKTEK 7



HASIL



PENJELASAN

Baris 1 = Membuat list arr yang berisi elemen-elemen: 1, 2, 3, 4, 5.

Baris 2 = Menghitung panjang (jumlah elemen) dari arr, hasilnya disimpan di variabel n (nilai n akan menjadi 5).

Baris 3 = Inisialisasi variabel i sebagai penghitung indeks, dimulai dari 0.

Baris 4 = Mencetak teks pembuka tanpa pindah ke baris baru karena end=" ".

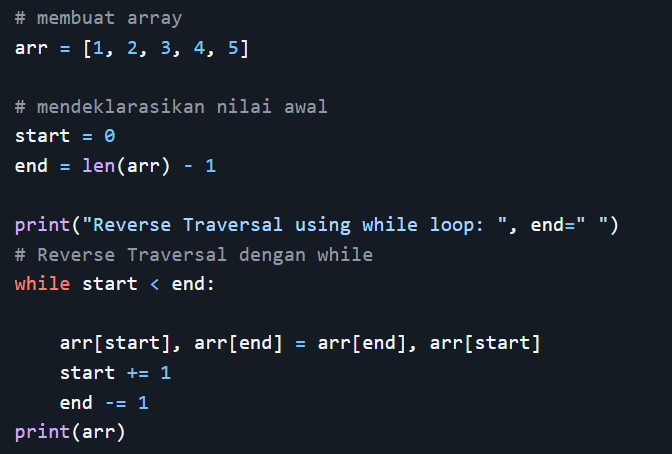
Baris 5 = Memulai loop while yang akan berjalan selama i kurang dari n (selama masih dalam rentang indeks array).

Baris 6 = Mencetak elemen arr pada indeks i di baris yang sama, diikuti spasi.

Baris 7 = Menambahkan 1 ke nilai i agar loop bergerak ke indeks berikutnya.

Baris 8 = Mencetak baris baru setelah loop selesai untuk merapikan output.

PRAKTEK 8



HAISL



PENJELASAN

Baris 1 = Membuat list arr yang berisi lima elemen: [1, 2, 3, 4, 5].

Baris 2 = Inisialisasi variabel start sebagai indeks awal (posisi kiri dari array).

Baris 3 = Inisialisasi variabel end sebagai indeks terakhir array (posisi kanan).  
len(arr) - 1 = 5 - 1 = 4.

Baris 4 = Mencetak teks penjelasan **tanpa pindah baris**, karena end=" ".

Baris 5 = Loop while akan berjalan selama indeks start masih lebih kecil dari end.  
Tujuannya adalah menukar elemen dari kedua ujung array menuju tengah.

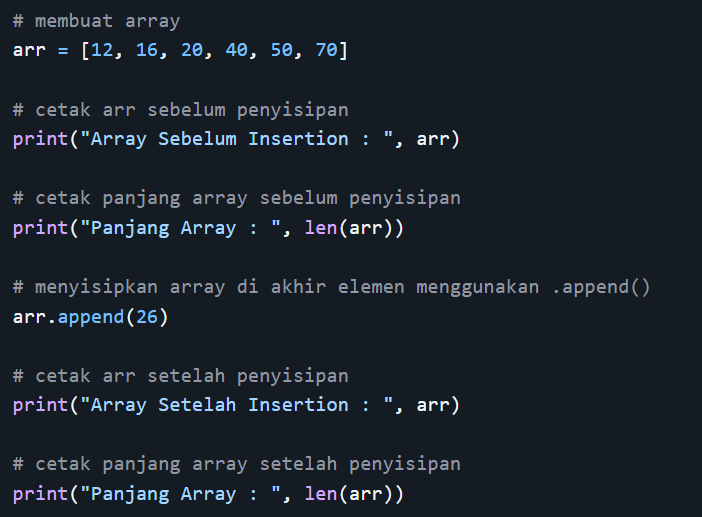
Baris 6 = Menukar elemen di posisi start dengan elemen di posisi end.  
Contoh pertama: arr[0] ditukar dengan arr[4].

Baris 7 = Memajukan indeks start ke kanan satu langkah.

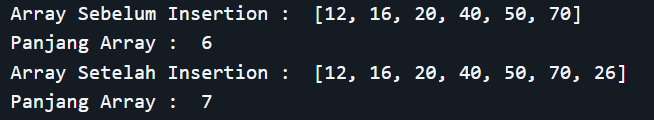
Baris 8 = Memundurkan indeks end ke kiri satu langkah.

Baris 9 = Memundurkan indeks end ke kiri satu langkah.

PRAKTEK 9



HASIL



PENJELASAN

Baris 1 = Membuat list arr berisi 6 elemen angka.

Baris 2 = Mencetak isi array sebelum elemen baru ditambahkan.

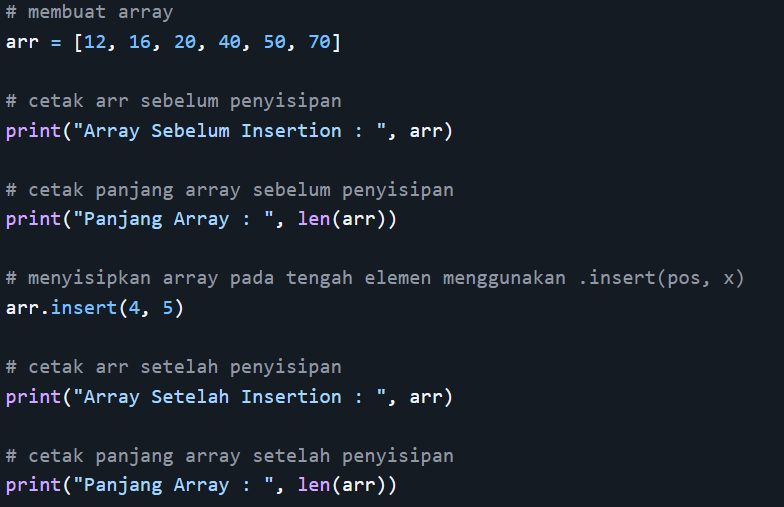
Baris 3 = Mencetak jumlah elemen dalam array sebelum penambahan. Fungsi len() menghitung panjang list.

Baris 4 = Menambahkan elemen 26 ke **akhir array** menggunakan metode .append().

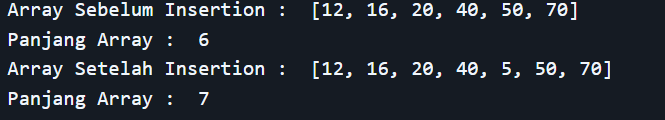
Baris 5 = Mencetak isi array setelah elemen 26 disisipkan di bagian akhir.

Baris 6 = Mencetak panjang array setelah penambahan. Harusnya bertambah satu elemen menjadi 7.

PRAKTEK 10



HASIL



PENJELASAN

Baris 1 = Membuat list arr dengan 6 elemen angka.

Baris 2 = Menampilkan isi array sebelum dilakukan penyisipan elemen baru.

Baris 3 = Menampilkan jumlah elemen dalam array sebelum penambahan. Hasilnya adalah 6.

Baris 4 = Menyisipkan nilai 5 ke dalam array di posisi indeks 4.  
Elemen yang sebelumnya berada di posisi tersebut (50) dan setelahnya akan bergeser ke kanan.

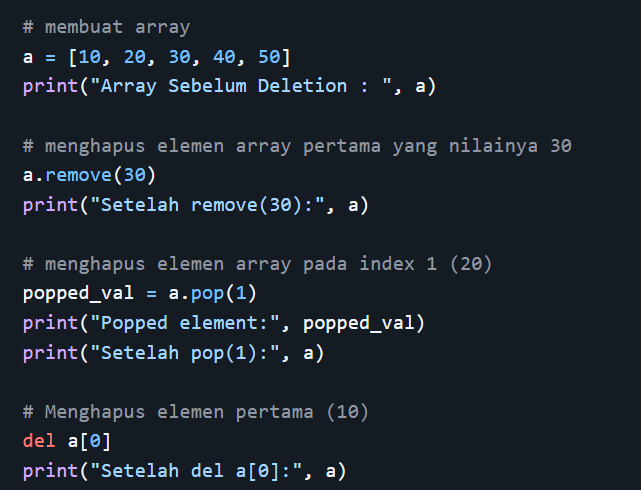
Setelah penyisipan, isi array menjadi:

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

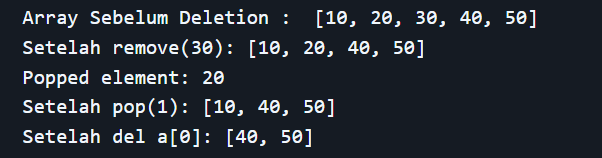
Baris 5 = Menampilkan isi array setelah elemen 5 disisipkan di indeks ke-4.

Baris 6 = Menampilkan panjang array setelah penambahan. Sekarang panjangnya menjadi 7.

PRAKTEK 11



HASIL



PENJELASAN

Baris 1 = Membuat sebuah list bernama a yang berisi elemen-elemen: [10, 20, 30, 40, 50].

Baris 2 = Mencetak string "Array Sebelum Deletion : " diikuti dengan isi list a saat ini. Output: Array Sebelum Deletion : [10, 20, 30, 40, 50]

Baris 3 = Metode remove() digunakan untuk menghapus elemen pertama yang memiliki nilai 30 dari list a.

Setelah perintah ini, list a menjadi [10, 20, 40, 50].

Jika nilai 30 tidak ditemukan dalam list, akan muncul ValueError.

Baris 4 = Mencetak string "Setelah remove(30):" diikuti dengan isi list a setelah elemen 30 dihapus. Output: Setelah remove(30): [10, 20, 40, 50].

Baris 5 = Metode pop() digunakan untuk menghapus elemen pada indeks ke-1 (nilai 20) dari list a dan mengembalikannya.

Nilai yang dihapus disimpan dalam variabel popped\_val.

Setelah perintah ini, list a menjadi [10, 40, 50].

Jika indeks yang diberikan tidak valid, akan muncul IndexError.

Baris 6 = Mencetak string "Popped element:" diikuti dengan nilai yang disimpan dalam variabel popped\_val. Output: Popped element: 20.

Baris 7 = Mencetak string "Setelah pop(1):" diikuti dengan isi list a setelah elemen pada indeks ke-1 dihapus. Output: Setelah pop(1): [10, 40, 50].

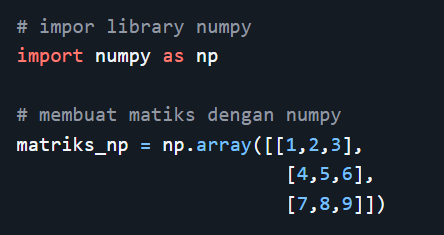
Baris 8 = Pernyataan del digunakan untuk menghapus elemen pada indeks ke-0 (nilai 10) dari list a.

 Setelah perintah ini, list a menjadi [40, 50].

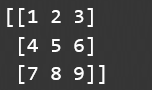
 Jika indeks yang diberikan tidak valid, akan muncul IndexError

Baris 9 = Mencetak string "Setelah del a[0]:" diikuti dengan isi list a setelah elemen pada indeks ke-0 dihapus. Output: Setelah del a[0]: [40, 50].

PRAKTEK 12



HASIL



PENJELASAN

Baris 1 = Pernyataan ini mengimpor library NumPy dan memberinya alias np. NumPy adalah library fundamental untuk komputasi ilmiah di Python, menyediakan objek array multidimensi dan berbagai fungsi untuk operasi matematika dan aljabar line

Baris 2-4 = Membuat objek array 2D (matriks) menggunakan np.array().

Array yang dihasilkan memiliki bentuk 3x3, dengan elemen-elemen sebagai berikut:

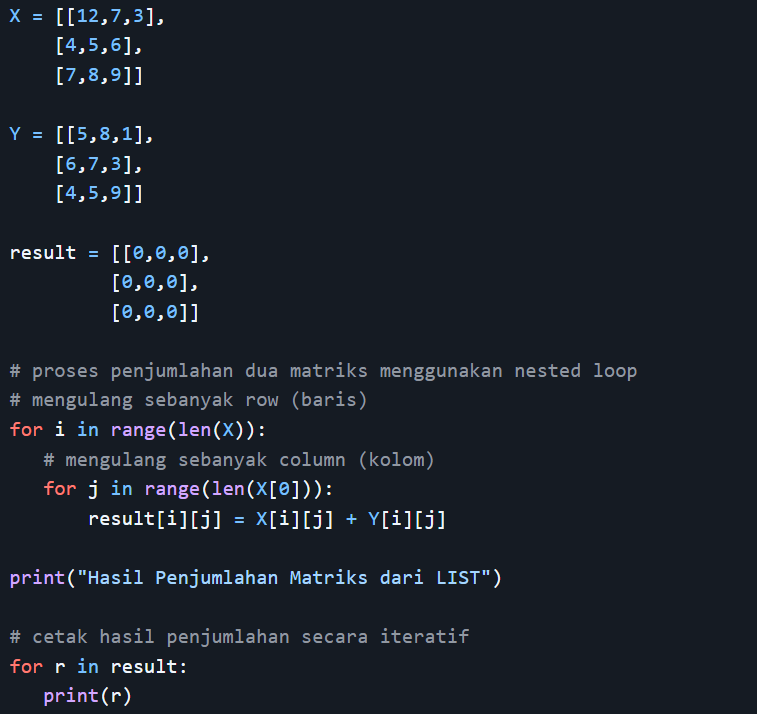
[[1, 2, 3],

[4, 5, 6],

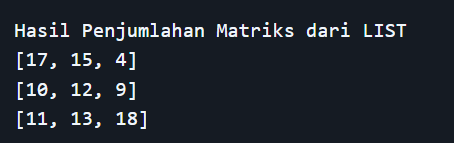
[7, 8, 9]]

Matriks ini dapat digunakan untuk berbagai operasi matematika dan aljabar linear, seperti perkalian matriks, inversi, dan determinan.

PRAKTEK 13



HASIL



PENJELASAN

Baris 1-3 = Matriks X dan Y adalah dua matriks 3x3 yang akan dijumlahkan.

Matriks result diinisialisasi dengan nilai 0, berukuran sama dengan X dan Y, untuk menyimpan hasil penjumlahan.

Baris 4 = len(X) mengembalikan jumlah baris dalam matriks X. Misalnya, jika X adalah matriks 3x3, maka len(X) akan menghasilkan 3.

Fungsi range(len(X)) menghasilkan urutan angka dari 0 hingga len(X)-1, yang digunakan untuk mengakses setiap indeks baris dalam X.

Baris 5 = X[0] mengakses baris pertama dari matriks X. Misalnya, jika X adalah matriks 3x3, maka X[0] akan menghasilkan [12, 7, 3].

len(X[0]) mengembalikan jumlah elemen dalam baris pertama, yang merupakan jumlah kolom dalam matriks. Dalam contoh ini, len(X[0]) akan menghasilkan 3.

Fungsi range(len(X[0])) menghasilkan urutan angka dari 0 hingga len(X[0])-1, yang digunakan untuk mengakses setiap indeks kolom dalam baris ke-i dari X.

Baris 6 = X[i][j] mengakses elemen pada baris ke-i dan kolom ke-j dari matriks X.

Y[i][j] mengakses elemen pada baris ke-i dan kolom ke-j dari matriks Y.

Penjumlahan X[i][j] + Y[i][j] dilakukan untuk setiap pasangan elemen yang bersesuaian dari X dan Y.

Hasil penjumlahan disimpan dalam result[i][j], yaitu elemen pada baris ke-i dan kolom ke-j dari matriks result.

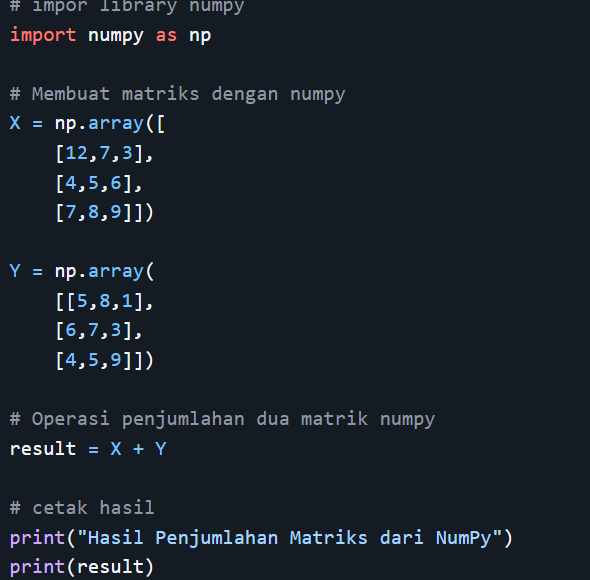
Baris 7 = Perintah ini mencetak string "Hasil Penjumlahan Matriks dari LIST" ke layar.

Tujuannya adalah untuk memberikan konteks atau judul sebelum menampilkan hasil penjumlahan matriks.

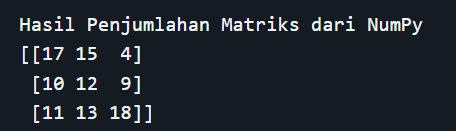
Baris 8 =  for r in result: adalah loop yang mengiterasi setiap elemen dalam list result.Setiap elemen r adalah sebuah list yang mewakili satu baris dalam matriks hasil penjumlahan.

Baris 9 = print(r) mencetak setiap baris r ke layar.

PRAKTEK 14



HASIL



PENJELASAN

Baris 1

* Mengimpor library NumPy dan memberinya alias np.
* NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi.

Baris 2-8

Membuat dua matriks 3x3 menggunakan np.array().

**Penjelasan**:

* X dan Y adalah objek array NumPy yang masing-masing berukuran 3x3.
* Setiap elemen dalam array adalah angka integer.

Baris 9

* Menjumlahkan dua matriks X dan Y secara elemen-wise.
* **Penjelasan**:
  + Operator + digunakan untuk menjumlahkan dua array NumPy. Ini adalah cara yang efisien dan umum digunakan dalam NumPy untuk operasi penjumlahan matriks.
  + Setiap elemen dalam matriks X dijumlahkan dengan elemen yang bersesuaian dalam matriks Y.

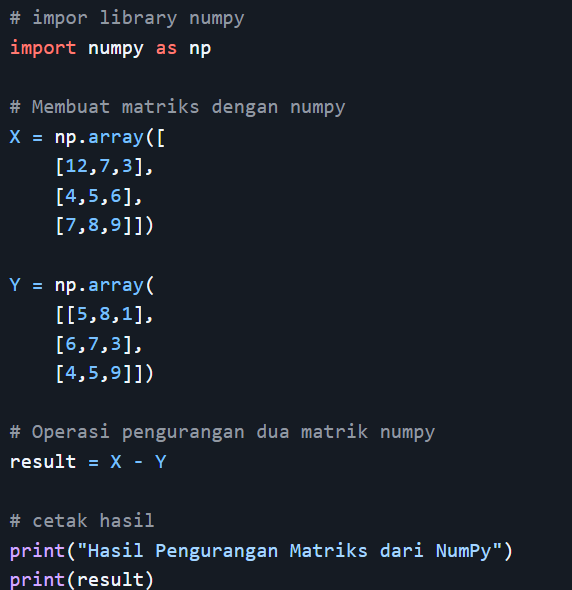
Baris 10-11

Mencetak hasil penjumlahan matriks ke layar.

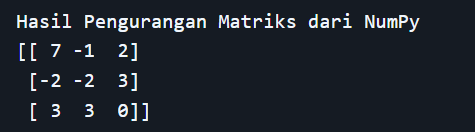
**Penjelasan**:

* print("Hasil Penjumlahan Matriks dari NumPy") mencetak judul atau keterangan.
* print(result) mencetak matriks hasil penjumlahan.

PRAKTEK 15



HAISL



PENJELASAN

Baris 1

* Mengimpor library NumPy dan memberinya alias np.
* **Penjelasan**: NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi.

Baris 2-9

Membuat dua matriks 3x3 menggunakan np.array().

**Penjelasan**:

* X dan Y adalah objek array NumPy yang masing-masing berukuran 3x3.
* Setiap elemen dalam array adalah angka integer.

Baris 10

Mengurangi dua matriks X dan Y secara elemen-wise.

**Penjelasan**:

* Operator - digunakan untuk melakukan pengurangan elemen demi elemen antara dua array NumPy. Ini adalah cara yang efisien dan umum digunakan dalam NumPy untuk operasi pengurangan matriks.
* Setiap elemen dalam matriks X dikurangi dengan elemen yang bersesuaian dalam matriks Y.

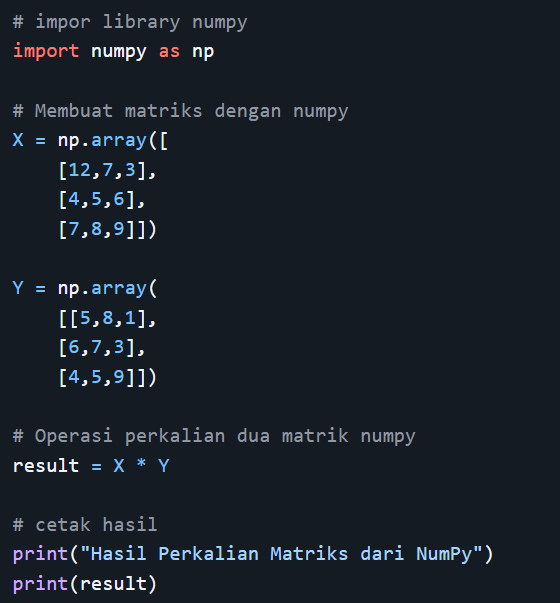
Baris 11-12

Mencetak hasil pengurangan matriks ke layar.

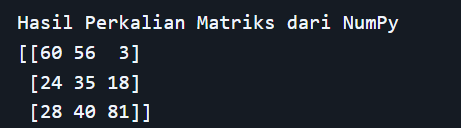
**Penjelasan**:

* print("Hasil Pengurangan Matriks dari NumPy") mencetak judul atau keterangan.
* print(result) mencetak matriks hasil pengurangan.

PRAKTEK 16



HASIL



PENJELASAN

Baris 1

* Mengimpor library NumPy dan memberinya alias np.
* **Penjelasan**: NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi.

Baris 2-9

Membuat dua matriks 3x3 menggunakan np.array().

**Penjelasan**:

* X dan Y adalah objek array NumPy yang masing-masing berukuran 3x3.
* Setiap elemen dalam array adalah angka integer.

Baris 10

Melakukan perkalian elemen demi elemen antara matriks X dan Y.

**Penjelasan**:

* Operator \* digunakan untuk melakukan perkalian elemen demi elemen antara dua array NumPy. Ini dikenal sebagai Hadamard product.
* Setiap elemen dalam matriks X dikalikan dengan elemen yang bersesuaian dalam matriks Y.
* Sebagai contoh, elemen pertama dari X (12) dikalikan dengan elemen pertama dari Y (5), menghasilkan 60. Proses ini diulang untuk setiap elemen yang bersesuaian.

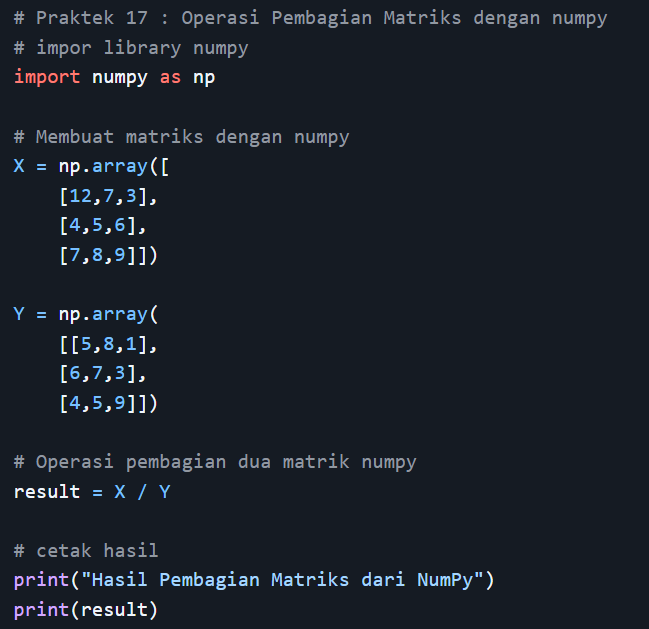
Baris 11-12

Mencetak hasil perkalian matriks ke layar.

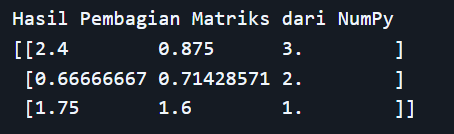
**Penjelasan**:

* print("Hasil Perkalian Matriks dari NumPy") mencetak judul atau keterangan.
* print(result) mencetak matriks hasil perkalian.

PRAKTEK 17



HASIL



PENJELASAN

Baris 1

Mengimpor library NumPy dan memberinya alias np.

**Penjelasan**: NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi.

Baris 2-9

Membuat dua matriks 3x3 menggunakan np.array().

**Penjelasan**:

* X dan Y adalah objek array NumPy yang masing-masing berukuran 3x3.
* Setiap elemen dalam array adalah angka

Baris 10

* Melakukan pembagian elemen demi elemen antara matriks X dan Y.
* **Penjelasan**:
  + Operator / digunakan untuk melakukan pembagian elemen demi elemen antara dua array NumPy. Ini dikenal sebagai Hadamard division.
  + Setiap elemen dalam matriks X dibagi dengan elemen yang bersesuaian dalam matriks Y.
  + Sebagai contoh, elemen pertama dari X (12) dibagi dengan elemen pertama dari Y (5), menghasilkan 2.4. Proses ini diulang untuk setiap elemen yang bersesuaian.

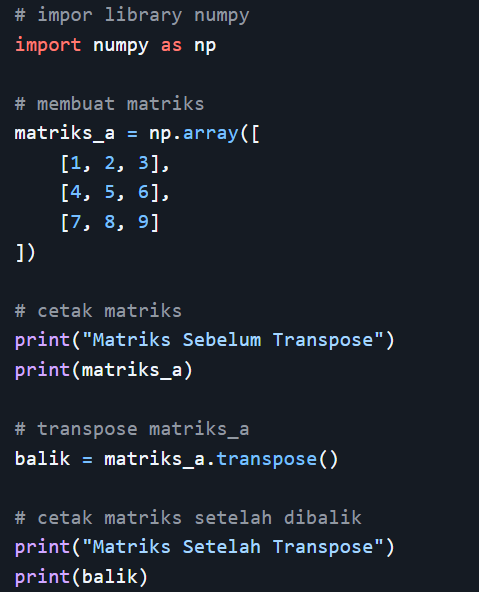
Baris 11

Mencetak hasil pembagian matriks ke layar.

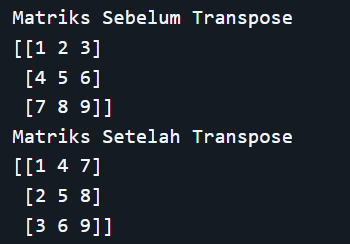
**Penjelasan**:

* print("Hasil Pembagian Matriks dari NumPy") mencetak judul atau keterangan.
* print(result) mencetak matriks hasil pembagian.

PRAKTEK 18



HASIL



PENJELASAN

Baris 1

* Mengimpor library NumPy dan memberinya alias np.
* **Penjelasan**: NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi.

Baris 2-5

Membuat matriks 3x3 menggunakan np.array().

**Penjelasan**:

* matriks\_a adalah objek array NumPy yang berukuran 3x3.
* Setiap elemen dalam array adalah angka integer.

Baris 6-7

Mencetak matriks sebelum dilakukan operasi transpose.

**Penjelasan**:

* print("Matriks Sebelum Transpose") mencetak judul atau keterangan.
* print(matriks\_a) mencetak matriks matriks\_a ke layar.

Baris 8

Melakukan operasi transpose pada matriks matriks\_a.

**Penjelasan**:

* transpose() adalah metode NumPy yang digunakan untuk membalikkan baris dan kolom dari sebuah matriks. Untuk matriks dua dimensi, operasi ini menukar posisi baris dan kolom.
* Hasil dari operasi ini adalah matriks baru yang disimpan dalam variabel balik.

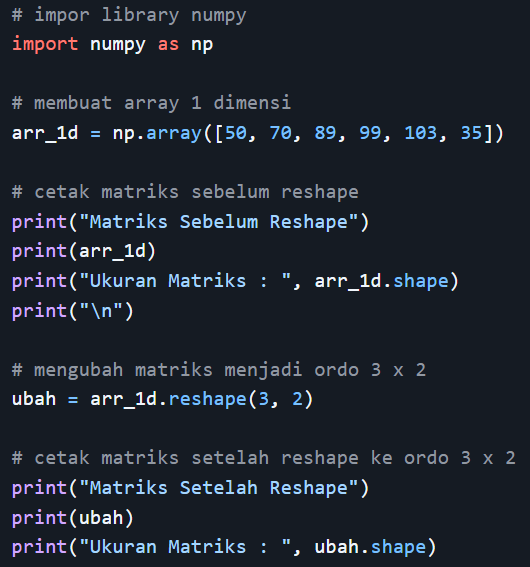
Baris 9-10

Mencetak matriks setelah dilakukan operasi transpose.

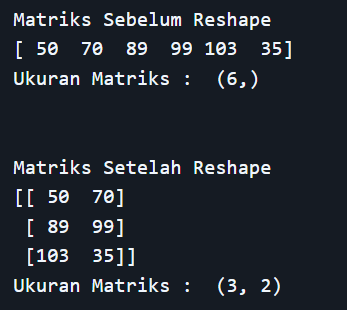
**Penjelasan**:

* print("Matriks Setelah Transpose") mencetak judul atau keterangan.
* print(balik) mencetak matriks balik ke layar.

PRAKTEK 19



HASIL



PENJELASAN

Baris 1

Mengimpor library NumPy dan memberinya alias np.

**Penjelasan**: NumPy adalah library Python yang menyediakan dukungan untuk array multidimensi dan berbagai fungsi matematika tingkat tinggi

Baris 2

* Membuat array 1D menggunakan np.array().
* **Penjelasan**:
  + arr\_1d adalah objek array NumPy yang berukuran 1x6.
  + Setiap elemen dalam array adalah angka integer.

Baris 3-6

Mencetak array sebelum dilakukan operasi *reshape*.

**Penjelasan**:

* print("Matriks Sebelum Reshape") mencetak judul atau keterangan.
* print(arr\_1d) mencetak array arr\_1d ke layar.
* print("Ukuran Matriks : ", arr\_1d.shape) mencetak ukuran (shape) dari array arr\_1d. Metode .shape mengembalikan tuple yang menunjukkan dimensi array

baris 7

Melakukan operasi *reshape* pada array arr\_1d menjadi matriks 2D dengan 3 baris dan 2 kolom.

**Penjelasan**:

* reshape(3, 2) mengubah array 1D menjadi matriks 2D dengan 3 baris dan 2 kolom.
* Setiap elemen dalam array arr\_1d akan diatur ulang dalam matriks ubah sesuai dengan urutan elemen aslinya.

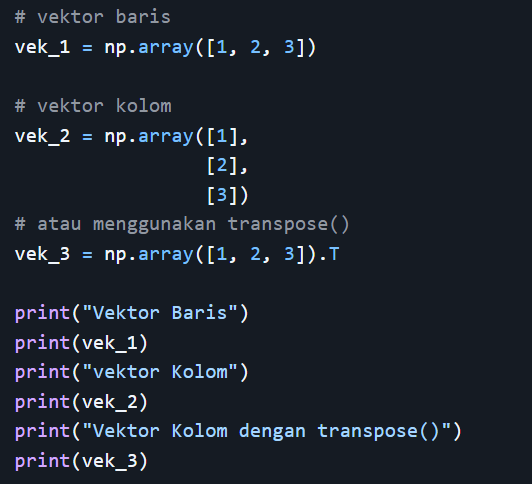
Baris 8-10

Mencetak matriks setelah dilakukan operasi *reshape*.

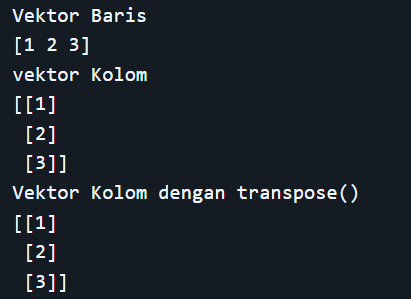
**Penjelasan**:

* print("Matriks Setelah Reshape") mencetak judul atau keterangan.
* print(ubah) mencetak matriks ubah ke layar.
* print("Ukuran Matriks : ", ubah.shape) mencetak ukuran (shape) dari matriks ubah. Metode .shape mengembalikan tuple yang menunjukkan dimensi matriks.

PRAKTEK 20



HASIL



PENJELASAN

Baris 1

Membuat vektor baris dengan elemen 1, 2, dan 3.

**Penjelasan**:

* np.array([1, 2, 3]) membuat array NumPy satu dimensi (1D) dengan elemen-elemen yang diberikan.
* Vektor ini memiliki bentuk (shape) (3,), yang berarti 3 elemen dalam satu dimensi

Baris 2-4

Membuat vektor kolom dengan elemen 1, 2, dan 3.

**Penjelasan**:

* Sintaks ini akan menghasilkan kesalahan (ValueError) karena np.array() mengharapkan satu argumen iterable (seperti list atau tuple) yang berisi elemen-elemen array.
* Untuk membuat vektor kolom yang benar, Anda perlu menyusun elemen-elemen dalam satu iterable, seperti [[1], [2], [3]].

Baris 5

Membuat vektor kolom dengan elemen 1, 2, dan 3 menggunakan operasi transpose.

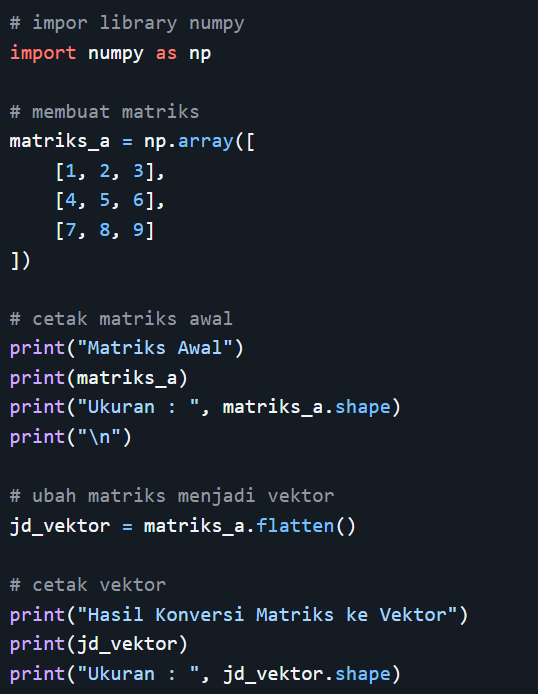
**Penjelasan**:

* np.array([1, 2, 3]) membuat array 1D dengan elemen-elemen yang diberikan.
* .T adalah atribut yang digunakan untuk mentranspose array. Namun, karena array ini satu dimensi, mentranspose-nya tidak mengubah bentuknya.
* Untuk memastikan array menjadi vektor kolom, Anda dapat menggunakan np.reshape(3, 1) atau np.newaxis.

Baris 6-8

* Mencetak vektor-vektor yang telah dibuat ke layar.
* **Penjelasan**:
  + print("Vektor Baris") mencetak judul untuk vektor baris.
  + print(vek\_1) mencetak vektor baris vek\_1.
  + print("Vektor Kolom") mencetak judul untuk vektor kolom.
  + print(vek\_2) mencetak vektor kolom vek\_2.
  + print("Vektor Kolom dengan Transpose") mencetak judul untuk vektor kolom yang dibuat dengan transpose.
  + print(vek\_3) mencetak vektor kolom vek\_3

PRAKTEK 21



HASIL



PENJELASAN

Baris 1 = Baris ini mengimpor pustaka NumPy dengan alias np, yang memungkinkan Anda menggunakan fungsionalitas NumPy dalam kode Anda

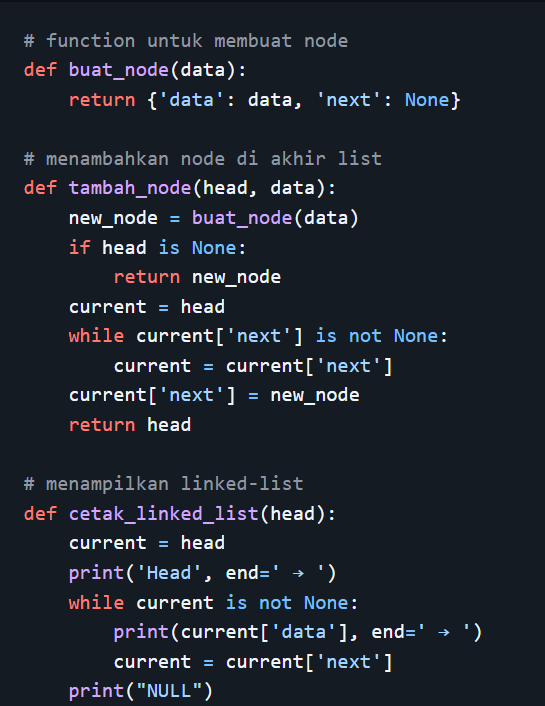
Baris 2-5 = Baris ini membuat objek matriks\_a yang merupakan array 2D (matriks) berukuran 3x3 menggunakan fungsi np.array(). Matriks ini berisi angka dari 1 hingga 9.

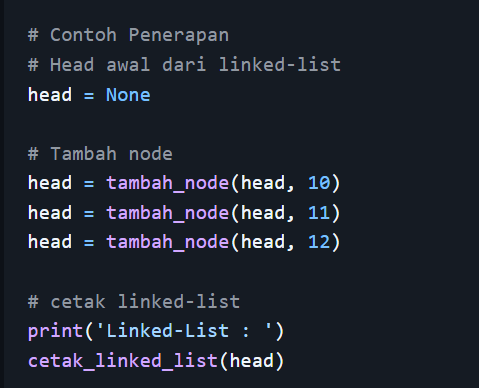
Baris 6-9 = Bagian ini mencetak teks "Matriks Awal", diikuti dengan isi matriks matriks\_a dan ukuran matriks menggunakan properti .shape.

Baris 10 = Baris ini menggunakan metode .flatten() untuk mengonversi matriks 2D matriks\_a menjadi array 1D (vektor) jd\_vektor. Secara default, .flatten() menyusun elemen-elemen matriks dalam urutan baris (row-major order) .

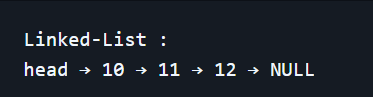
Baris 11-13 = Bagian ini mencetak teks "Hasil Konversi Matriks ke Vektor", diikuti dengan isi dari jd\_vektor dan ukuran vektor tersebut menggunakan properti .shape.

PRAKTEK 22





HASIL



PENJELASAN

Baris 1-2 = Fungsi buat\_node menerima parameter data dan mengembalikan sebuah dictionary yang merepresentasikan sebuah node. Node tersebut memiliki dua kunci: data untuk menyimpan nilai data dan next yang diinisialisasi dengan None, menunjukkan bahwa node ini belum terhubung ke node lain.

Baris 3-12 = Fungsi tambah\_node menambahkan node baru dengan nilai data ke akhir linked list yang dimulai dari head:

* Jika head adalah None (linked list kosong), maka node baru menjadi head.
* Jika tidak, fungsi akan menelusuri linked list hingga menemukan node terakhir (yang next-nya adalah None), kemudian menghubungkannya dengan node baru.

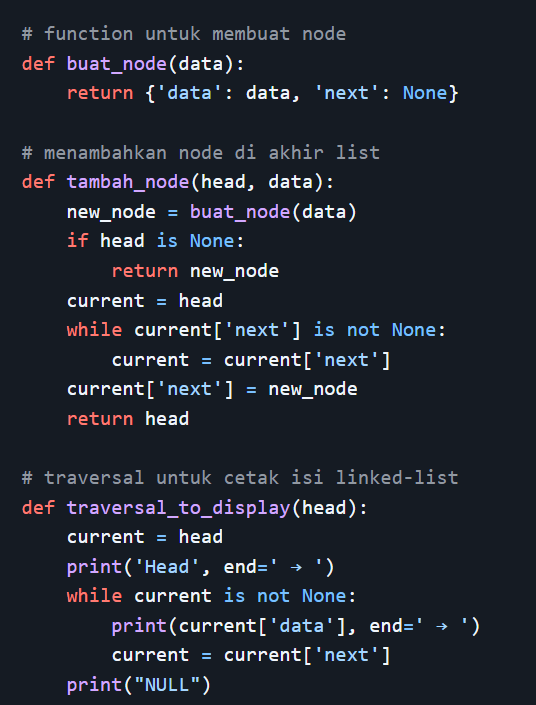
Baris 13-19 = Fungsi cetak\_linked\_list menampilkan isi linked list mulai dari head:

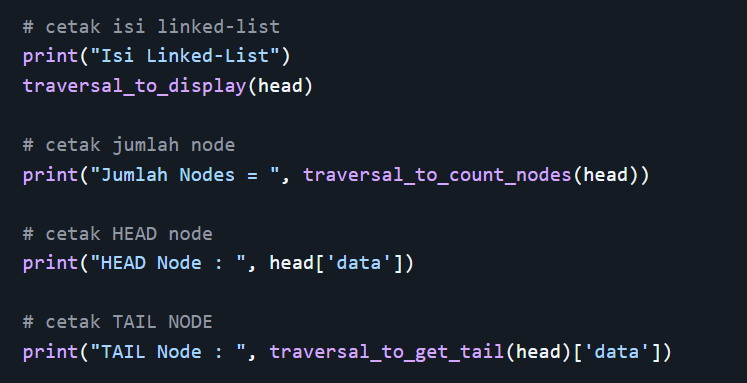
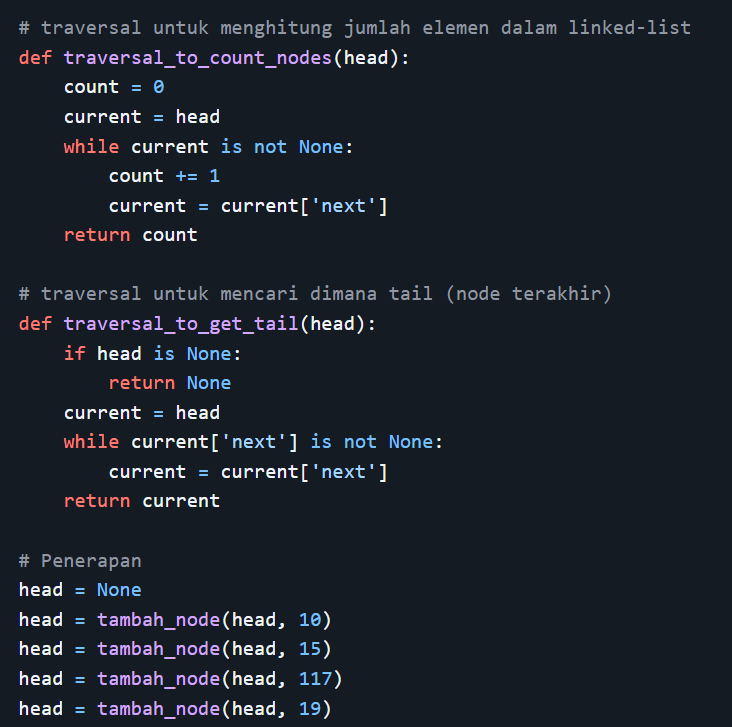
* Dimulai dengan mencetak 'Head → '.
* Kemudian, fungsi menelusuri setiap node, mencetak nilai data dari node tersebut, dan bergerak ke node berikutnya melalui next.
* Proses ini berlanjut hingga mencapai node terakhir (yang next-nya adalah None), kemudian mencetak 'NULL' sebagai penanda akhir linked list.

Baris 20-25 = Bagian ini mendemonstrasikan penggunaan fungsi-fungsi di atas:

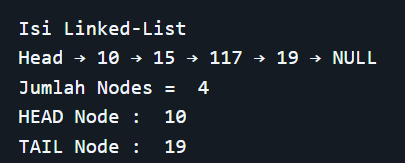
* head diinisialisasi sebagai None, menandakan linked list kosong.
* Tiga node dengan nilai 10, 11, dan 12 ditambahkan ke linked list menggunakan tambah\_node.
* Terakhir, cetak\_linked\_list digunakan untuk menampilkan isi linked list yang telah terbentuk.

PERAKTEK 23





HASIL



PENJELASAN

Baris 1-2 = Fungsi buat\_node menerima parameter data dan mengembalikan sebuah dictionary yang merepresentasikan sebuah node. Node tersebut memiliki dua kunci:

* data: untuk menyimpan nilai data.
* next: diinisialisasi dengan None, menunjukkan bahwa node ini belum terhubung ke node lain.

Baris 3-11 = Fungsi tambah\_node menambahkan node baru dengan nilai data ke akhir linked list yang dimulai dari head:

* Jika head adalah None (linked list kosong), maka node baru menjadi head.
* Jika tidak, fungsi akan menelusuri linked list hingga menemukan node terakhir (yang next-nya adalah None), kemudian menghubungkannya dengan node baru.

Baris 12-18 = Fungsi traversal\_to\_display menampilkan isi linked list mulai dari head:

* Dimulai dengan mencetak 'Head → '.
* Kemudian, fungsi menelusuri setiap node, mencetak nilai data dari node tersebut, dan bergerak ke node berikutnya melalui next.
* Proses ini berlanjut hingga mencapai node terakhir (yang next-nya adalah None), kemudian mencetak 'NULL' sebagai penanda akhir linked list.

Baris 19-25 = Fungsi traversal\_to\_count\_nodes menghitung jumlah node dalam linked list:

* Dimulai dengan count diinisialisasi dengan 0.
* Fungsi menelusuri setiap node, menambah count setiap kali menemukan node baru, dan bergerak ke node berikutnya melalui next.
* Proses ini berlanjut hingga mencapai node terakhir (yang next-nya adalah None), kemudian mengembalikan nilai count.

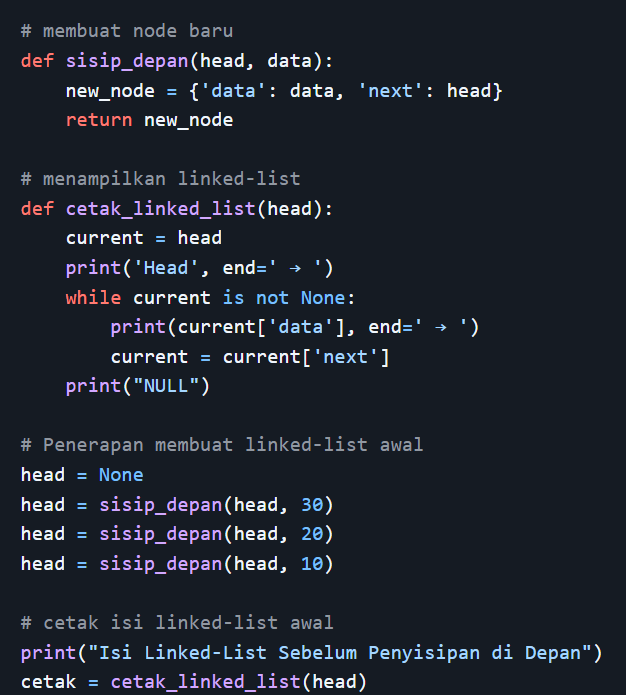
Baris 26-32 = Fungsi traversal\_to\_get\_tail menemukan node terakhir (tail) dalam linked list:

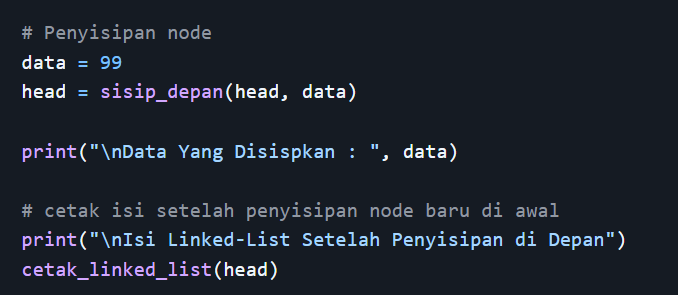
* Jika head adalah None (linked list kosong), maka fungsi mengembalikan None.
* Fungsi menelusuri linked list hingga menemukan node terakhir (yang next-nya adalah None), kemudian mengembalikan node tersebut.

Baris 33-42 = Bagian ini mendemonstrasikan penggunaan fungsi-fungsi di atas:

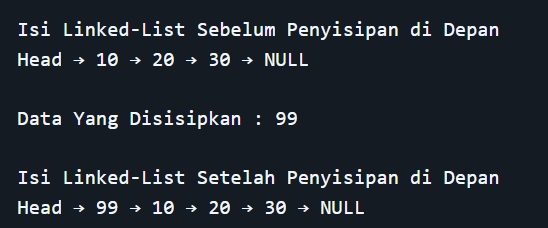
* head diinisialisasi sebagai None, menandakan linked list kosong.
* Empat node dengan nilai 10, 15, 117, dan 19 ditambahkan ke linked list menggunakan tambah\_node.
* Fungsi traversal\_to\_display digunakan untuk menampilkan isi linked list yang telah terbentuk.
* Fungsi traversal\_to\_count\_nodes digunakan untuk menghitung jumlah node dalam linked list.
* Fungsi traversal\_to\_get\_tail digunakan untuk menemukan node terakhir (tail) dalam linked list.

PRAKTEK 24





HASIL



PENJELASAN

Baris 1-3 = Fungsi sisip\_depan menerima dua parameter: head (node pertama dari linked list) dan data (nilai yang akan disisipkan). Fungsi ini membuat sebuah dictionary baru yang merepresentasikan node dengan dua kunci:

* data: menyimpan nilai yang diberikan.
* next: menunjuk ke node sebelumnya yang menjadi head.

Node baru ini kemudian dikembalikan sebagai head yang baru dari linked list.

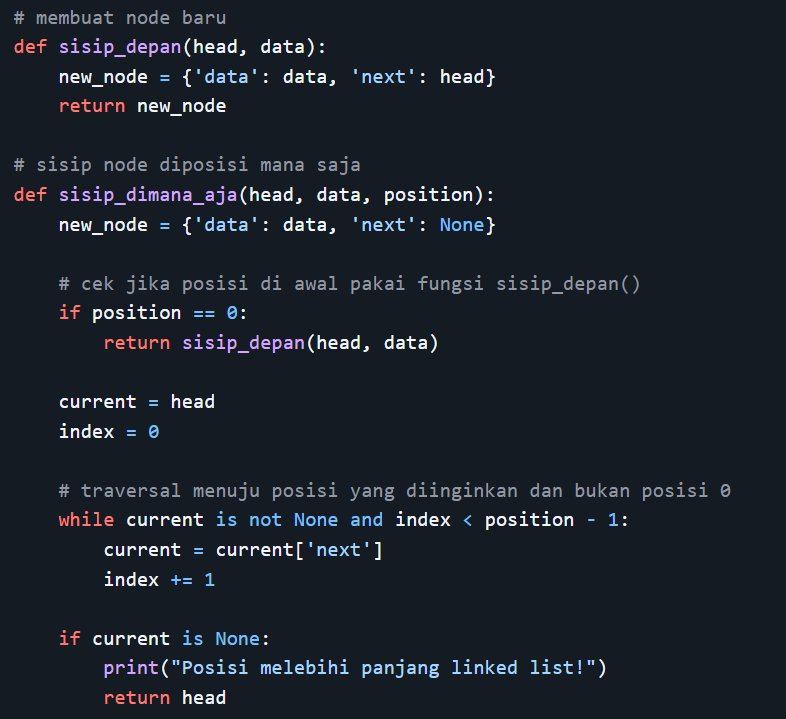
Baris 4-10 = Fungsi cetak\_linked\_list digunakan untuk menampilkan isi linked list mulai dari head:

* Dimulai dengan mencetak 'Head → '.
* Kemudian, fungsi menelusuri setiap node, mencetak nilai data dari node tersebut, dan bergerak ke node berikutnya melalui next.
* Proses ini berlanjut hingga mencapai node terakhir (yang next-nya adalah None), kemudian mencetak 'NULL' sebagai penanda akhir linked list.

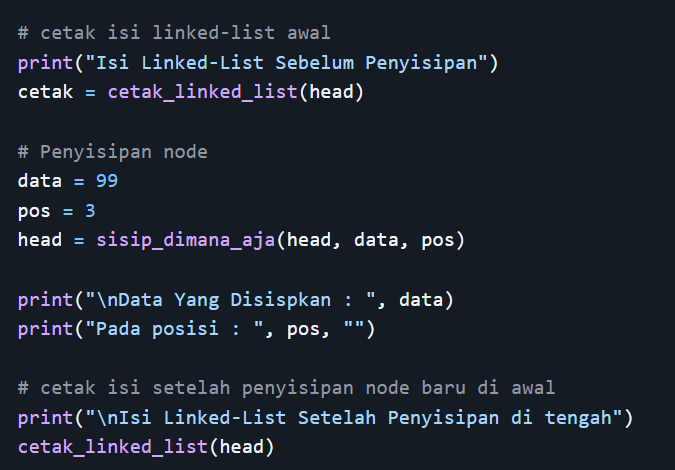
Baris 11-21 = Bagian ini mendemonstrasikan penggunaan fungsi sisip\_depan untuk menyisipkan node baru di awal linked list:

* head diinisialisasi sebagai None, menandakan linked list kosong.
* Tiga node dengan nilai 30, 20, dan 10 ditambahkan ke linked list menggunakan sisip\_depan.
* Fungsi cetak\_linked\_list digunakan untuk menampilkan isi linked list sebelum dan setelah penyisipan node baru dengan nilai 99.

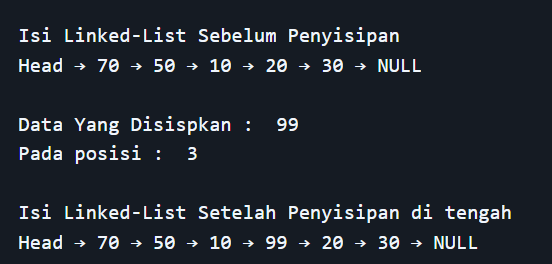
PRAKTEK 25







HASIL



PENJELASAN

Baris 1-3 = **Fungsi sisip\_depan**: Menambahkan node baru di awal linked list.

**Parameter**:

* head: Node pertama dari linked list yang ada.
* data: Nilai yang akan disimpan dalam node baru.

**Proses**:

* Membuat new\_node, yaitu dictionary dengan dua kunci:
  + 'data': Menyimpan nilai data.
  + 'next': Menyimpan referensi ke node pertama (head) dari linked list yang ada.
* Mengembalikan new\_node sebagai node pertama yang baru.

Baris 4-5 = **Fungsi sisip\_dimana\_aja**: Menambahkan node baru di posisi tertentu dalam linked list.

**Parameter**:

* head: Node pertama dari linked list yang ada.
* data: Nilai yang akan disimpan dalam node baru.
* position: Indeks posisi (dimulai dari 0) di mana node baru akan disisipkan.

**Proses**:

* Membuat new\_node, yaitu dictionary dengan dua kunci:
  + 'data': Menyimpan nilai data.
  + 'next': Menyimpan referensi ke node berikutnya (diinisialisasi dengan None).

Baris 6-7 = **Pengecekan Posisi 0**: Jika posisi yang diminta adalah 0 (awal), maka fungsi akan memanggil sisip\_depan untuk menambahkan node baru di awal linked list.

Baris 8-9 = **Inisialisasi Traversal**: Menyiapkan variabel current untuk menelusuri linked list dan index untuk menghitung posisi saat traversal.

Baris 10-12 = **Traversal**: Melakukan perulangan untuk menelusuri linked list hingga mencapai posisi sebelum posisi yang diinginkan (position - 1).

* **Kondisi Berhenti**:
  + current is not None: Masih ada node berikutnya.
  + index < position - 1: Belum mencapai posisi yang diinginkan.

Baris 13-15 = **Pengecekan Posisi Valid**: Jika setelah traversal, current adalah None, berarti posisi yang diminta melebihi panjang linked list. Fungsi akan mencetak pesan kesalahan dan mengembalikan head tanpa perubahan.

Baris 16-19 = **Penyisipan Node Baru**:

* Mengatur 'next' dari new\_node untuk menunjuk ke node setelah current.
* Mengubah 'next' dari current untuk menunjuk ke new\_node, sehingga new\_node disisipkan setelah current.

**Mengembalikan head**: Mengembalikan node pertama dari linked list yang telah diperbarui.

Baris 20-26 = **Fungsi cetak\_linked\_list**: Menampilkan isi dari linked list mulai dari head.

**Proses**:

* Menelusuri setiap node dalam linked list.
* Mencetak nilai 'data' dari setiap node diikuti dengan tanda panah (→).
* Setelah mencapai akhir linked list (current adalah None), mencetak "NULL" sebagai penanda akhir.

Baris 27-32 = **Membuat Linked List**: Menambahkan beberapa node di awal linked list menggunakan fungsi sisip\_depan.

**Hasil**: Linked list menjadi: 70 → 50 → 10 → 20 → 30 → NULL

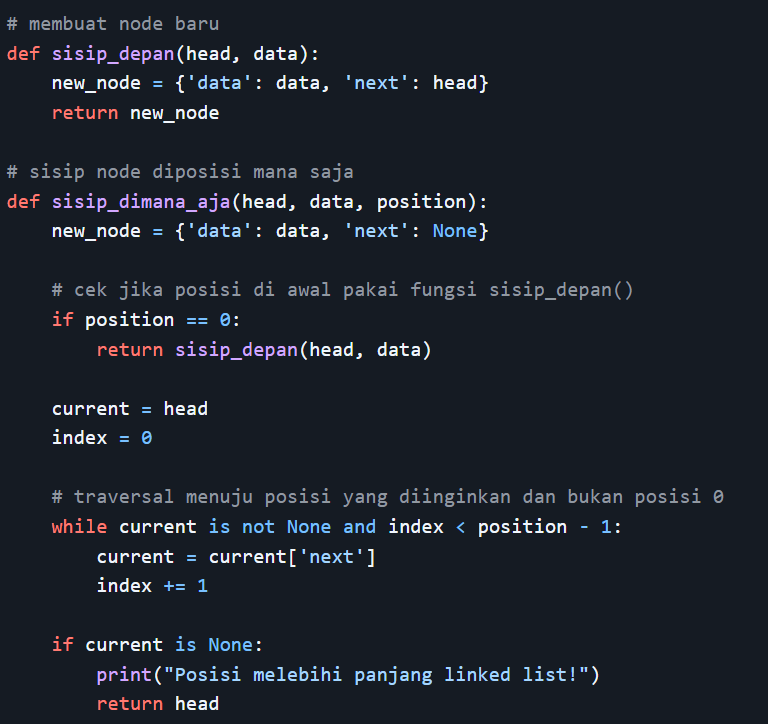
Baris 33-34 = **Menampilkan Linked List**: Mencetak isi linked list sebelum penyisipan node baru.

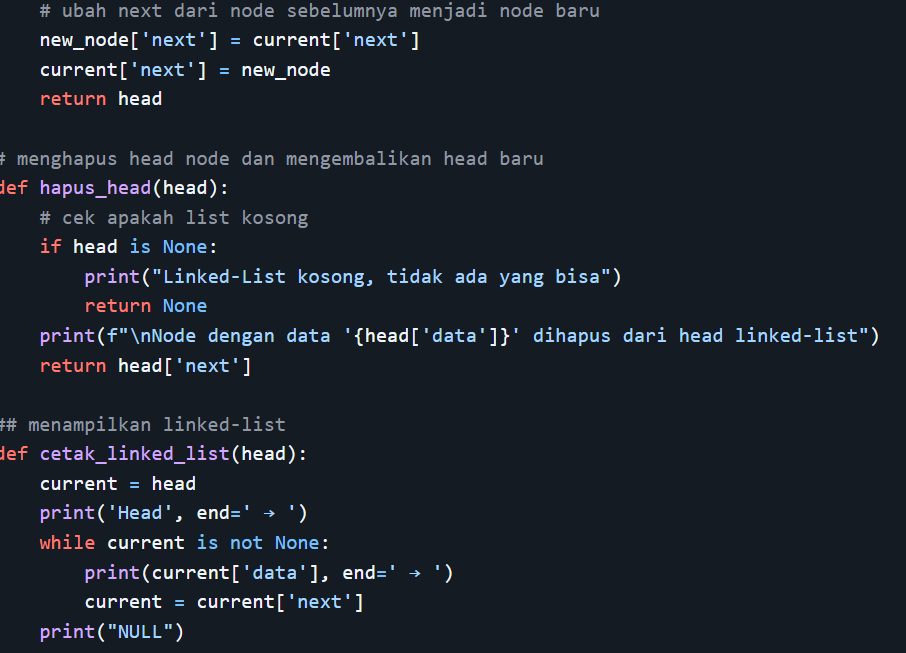
Baris 35-37 = **Menyisipkan Node Baru**: Menambahkan node baru dengan nilai 99 di posisi ke-3 dalam linked list menggunakan fungsi sisip\_dimana\_aja.

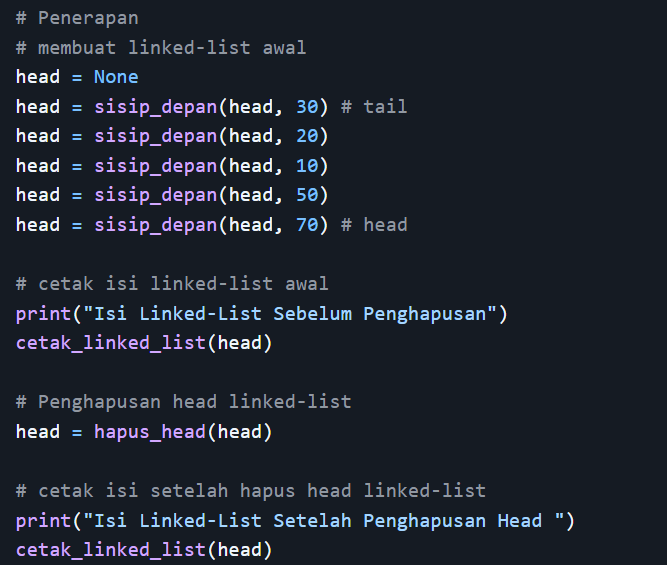
* **Hasil**: Linked list menjadi: 70 → 50 → 10 → 99 → 20 → 30 → NULL.

Baris 38-39 = **Menampilkan Informasi Penyisipan**: Mencetak informasi mengenai data yang disisipkan dan posisi penyisipan.

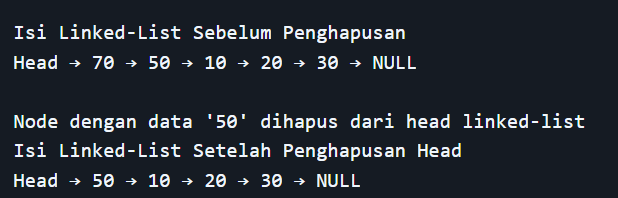
PRAKTEK 26







HASIL



PENJELASAN

Baris 1-3 = **Tujuan**: Menambahkan node baru di awal linked list.

**Proses**:

* Membuat dictionary new\_node dengan dua kunci:
  + 'data': Menyimpan nilai data yang diberikan.
  + 'next': Menunjuk ke node pertama yang lama (head).
* Mengembalikan new\_node sebagai node pertama yang baru.

Baris 4-18 = **Tujuan**: Menyisipkan node baru pada posisi tertentu dalam linked list.

**Proses**:

* Membuat dictionary new\_node dengan dua kunci:
  + 'data': Menyimpan nilai data yang diberikan.
  + 'next': Diinisialisasi sebagai None.
* Jika position adalah 0, panggil fungsi sisip\_depan untuk menambahkan node di awal.
* Melakukan traversal untuk mencapai posisi sebelum posisi yang diinginkan.
* Jika posisi valid, sisipkan new\_node setelah node yang ditemukan.

Baris 19-24 = **Tujuan**: Menghapus node pertama (head) dari linked list.

**Proses**:

* Jika head adalah None, berarti linked list kosong; tampilkan pesan kesalahan dan kembalikan None.
* Jika tidak kosong, tampilkan informasi node yang dihapus.
* Kembalikan node kedua (head['next']) sebagai head yang baru.

Baris 25-31 = **Tujuan**: Menampilkan isi linked list dari head hingga akhir.

**Proses**:

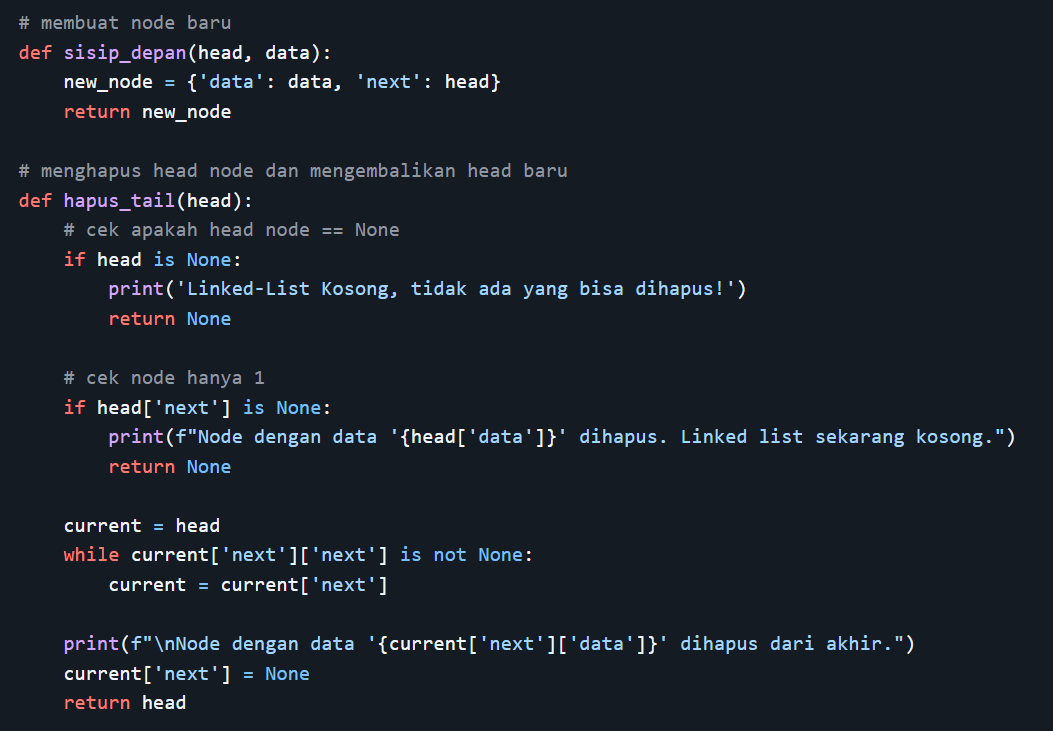
* Melakukan traversal dari head hingga akhir linked list.
* Mencetak nilai 'data' dari setiap node diikuti dengan tanda panah (→).
* Setelah mencapai akhir, mencetak "NULL" sebagai penanda akhir.

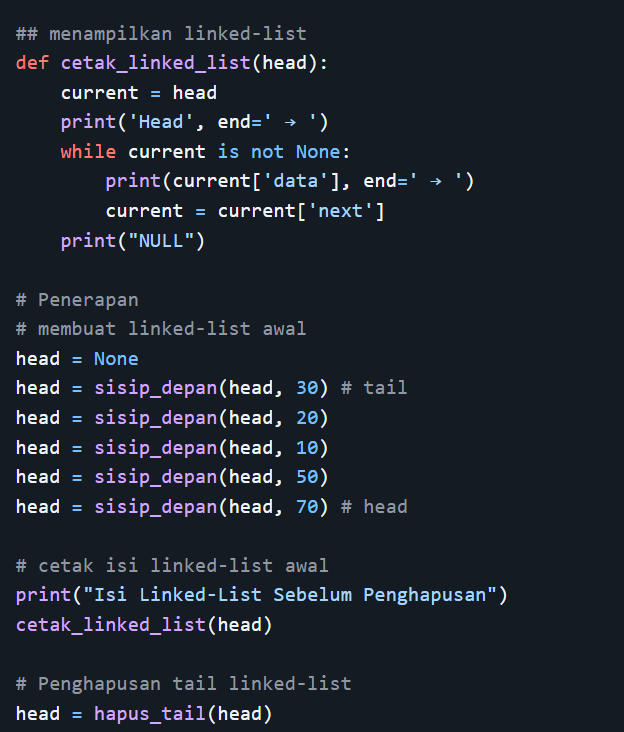
Baris 32-42 = **Tujuan**: Membuat linked list, menampilkan isinya, menghapus node pertama, dan menampilkan isi linked list setelah penghapusan.

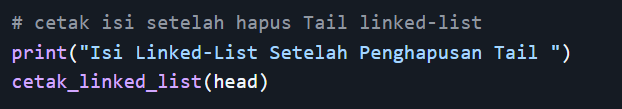
**Proses**:

* + Membuat linked list dengan menambahkan beberapa node di awal menggunakan fungsi sisip\_depan.
  + Menampilkan isi linked list sebelum penghapusan.
  + Menghapus node pertama menggunakan fungsi hapus\_head.
  + Menampilkan isi linked list setelah penghapusan.

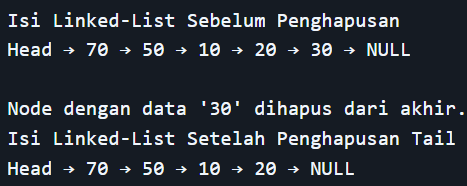
PRAKTEK 27







HASIL



PENJELASAN

**Baris 1**: Mendefinisikan fungsi sisip\_depan yang menerima dua parameter: head (node pertama dari linked list) dan data (nilai yang akan disisipkan).

**Baris 2**: Membuat node baru berupa dictionary dengan dua kunci:

* 'data': Menyimpan nilai yang diberikan (data).
* 'next': Menyimpan referensi ke node sebelumnya (head), sehingga node baru ini akan menjadi node pertama dalam linked list.

**Baris 3**: Mengembalikan node baru sebagai head yang baru dari linked list.

**Baris 4**: Mendefinisikan fungsi hapus\_tail yang menerima satu parameter: head (node pertama dari linked list).

**Baris 5-7**: Mengecek apakah linked list kosong (head adalah None). Jika ya, mencetak pesan bahwa tidak ada yang bisa dihapus dan mengembalikan None.

**Baris 8-10**: Mengecek apakah hanya ada satu node dalam linked list (head adalah satu-satunya node). Jika ya, mencetak pesan bahwa node tersebut dihapus dan mengembalikan None, menjadikan linked list kosong.

**Baris 11-13**: Membuat variabel current yang dimulai dari head. Kemudian, melakukan iterasi untuk menemukan node kedua terakhir (sebelum tail). Iterasi berhenti ketika current['next']['next'] adalah None, yang berarti current adalah node kedua terakhir.

**Baris 14-16**: Mencetak pesan bahwa node dengan data tertentu (tail) dihapus dari akhir. Kemudian, mengubah referensi next dari node kedua terakhir menjadi None, menghapus node terakhir dari linked list. Fungsi mengembalikan head yang tidak berubah.

**Baris 17**: Mendefinisikan fungsi cetak\_linked\_list yang menerima satu parameter: head (node pertama dari linked list).

**Baris 18**: Membuat variabel current yang dimulai dari head.

**Baris 19**: Mencetak kata 'Head' sebagai penanda awal linked list.

**Baris 20-26**: Melakukan iterasi melalui linked list dan mencetak nilai dari setiap node (current['data']), diikuti dengan tanda panah ('→'). Setelah mencetak data, current diperbarui ke node berikutnya (current['next']).

**Baris 27**: Setelah iterasi selesai, mencetak "NULL" sebagai penanda akhir linked list.

**Baris 28**: Inisialisasi head sebagai None, menandakan linked list kosong.

**Baris 29-33**: Menyisipkan node baru di depan linked list dengan nilai 30, 20, 10, 50, dan 70 secara berurutan. Setiap penyisipan menjadikan node baru sebagai head yang baru.

**Baris 34**: Mencetak pesan "Isi Linked-List Sebelum Penghapusan".

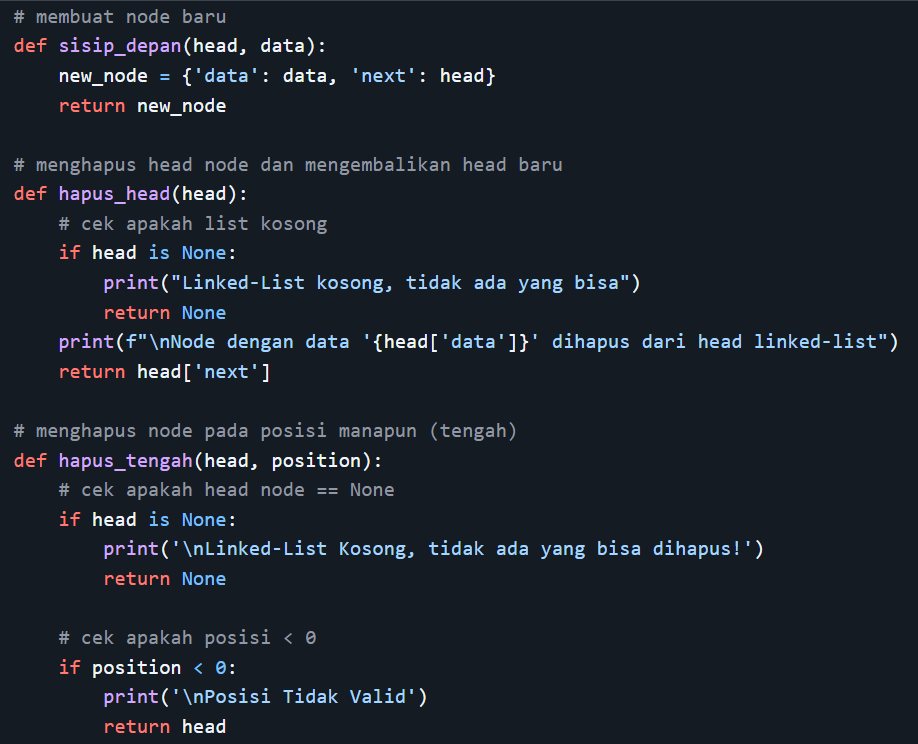
**Baris 35**: Memanggil fungsi cetak\_linked\_list untuk menampilkan isi linked list sebelum penghapusan tail.

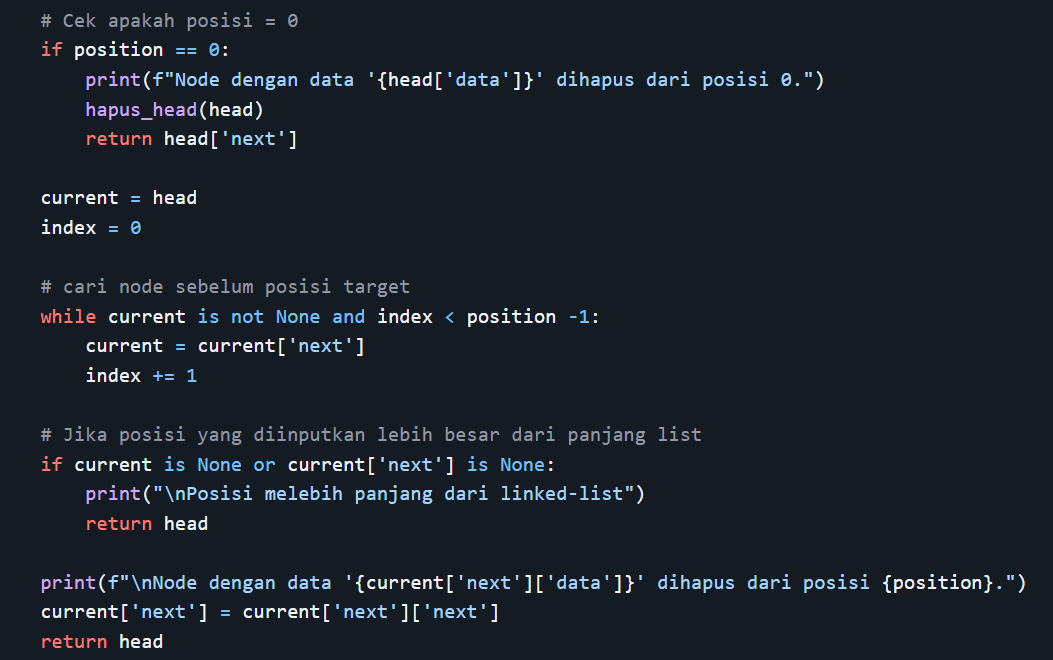
**Baris 36**: Memanggil fungsi hapus\_tail untuk menghapus node terakhir (tail) dari linked list.

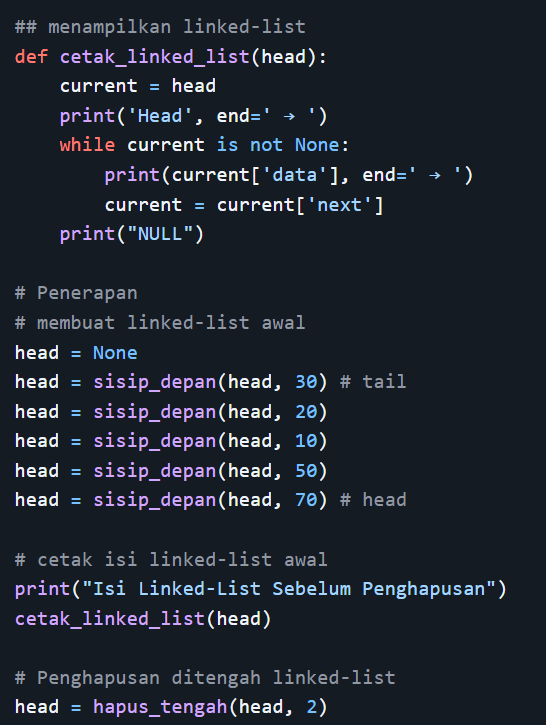
**Baris 37**: Mencetak pesan "Isi Linked-List Setelah Penghapusan Tail".

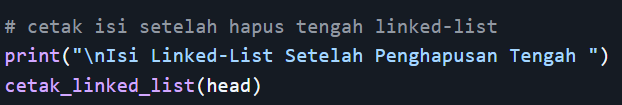
**Baris 38**: Memanggil fungsi cetak\_linked\_list untuk menampilkan isi linked list setelah penghapusan tail.

PRAKTEK 28

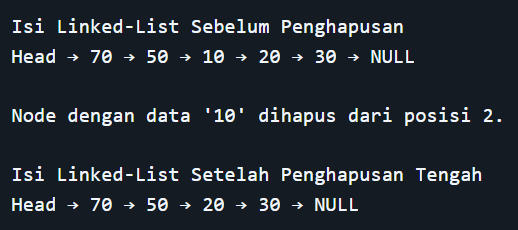








HASIL



PENJELASAN

Baris 1-22: Fungsi ini memiliki tujuan untuk menghapus node pada posisi tertentu (misalnya node ke-1, ke-2, dst.) dalam *linked-list*. Posisi dimulai dari 0 (seperti indeks pada list Python).

Oke, mari kita mulai dari bagaian awal, fungsi hapus\_tengah() menerima dua parameter yaitu head dan position. Dimana head adalah node pertama dalam *linked-list* sementara position sebagai posisi index dari node yang ingin dihapus.

Baris 23:Selanjutnya dalam fungsi ini ada beberapa pengecekan validitas yang pertama adalah mengecek apakah *linked-list* kosong

Baris 24-27:jika head == None, maka list kosong, dan tidak ada node yang bisa dihapus. Tetapi jika tidak lanjutkan ke pengecekan validitas berikutnya

Baris 28-31:Pada kode diatas pengecekan validitas dilakukan ketika user memasukkan parameter posisi kurang nail 0 atau nilai negatif if position < 0:, maka program akan mengeluarkan pesan bahwa posisi yang dimasukkan tidak valid print('\nPosisi Tidak Valid'), dan tidak terjadi penghapusan sehingga kembali ke *linked-list* semula

Baris 32-36: Pada potongan kode diatas kita akan melakukan pengecekan validitas ketika user memamsukkan posisi node adalah 0 if position == 0, maka program akan memberi pesan bahwa node dengan data tertentu dihapus dari posisi 0 print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.") dan menjalankan fungsi hapus\_head(head) yang terpisah dari fungsi ini. Kemudian mengembalikan node seteleh head menjadi head baru.

Baris 37-42: Pada bagian awal kita inisialisasi pointer current pada node pertama (head), dan current akan digunakan untuk menelusuri *linked-list*, selanjutnya index = 0 merupakan penanda awal posisi saat traversal, dan digunakan untuk membandingkan posisition.

Berikutnya adalah melakukan perulangan untuk menemukan node sebelum posisi target. Kemudian position - 1 dimaksudkan untuk menghentikan perulangan ketika menemukan satu node sebelum node yang ingin dihapus. contohnya jika kita ingin menghapus node ke dua maka perulangan akan dihentikan pada ketika mencapai node ke satu.

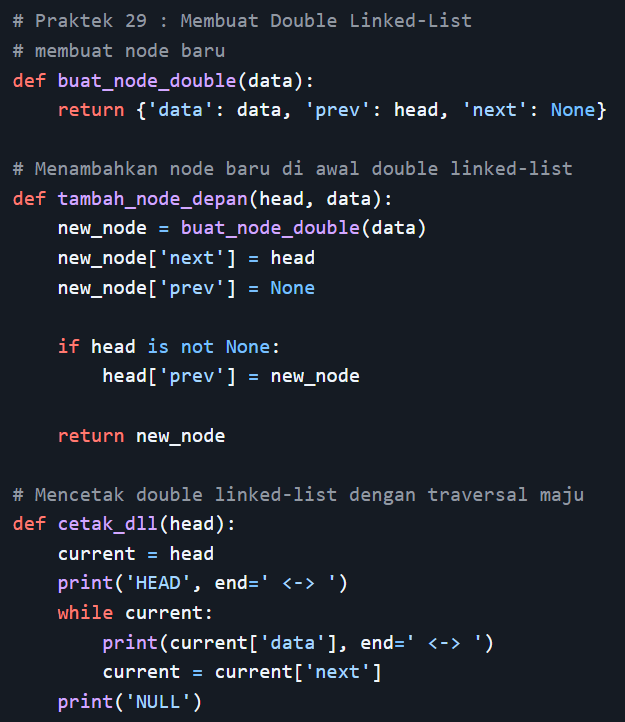
Selama kondisi perulangan diatas masih terpenuhi maka jalankan perintah current = current['next'], artinya pindahkan current ke node berikutnya, dan perbarui nilai index += 1

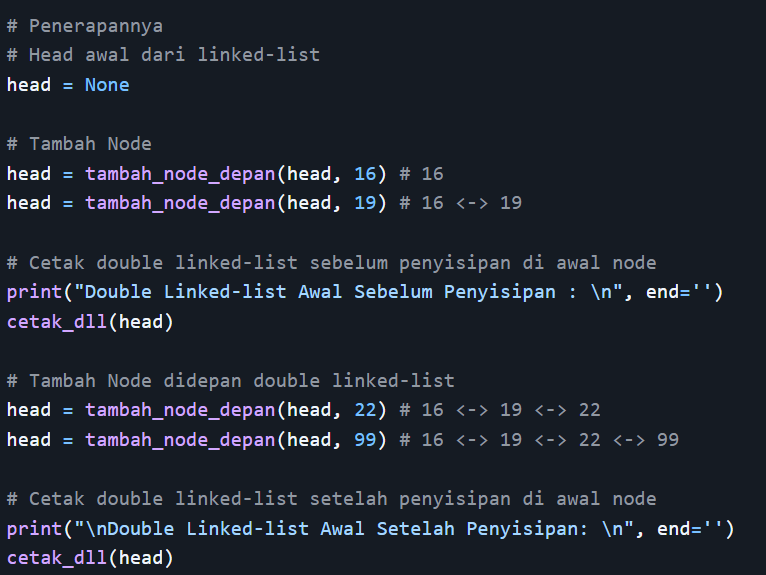
Baris 43-46:potongan kode diatas merupakan sebuah pengecekan validitas yang berfungsi untuk memastikan posisi node yang diinputkan tidak melebih panjang dari *linked-list* if current is None or current['next'] is None. Jika kondisi ini terpenuhi maka cetak pesan print("\nPosisi melebih panjang dari linked-list") dan kembalikan nilai *linked-list* yang awal.

Baris 47-50:potongan kode diatas print(f"\nNode dengan data '{current['next']['data']}' dihapus dari posisi {position}.") diawali pemberitahuan kepada user bahwa node yang berada pada posisi yang dimaksud akan dihapus. Kode current['next']['data'] → itulah node target yang ingin kita hapus. dan mengembalikan nilai head *linked-list* setelah node yang terpilih dihapus.

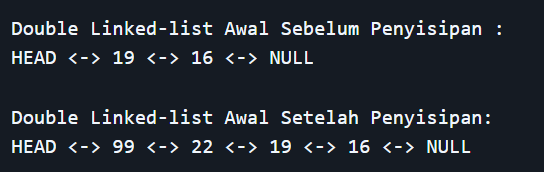
Baris 51-59:Untuk menyisipkan node baru di awal/depan *double linked-list*, kita buat node baru dengan penunjuk/pointer sebelumnya (prev) sebagai NULL dan penunjuk/pointer berikutnya (next) ke head *double linked-list* saat ini. Kemudian, kita periksa apakah *linked-list* tidak kosong, lalu kita perbarui penunjuk/pointer sebelumnya (prev) dari head saat ini ke node baru. Terakhir, kita kembalikan node baru sebagai head *linked-list*.

PRAKTEK 29





HASIL



PENJELASAN

Baris 1-2:Kode diatas adalah sebuah fungsi dengan nama buat\_node\_double(data), yang menerima satu parameter yaitu data. Dimana node baru ini nantinya akan memiliki komponen seperti ini node = {

'data': data, # bergantung pada value yang diinputkan

'prev': None,

'next': None

}

Baris 3-9: Pada potongan kode diatas terdapat 5 langkah, yang terdiri dari :

1. Langkah 1 : membuat node baru dengan memanggil fungsi buat\_node\_double(data)
2. Langkah 2 : setelah node baru terbuat, ubah nilai dari bagaian next dari node untuk menyambungkannya dengan head sebelumnya.
3. Langkah 3 : prev dari node baru diberi nilai None
4. Langkah 4 : jika list tidak kosong, head['prev'] node sebelumnya arahkan ke node baru
5. Langkah 5 : node baru menjadi head baru

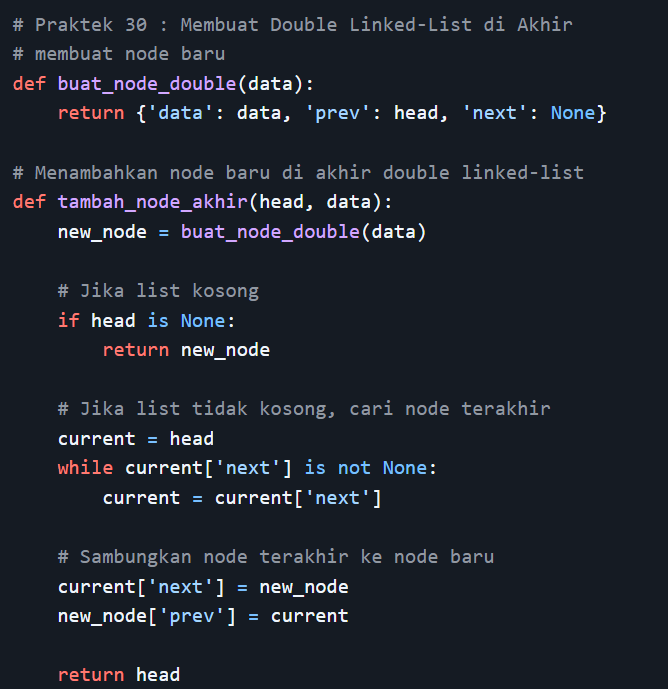
Penyisipan di akhir melibatkan penelusuran seluruh *linked-list* hingga mencapai node terakhir. Kemudian, kita tetapkan referensi berikutnya (next) dari node terakhir untuk menunjuk ke node baru dan referensi sebelumnya (prev) dari node baru untuk menunjuk ke node terakhir. Dengan demikian, node baru menjadi elemen terakhir dalam *linked-list*.

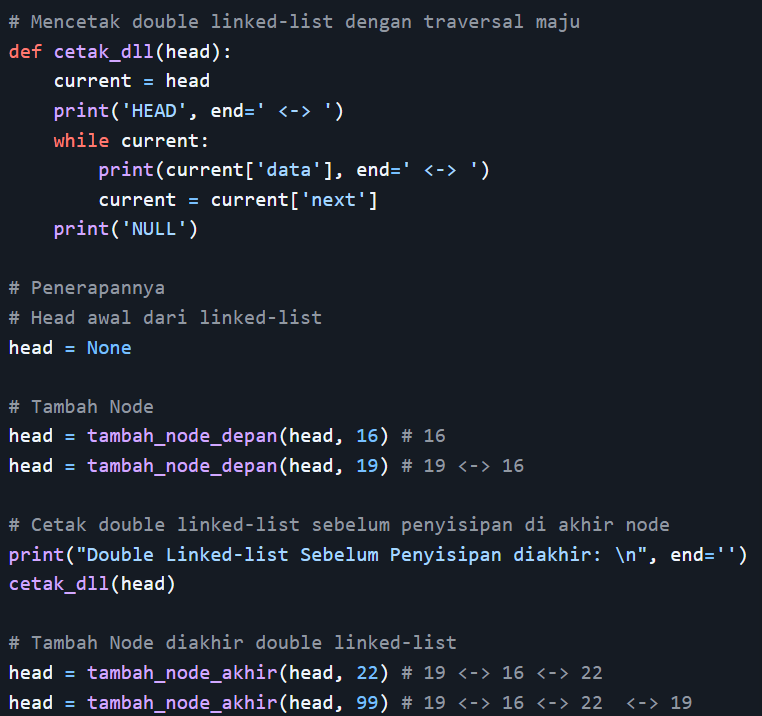
Berikut ini adalah algoritmanya :

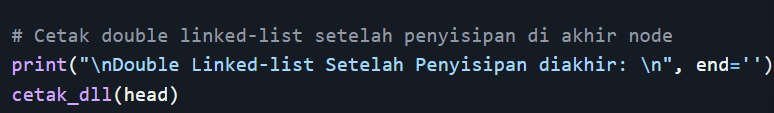
1. Buat node baru (new\_node) dengan data yang baru
2. cek apakah head kosong :
   * Jika kosong -> node baru langsung jadi head
3. Jika tidak kosong :
   * Traversal (jelajahi) dari head ke tail (node akhir)
   * Atur :
     + tail['next'] = new\_node
     + new\_node['prev'] = tail
4. Kembalikan head (karena head tidak berubah)

Berikut adalah implementasi algoritma penyisipan node baru di akhir *double linked-list* :

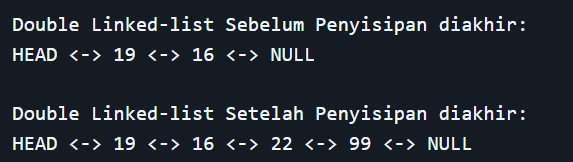
PRAKTEK 30







HASIL



PENJELASAN

Baris 1: bertujuan untuk membuat node baru dengan nilai data pada akhir *double linked-list*. Terdapat dua parameter yaitu :

1. head yang merupakan node pertama (awal) dari *linked-list*
2. data yang merupakan nilai (isi) yang akan dimasukkan dalam node baru

baris 2: baris kode ini digunakan untuk membuat node baru dengan memanggil fungsi diluar yaitu buat\_node\_double(data). Variabel new\_node menjadi *dictionary* yang mewakili node baru, misalnya :

{'data': 50, 'prev': None, 'next': None}

node baru dengan nilai data = 50, prev = None, dan next = None. Nilai prev dan next masih bernilai None karena belum terhubung dengan node lainnya.

Baris 3-4:Potongan kode ini berfungsi untuk pengecekan validitas, apakah list dalam kondisi kosong atau tidak, jiak iya maka new\_node yang baru dibuat, langsung menjadi head dari *double linked-list*, dan kemudian node baru dikembalikan return.

Baris 5-7: dengan potongan kode diatas kita akan menjelajahi (traversal) dari node pertama sama akhir. Sebelum penjelajahan dimulai pertama kita buat variabel bantuan yaitu current yang akan menampung pointer dari setiap node dalam hal ini karena penjelajahan dilakukan maju jadi yang dibutuhkan hanya bagian pointer next saja.

Selanjutnya while current['next'] is not None: atau selama isi dari next pointer tidak None, maka perintah current = current['next'] dijalankan, artinya pindah ke node berikutnya. Perulangan ini akan dilakuakan sampai dengan current['next'] == None atau sudah tidak ada node lagi (node terakhir).

Baris 8-10: kita akan menyambungkan pointer next dari node terakhir yang awalnya None menjadi new\_node, current['next'] = new\_node, dan pointer prev dari new\_node menjadi current, new\_node['prev'] = current.

Terakhir kita kembalikan nilai head dari *linked-list*.

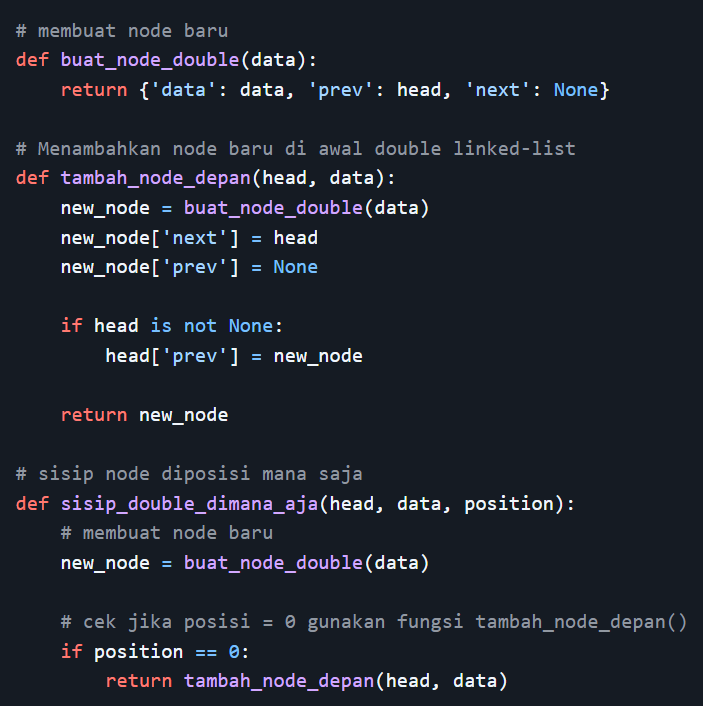
**Penyisipan node pada Tengah atau Posisi Spesifik Double *Linked-List*.**

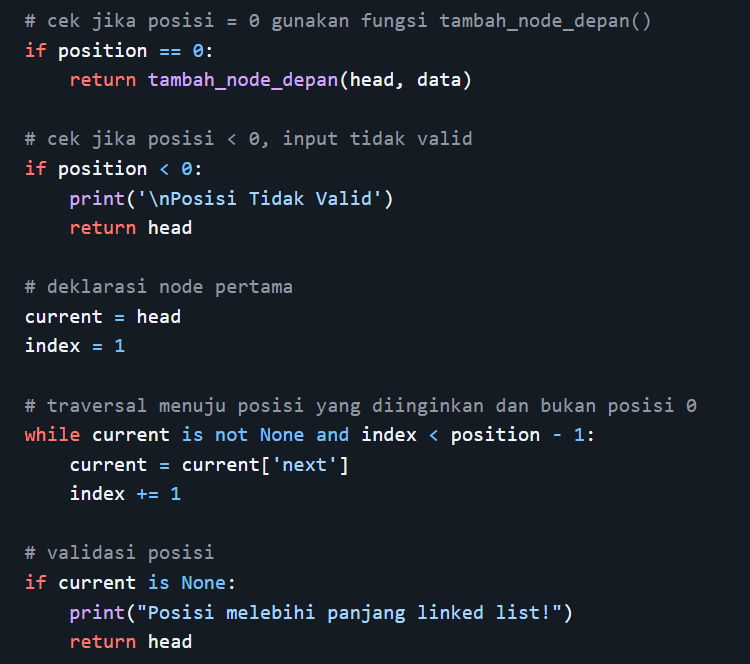
Idenya adalah untuk menelusuri (traversal) *linked list* guna menemukan node pada posisi - 1, katakanlah node saat ini. Jika posisinya valid, buat node baru dengan data yang diberikan dan perbarui pointer-nya: Tetapkan pointer next dari node baru ke next dari node saat ini dan pointer prev dari node baru ke node saat ini. Demikian pula, perbarui pointer next dari node saat ini ke node baru dan pointer prev dari node baru di next ke node baru

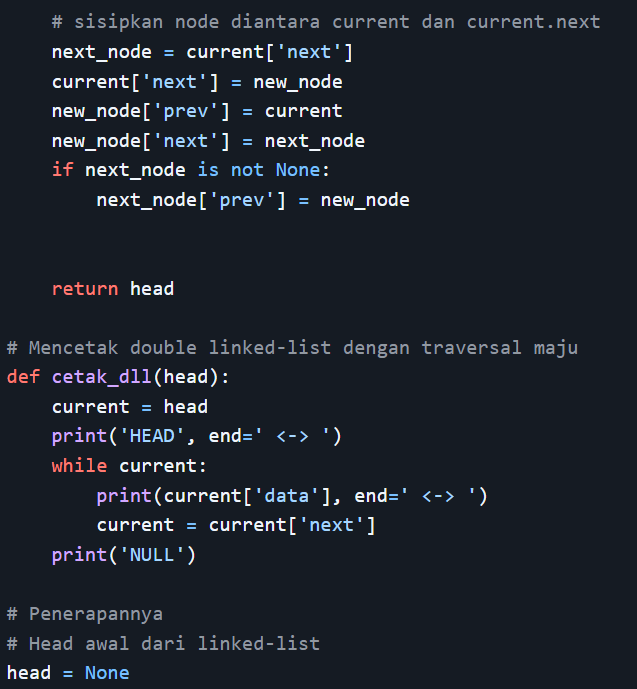
Algoritmanya adalah sebagai berikut ini :

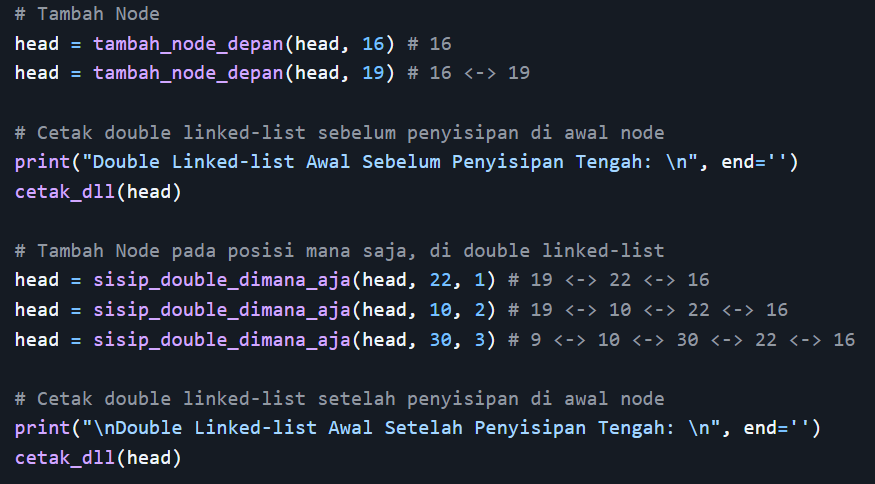
1. Buat node baru dengan data baru
2. Lakukan pengecekan validitas, jika posisi node = 1, maka gunakan fungsi sisip depan
3. Jika posisi < 0, maka munculkan pesan error posisi tidak valid
4. jika posisi bukan 0, atau kurang dari 0, maka lakukan traversal dari head sampai dengan posisi yang diinginkan
5. sisipkan node baru diantara dua node
6. update pointer next dan prev dari node baru agar terhubung
7. kembalikan nilai head

PRAKTEK 31

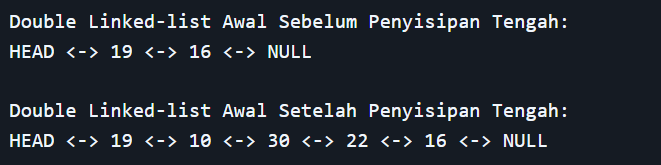








HASIL



PENJELASAN

Baris 1: bertujuan untuk menambahkan atau menyisipkan sebuah node ditengah *double linked-list*. Fungsi ini menerima tiga parameter yaitu :

1. head, merupakan node pertama dari *linked-list*
2. data, merupakan nilai pada node yang akan disisipkan
3. position, merupakan posisi (indeks, mulai dari 0) dimana lokasi node baru akan disispkan ke ``*linked-list*

baris 2: perintah ini digunakan untuk membuat node baru dengan memanggil fungsi buat\_node\_double(data) diluar fungsi ini, dan akan mengembalikan nilai berupa *dictionary* seperti dibawah ini :

{'data': data, 'prev': None, 'next': None}

node baru dengan nilai data = 50, prev = None, dan next = None. Nilai prev dan next masih bernilai None karena belum terhubung dengan node lainnya.

Baris 3-4:Potongan kode ini akan melakukan pengecekan validitas posisi node, dimana jika posisi yang diinputkan adalah sama dengan 0 if position == 0, artinya node baru ingin diletakkan pada awal *linked-list*. Oleh karena itu kita akan memanggil fungsi tambah\_node\_depan() untuk melakukannya.

Baris 5-7:Potongan kode diatas adalah pengecekan validatas inputan dari user, yang mana mengecek apakah posisi node kurang dari 0 (if position < 0>), jika ya, maka tampilkan pesan print('\nPosisi Tidak Valid') dan kembalikan nilai head semula. Potongan kode ini bertujuan untuk mengantisipasi user memasukkan posisi kurang dari 0.

Baris 8-12: pertama adalah kita deklarasikan lebih dulu variabel current sebagai variabel sementara yang kita gunakan untuk memulai proses traversal, dimana variabel ini diberi nilai awal node pertama (current = head). Selain itu kita buat juga variabel penghitung posisi yaitu index yang diberi nilai awal adalah 1, yang menunjukkan proses traversal dimulai dari node berindex 1.

Berikutnya while current is not None and index < position - 1, selama current tidak sama dengan None dan index kurang dari position -1 maka perintah current = current['next'] dijalankan artinya pindah ke node berikutnya. Selain itu update juga nilai dari index dengan menambahkan 1, index +=1 agar kita tahu ada diposisi ke berapa. Perulangan ini akan berhenti sampai node sebelum index node yang menjadi tujuan kita. Contoh node baru akan disisipkan pada index ke 3 maka, traversal akan berhenti tepat satu node sebelumnya, yaitu pada index ke 2.

Baris 13-16:potongan kode diatas untuk mengecek validitas posisi yang diinputkan oleh user tidak melebih panjang dari *linked-list*, maka tampilkan pesan print("Posisi melebihi panjang linked list!") dan kembalikan nilai head semula

Baris 17-23: Lalu bagaimana cara menyisipkannya? Sederhananya kita akan menyisipkan sebuah node diantar dua node yaitu :

1. Setelah current, dan
2. Sebelum current['next'] yang kita sebut next\_node

kode next\_node = current['next'] berfungsi memindahkan node pada posisi sebelum node yang ingin sisipkan ke dalam variabel next\_node. Artinya ini adalah node setelah node yang akan disispkan.

Sebagai analogi, Misalkan kita ingin menyisipkan seorang siswa baru (new\_node) di barisan antara siswa bernama **Andi** (current) dan **Budi** (next\_node). kita simpan dulu informasi tentang siapa Budi next\_node = current['next'], agar tidak "terlupakan" setelah kita ubah koneksi antara Andi dan siswa baru nanti

Selanjutnya kode current['next'] = new\_node untuk mengubah koneksi current, sehingga current sekarang node berikutnya dalah new\_node. Analoginya kita meminta **siswa baru** untuk baris di belakang **Andi**. Artinya setelah Andi ada siswa baru dibelakangya.

Lalu node baru harus mengetahui node apa yang ada didepannya oleh karena itu kode new\_node['prev'] = current. Analoginya **siswa baru** mengetahui **Andi** berada didepannya yang artinya Andi adalah node sebelum **siswa baru**.

Nah selanjutnya **siswa baru** juga harus tahu siapa setelah dia, karenanya kode new\_node['next'] = next\_node digunakan agar node baru terkoneksi dengan node setelahnya ,yaitu next\_node. Analoginya ketika **siswa baru** ditanya siapa setelahnya dia tahu **Budi** setelahnya.

Selanjutnya kita perlu memvalidasi apakah node setelah node baru benar-benar ada if next\_node is not None, jika next\_node == None berarti node\_baru ditempatkan di akhir list. Namun jika next\_node is not None maka kita perlu next\_node perlu mengubah pointernya agar node sebelumnya adalah new\_node. Analoginya kita memberi tahu **Budi** bahwa siswa sebelum dia adalah **siswa baru** dan bukan **Andi** lagi.