Implementasi Stack dalam Python

Aplikasi Undo/Redo dan Riwayat Navigasi

Dibuat oleh Ammar

Juni 2025

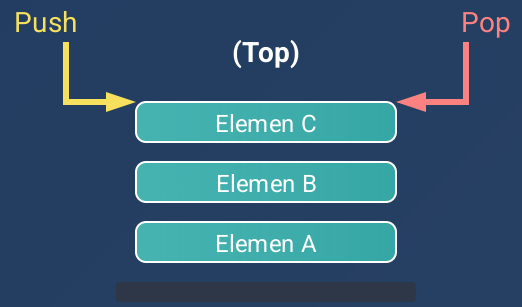
Pendahuluan: Apa itu Stack?

Stack (Tumpukan) adalah struktur data linear abstrak yang mengikuti prinsip:

Last-In, First-Out (LIFO)

Artinya, elemen terakhir yang dimasukkan ke dalam stack adalah elemen pertama yang akan dikeluarkan.

Bayangkan tumpukan piring: Anda menambahkan piring baru di atas (Push), dan saat mengambil piring, Anda mengambil dari paling atas (Pop).



Operasi Dasar Stack

Push
Menambahkan elemen ke puncak stack.

isEmpty
Memeriksa apakah stack kosong.

Pop
Menghapus dan mengembalikan elemen dari puncak stack.

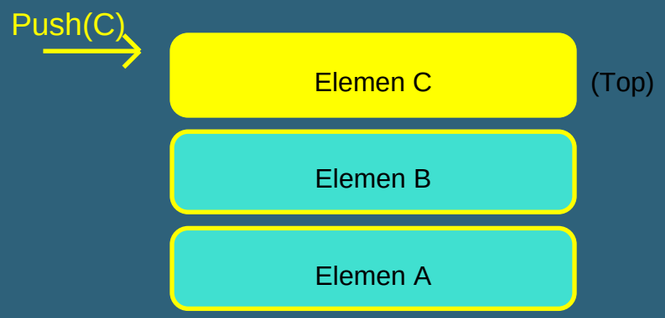
Size
Mendapatkan jumlah elemen dalam stack.

Peek (atau Top)
Melihat elemen di puncak stack tanpa menghapusnya.

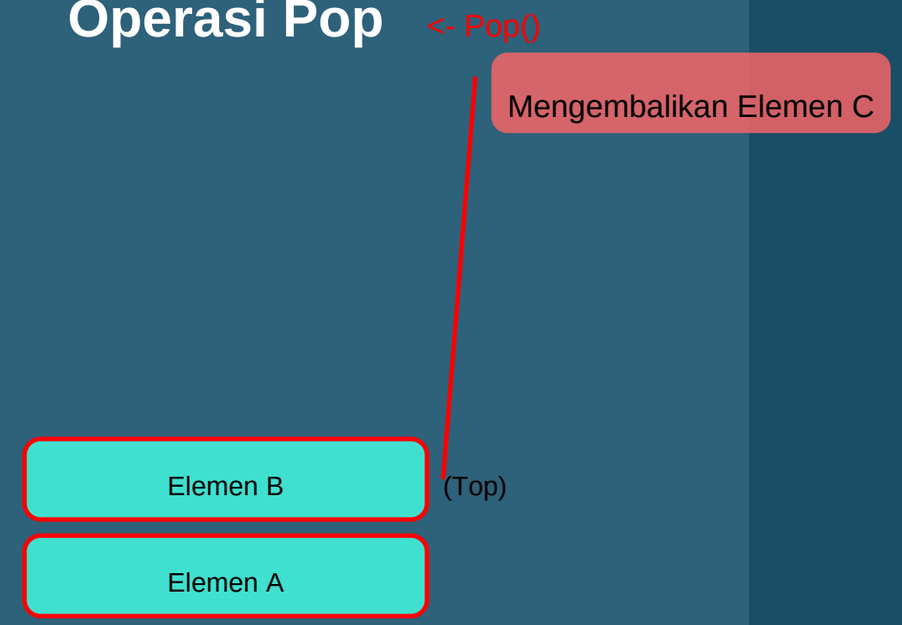
Semua operasi ini memiliki kompleksitas waktu O(1), menjadikan Stack struktur data yang sangat efisien.

Visualisasi Operasi Stack

Operasi Push



Operasi Pop



Visualisasi sederhana operasi Push dan Pop pada Stack.

Implementasi Dasar Stack di Python

Kode Sumber (stack_app.py)

```
class Stack:
    def __init__(self):
        self._items = []

    def push(self, item):
        self._items.append(item)

    def pop(self):
        if not self.is_empty():
            return self._items.pop()
        return None

    def peek(self):
        if not self.is_empty():
            return self._items[-1]
        return None
```

Implementasi ini memanfaatkan metode bawaan list Python yang efisien untuk operasi stack.

Penjelasan Kode Implementasi

Penjelasan Metode Utama:

`__init__`

Menginisialisasi stack dengan list kosong.

`push(item)`

Menambahkan item ke akhir list (puncak stack). Menggunakan list.append().

`pop()`

Menghapus dan mengembalikan elemen terakhir dari list (puncak stack). Menggunakan list.pop(). Mengembalikan None jika stack kosong.

`peek()`

Mengembalikan elemen terakhir tanpa menghapus (_items[-1]). Mengembalikan None jika stack kosong.

`is_empty(), size(), clear(), __str__()`

Memeriksa kekosongan, mengembalikan ukuran, mengosongkan stack, dan representasi string.

Implementasi ini memanfaatkan metode bawaan list Python yang efisien untuk operasi stack.

Aplikasi Stack: Undo/Redo

Konsep Dasar

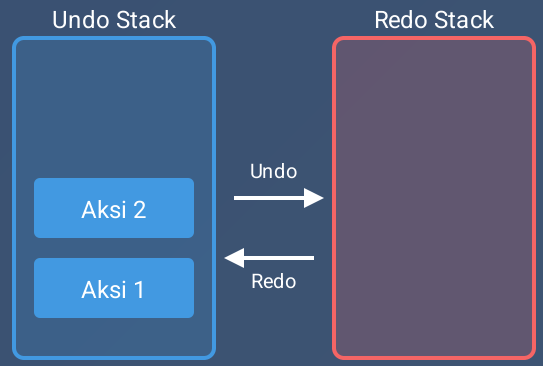
Fitur Undo/Redo yang umum ditemukan di banyak aplikasi (editor teks, software desain) dapat diimplementasikan secara efisien menggunakan dua stack:

Undo Stack
Menyimpan riwayat aksi yang telah dilakukan pengguna. Setiap aksi baru di-push ke stack ini.

Redo Stack
Menyimpan riwayat aksi yang telah di-undo. Stack ini dikosongkan setiap kali pengguna melakukan aksi baru.

Alur Kerja

1. **Aksi Baru:** Aksi dicatat dan di-push ke undo_stack. redo_stack dikosongkan.
2. **Undo:** Aksi terakhir di-pop dari undo_stack dan di-push ke redo_stack. State aplikasi dikembalikan ke sebelum aksi tersebut.
3. **Redo:** Aksi terakhir yang di-undo di-pop dari redo_stack dan di-push kembali ke undo_stack. State aplikasi dikembalikan ke setelah aksi tersebut.



Implementasi Undo/Redo Manager

Kelas UndoRedoManager menggunakan dua instance dari kelas Stack

```
class UndoRedoManager:
    def __init__(self):
        self.undo_stack = Stack()
        self.redo_stack = Stack()
        self.current_state = None

    def perform_action(self, action):
        print(f"Aksi Dilakukan: {action}")
        self.undo_stack.push(action)
        self.redo_stack.clear()
        self.current_state = action

    def undo(self):
        if not self.undo_stack.is_empty():
            action_to_undo = self.undo_stack.pop()
            print(f"Undo: {action_to_undo}")
```

perform_action(action)
Mencatat aksi baru, menambahkannya ke undo_stack, dan mengosongkan redo_stack.

undo()
Memindahkan aksi terakhir dari undo_stack ke redo_stack dan memperbarui state.

redo()
Memindahkan aksi terakhir dari redo_stack kembali ke undo_stack dan memperbarui state.

Demonstrasi Undo/Redo

Output Eksekusi:

```
--- Demonstrasi Undo/Redo ---
Aksi Dilakukan: Mengetik 'Halo'
Aksi Dilakukan: Mengetik ' Dunia'
Aksi Dilakukan: Memformat teks menjadi tebal
Mencoba Undo...
Undo: Memformat teks menjadi tebal
State Sekarang (setelah undo): Mengetik ' Dunia'
Undo: Mengetik ' Dunia'
State Sekarang (setelah undo): Mengetik 'Halo'
Mencoba Redo...
Redo: Mengetik ' Dunia'
State Sekarang (setelah redo): Mengetik ' Dunia'
Mencoba Undo lagi...
```

Output ini menunjukkan bagaimana aksi ditambahkan ke undo_stack, dipindahkan ke redo_stack saat Undo, dan dikembalikan saat Redo.

Perhatikan bagaimana redo_stack dikosongkan ketika aksi baru ("Menghapus semua teks") dilakukan.

Aplikasi Stack: Riwayat Navigasi Browser

Konsep Dasar

Fitur "Back" dan "Forward" pada browser web adalah contoh klasik lain dari penggunaan stack. Biasanya diimplementasikan menggunakan dua stack:

Back Stack

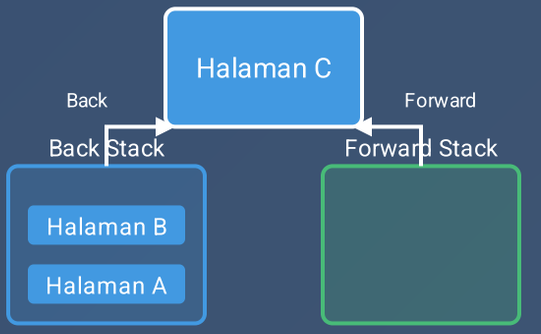
Menyimpan riwayat halaman yang telah dikunjungi sebelumnya. Saat mengunjungi halaman baru, halaman sebelumnya di-push ke stack ini.

Forward Stack

Menyimpan riwayat halaman yang telah di-"Back" dari. Stack ini dikosongkan setiap kali pengguna mengunjungi halaman baru secara langsung (bukan melalui "Forward").

Alur Kerja

1. **Visit Halaman Baru (P):** Jika ada halaman saat ini (C), push C ke back_stack. Jadikan P halaman saat ini. Kosongkan forward_stack.
2. **Go Back:** Jika back_stack tidak kosong, push halaman saat ini (C) ke forward_stack. Pop halaman dari back_stack dan jadikan halaman saat ini.
3. **Go Forward:** Jika forward_stack tidak kosong, push halaman saat ini (C) ke back_stack. Pop halaman dari forward_stack dan jadikan halaman saat ini.



Implementasi Riwayat Navigasi

Kelas NavigationHistory menggunakan dua stack (back_stack dan forward_stack)

```
class NavigationHistory:
    def __init__(self):
        self.back_stack = Stack()
        self.forward_stack = Stack()
        self.current_page = None

    def visit(self, page):
        print(f"Mengunjungi: {page}")
        if self.current_page is not None:
            self.back_stack.push(self.current_page)
        self.current_page = page
        self.forward_stack.clear()
        # self._print_state() # (Output di slide demo)

    def go_back(self):
        if not self.back_stack.is_empty():
```

visit(page)
Mencatat kunjungan ke halaman baru, menyimpan halaman saat ini ke back_stack, dan mengosongkan forward_stack.

go_back()
Memindahkan halaman saat ini ke forward_stack dan mengembalikan halaman sebelumnya dari back_stack.

go_forward()
Memindahkan halaman saat ini ke back_stack dan mengembalikan halaman berikutnya dari forward_stack.

Demonstrasi Riwayat Navigasi

Output Eksekusi (Sebagian):

```
--- Demonstrasi Riwayat Navigasi ---
Mengunjungi: Halaman Utama
[Back]: []
[Current]: Halaman Utama
[Forward]: []
Mengunjungi: Halaman Produk
[Back]: ["Halaman Utama"]
[Current]: Halaman Produk
[Forward]: []
Mengunjungi: Detail Produk A
[Back]: ["Halaman Utama", "Halaman Produk"]
[Current]: Detail Produk A
[Forward]: []
```

Perhatikan bagaimana back_stack terisi saat mengunjungi halaman baru, forward_stack terisi saat kembali (Back), dan forward_stack dikosongkan saat mengunjungi halaman baru setelah melakukan Back/Forward.

Kesimpulan

Struktur data Stack, dengan prinsip LIFO-nya, merupakan alat yang sangat berguna dan fundamental dalam ilmu komputer. Presentasi ini telah menunjukkan:

Implementasi Dasar Stack

Bagaimana membuat kelas Stack yang fungsional menggunakan list Python.

Aplikasi Praktis

Bagaimana Stack dapat digunakan untuk mengimplementasikan fitur-fitur dunia nyata seperti:

- **Undo/Redo:** Menggunakan dua stack untuk mengelola riwayat aksi dan pembatalannya.
- **Riwayat Navigasi:** Menggunakan dua stack untuk mengelola tombol Back dan Forward pada browser atau aplikasi serupa.

Visualisasi

Pentingnya visualisasi untuk memahami bagaimana data bergerak masuk dan keluar dari stack selama operasi.

Fleksibilitas Python

Kemudahan implementasi konsep struktur data ini menggunakan fitur bawaan Python.

Terima Kasih & Referensi

Referensi & Penutup

Kode sumber utama untuk presentasi ini berasal dari file yang Anda berikan:

- **stack_app.py:** Implementasi dasar Stack, Undo/Redo, dan Navigasi.
- **interactive_demo_full.py:** Demo interaktif dengan visualisasi dan operasi tambahan.

Konsep struktur data Stack adalah topik fundamental dalam ilmu komputer dan dibahas secara luas dalam berbagai buku teks dan sumber online.

Terima Kasih!

Semoga presentasi ini memberikan pemahaman yang jelas mengenai implementasi dan aplikasi struktur data Stack dalam Python.

Dibuat oleh Ammar