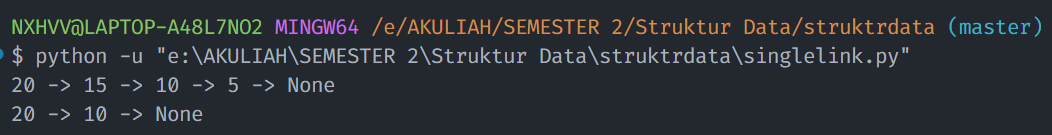
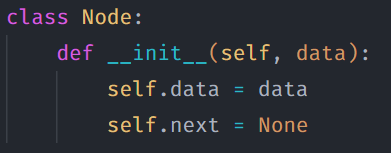
**Latihan 1 Single Linked List**

1. Code Single Linked list

1. Hasil Running dari single Linked List



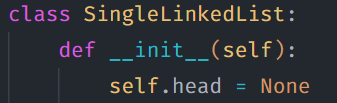
1. Penjelaskan Kode
2. **Kelas Node**



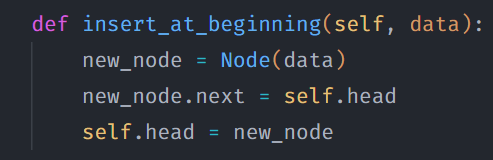
* Kelas ini mewakili satu simpul (node) dalam linked list.
* data menyimpan nilai/data node.
* next adalah pointer ke node berikutnya (default-nya None).

1. **Kelas SingleLinkedList**

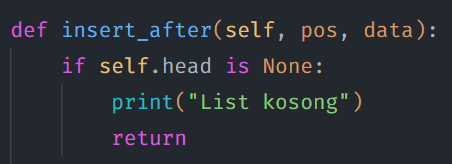
Kelas ini berisi berbagai metode manipulasi pada linked list.



* Inisialisasi linked list. head adalah simpul awal, dimulai dengan None.

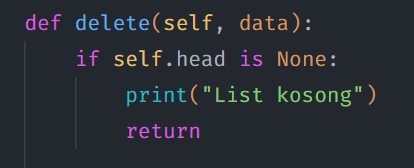
1. **insert\_at\_beginning(data)**

* Menambahkan node di awal linked list.
* Node baru akan menjadi head yang baru.

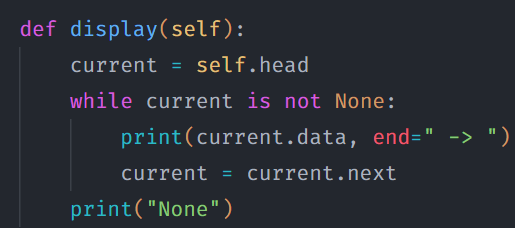
1. **Insert\_after(pos, data)**

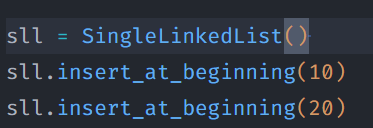
* Menambahkan node **setelah node tertentu** (berdasarkan nilai pos).
* Jika nilai pos ditemukan, node baru disisipkan setelahnya.

1. **delete(data)**



* Menghapus node dengan nilai tertentu.
* Jika nilai ada di head, maka head langsung diganti.
* Jika nilai ada di tengah atau akhir, sambungan node diubah agar node tersebut dilewati.

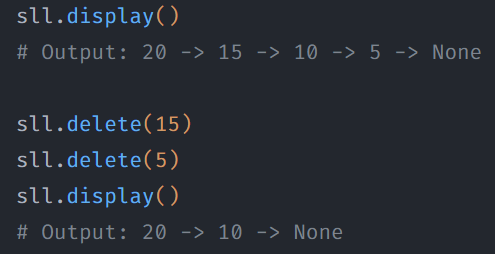
1. display()

* Menampilkan isi linked list dari **head sampai akhir**.
* Menyambungkan tiap data dengan -> sampai None.

* Tambah 10 → [10]
* Tambah 20 di depan → [20 -> 10]

 Tambah 15 setelah node bernilai 20 → [20 -> 15 -> 10]

Tambah 5 setelah node bernilai 10 → [20 -> 15 -> 10 -> 5]

1. 

Node 15 dan 5 dihapus → tersisa [20 -> 10]

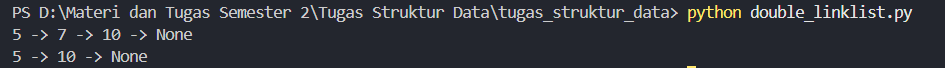
Latihan 2 Double Link List

1. Code Double Link List

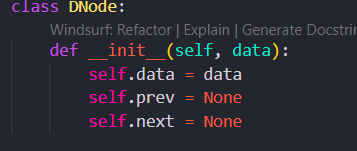
A screen shot of a computer screen

Description automatically generated

1. Hasil Running



1. Penjelasan Code
2. Class Node



Setiap DNode menyimpan:

* data (isi node)
* prev (penunjuk ke node sebelumnya)
* next (penunjuk ke node selanjutnya)

1. Class Double Linked List

A screen shot of a computer code

Description automatically generated

Membuat linked list kosong. self.head adalah node pertama (bisa None jika kosong).

1. Method insert\_before()



Tujuan: **menambahkan node data sebelum node yang bernilai target\_data**

**Prosesnya:**

Jika list kosong, operasi dibatalkan:

A screen shot of a computer code

Description automatically generated

Jika target ada di kepala list, maka node baru dimasukkan paling awal:

A screen shot of a computer code

Description automatically generated

Mencari node yang datanya = target\_data. Jika tidak ada, berhenti:

A screenshot of a computer screen

Description automatically generated

Menyesuaikan semua koneksi:

* prev.next dari node sebelumnya sekarang mengarah ke node baru
* current.prev dari target sekarang adalah node baru

A screen shot of a computer code

Description automatically generated

1. Method delete()

Menghapus node yang berisi data:



Mencari node yang mengandung data tersebut:

A black background with white text

Description automatically generated

Jika tidak ketemu, operasi dibatalkan:

A screen shot of a computer

Description automatically generated

* Jika node yang dihapus bukan di kepala, sesuaikan pointer node sebelumnya dan sesudahnya
* Jika node yang dihapus adalah kepala (head), ubah head ke node berikutnya

A computer screen shot of a program

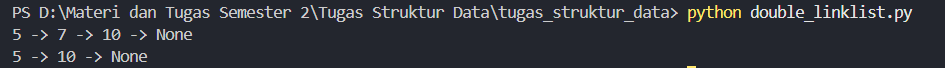
Description automatically generated

1. Method display()

A computer screen shot of text

Description automatically generated

Menampilkan isi linked list dari depan ke belakang, formatnya:



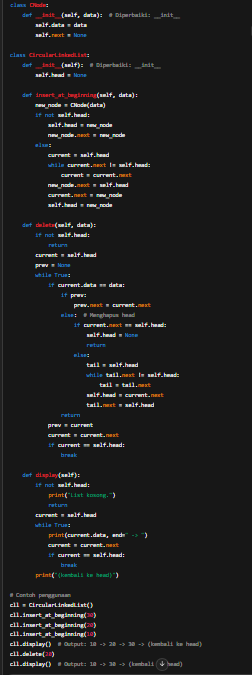
1. Contoh Penggunaan

A screen shot of a computer program

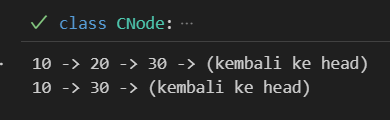
Description automatically generated

Latihan 3 Circular Linked List

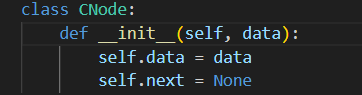
1. Code Circular Linked List



1. Hasil running



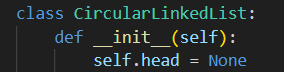
1. Penjelasan Code
2. Class node



 data: menyimpan isi dari node (bisa angka, string, dsb).

* next: menunjuk ke node berikutnya. Karena ini Circular Linked List, maka node terakhir akan menunjuk ke node pertama.

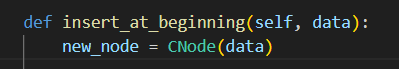
1. Circular Linked List



 head: titik awal dari list.

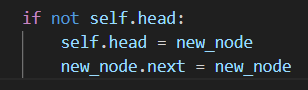
* Jika list kosong, head = None.

1. Metode insert at beginning

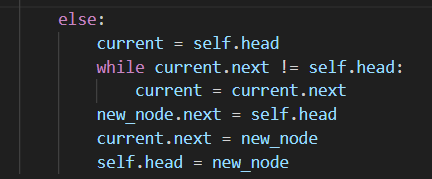


* Membuat node baru dengan data yang diberikan.

1. Metode if not

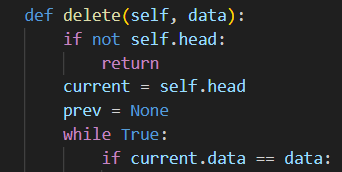


* Jika list kosong, jadikan new\_node sebagai head, dan arahkan next-nya ke dirinya sendiri → membentuk lingkaran.

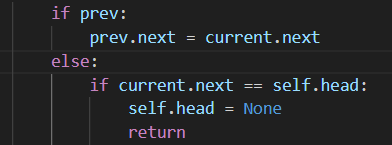


* Menemukan node terakhir (yang menunjuk kembali ke head).
* Update pointer untuk mempertahankan struktur circular.
* Geser head ke node baru (karena kita menyisipkan di depan).

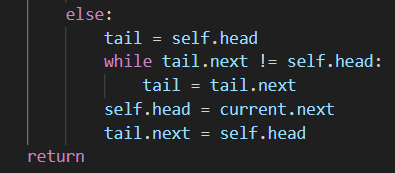
1. Metode delete



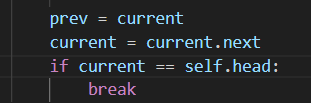
* Mulai dari head, dan siapkan prev untuk melacak node sebelumnya.
* Jika data cocok, kita akan hapus current.



* Kalau hanya satu node di list → hapus dan kosongkan list.

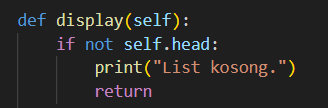


* Jika lebih dari satu node → diperbarui hingga node terakhir

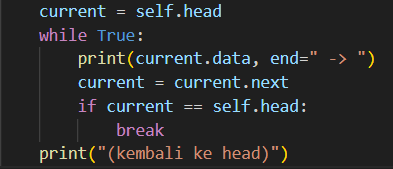


* Kembali ke awal, artinya data tidak ditemukan

1. Metode display

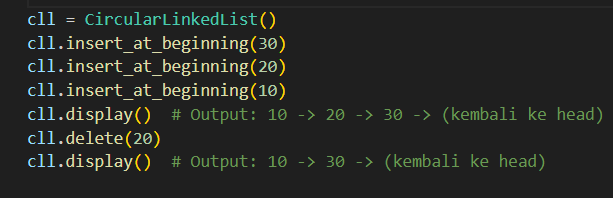


* Tampilkan pesan jika list kosong.



* Mulai dari head, cetak semua data.
* Karena ini circular, kita berhenti saat current kembali ke head.

1. Contoh penggunaan

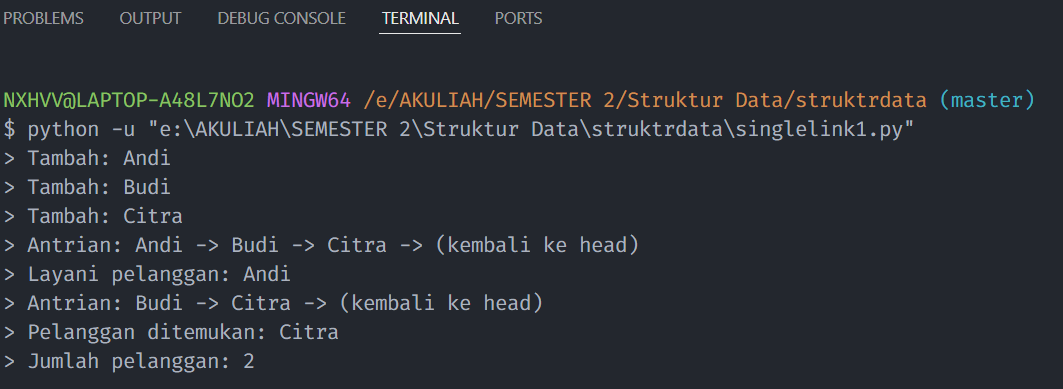


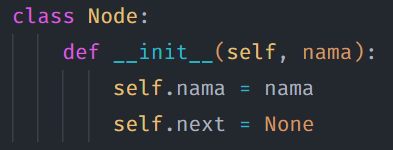
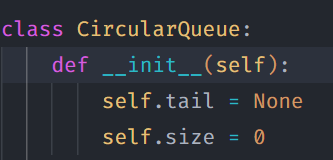
**Tugas Linked List**

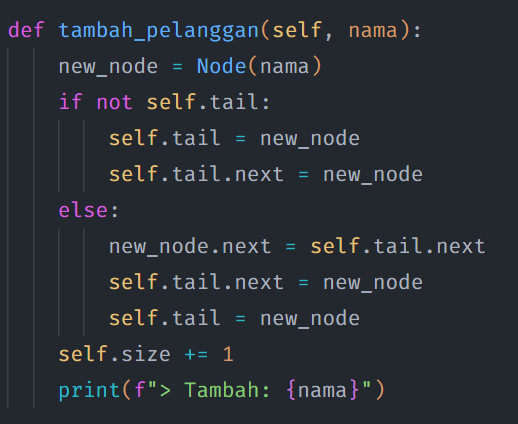
* 1. Code Single Linked List

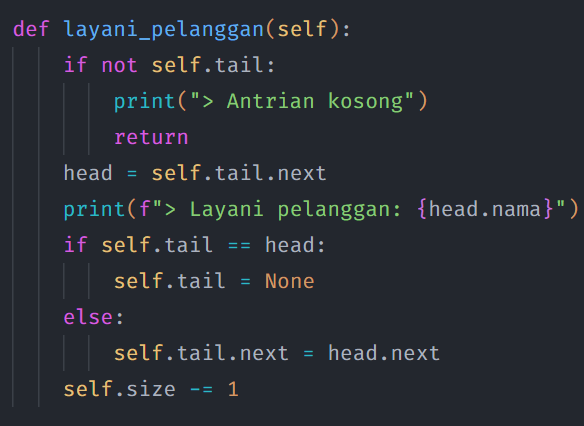


* 1. Hasil Running



* 1. Penjelasan Code
     1. **Kelas Node**
* Setiap Node mewakili satu pelanggan.
* Atribut nama menyimpan nama pelanggan.
* Atribut next menunjuk ke node berikutnya (untuk membentuk circular linked list).
  + 1. **Kelas CircularQueue**

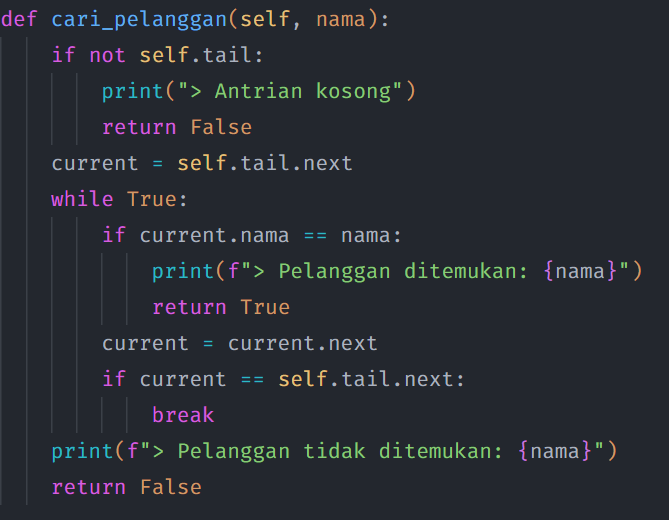
* **tail** menunjuk ke node terakhir (yang menyambung ke head).
* **size** menyimpan jumlah node dalam antrian.
  + 1. **Menambah pelanggan**
* Jika antrian kosong: node pertama menunjuk ke dirinya sendiri.
* Jika tidak kosong: node baru ditambahkan setelah tail, dan menjadi tail baru.
* Karena circular, tail.next selalu menunjuk ke head.
  + 1. **Melayani Pelanggan (hapus dari depan/head)**



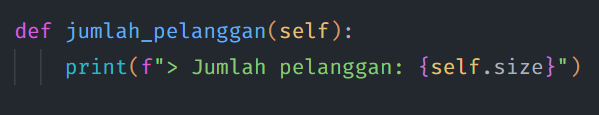
* head adalah node di depan (yaitu tail.next).
* Jika hanya 1 node (tail == head), maka setelah dilayani, antrian menjadi kosong.
* Jika lebih dari 1 node, hapus head dengan mengubah tail.next.
  + 1. **Menampilkan seluruh antrian**

****

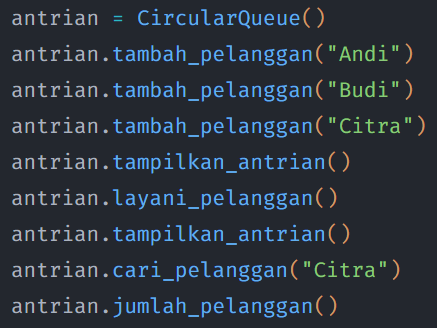
* Mulai dari head (tail.next), iterasi hingga kembali ke head.
* Disimpan di list hasil untuk ditampilkan dengan panah (->).
  + 1. **Mencari pelanggan**

****

* Mencari nama mulai dari head.
* Berhenti jika ditemukan atau sudah satu putaran penuh.
  + 1. **Menampilkan jumlah pelanggan**



Cukup menampilkan nilai dari self.size

* + 1. **Contoh Penggunaan**

ini menunjukkan bagaimana fitur-fitur tadi digunakan