Percobaan 1

Pada percobaan ini merupakan implementasi dasar dari stack dalam bahasa python. Dalam percobaaan 1. kita mengunakan fungsi list data array yang merupakan plugin bawaan dari python sehingga dapat memudahkan bagi setiap orang yang ingin mempelajari stack secara mendasar Pada percobaan 2 memuat fungsi kode , antara lain :

- stack.append('a') berfungsi untuk menambahkan huruf a dalam list stack yang dibuat
- stack.pop() berfungsi untuk menghapus dan mengembalikan elemen teratas (atau elemen terakhir) dari stack

Percobaan 2

Pada percobaan ini merupakan implementasi dasar dari stack dalam bahasa python. Dalam percobaaan 2. kita mengunakan collection dari deque yang dapat memberikan kelebihan dari implementasi stack , seperti :

- Memberikan fleksibilitas kinerja yang baik dalam operasi penambahan dan penghapusan elemen
- Kode menjadi mudah di mengerti dan di pahami
- Penulisan kode menjadi lebih singkat dan jelas

Pada percobaan 2 memuat fungsi kode , antara lain :

- stack.append('a') berfungsi untuk menambahkan huruf a dalam list stack yang dibuat
- stack.pop() berfungsi untuk menghapus dan mengembalikan elemen teratas (atau elemen terakhir) dari stack

Percobaan 3

Pada percobaan ini kita mengunakan collection queue.LifeQueue.LifoQueue merupakan gambaran tumpukan yang kamu temui di kehidupan nyata, di mana barang yang terakhir dimasukkan adalah yang pertama diambil. Keuntungan dari lifequeue ialah Thread-Safe:

- LifoQueue didesain untuk keamanan penggunaan bersama antar-thread (thread-safe). Ini membuatnya aman digunakan dalam aplikasi multi-threading, jika dibutuhkan.
- Kemudahan Penggunaan: Fungsi-fungsi standar seperti put() dan get() memudahkan penambahan dan penghapusan elemen dari tumpukan.
- Fungsi Size dan Status: qsize(), full(), dan empty() memberikan informasi tentang ukuran dan status tumpukan, yang bisa berguna dalam kontrol alur program.

Penjelasan Kode dari percobaan 3

- stack = LifoQueue(maxsize=3): Membuat objek LifoQueue dengan batasan ukuran maksimum sebesar 3.
- print(stack.qsize()): Menampilkan ukuran tumpukan saat ini, awalnya 0.
- stack.put('a'), stack.put('b'), stack.put('c'): Menambahkan elemen 'a', 'b', dan 'c' ke dalam tumpukan.
- print("Full: ", stack.full()): Menampilkan apakah tumpukan penuh atau tidak.
- print("Size: ", stack.qsize()): Menampilkan ukuran tumpukan setelah menambahkan elemen.
- print('\nElement yang dihapus dari stack:'): Mencetak pesan sebelum mengeluarkan elemen dari tumpukan.
- print(stack.get()), print(stack.get()), print(stack.get()): Mengeluarkan elemen dari tumpukan (LIFO).

- print("\nEmpty: ", stack.empty()): Menampilkan apakah tumpukan kosong atau tidak setelah mengeluarkan elemen.

Penjelasan Tugas Implementasi

Definisi Kelas KeranjangBelanja

- __init__(self): Konstruktor kelas yang membuat objek keranjang belanja dengan atribut barang yang awalnya kosong.
- tambah_barang(self, nama_barang): Metode untuk menambahkan barang ke dalam keranjang belanja.
- hapus_barang_terakhir(self): Metode untuk menghapus barang terakhir dari keranjang belanja.
- lihat_keranjang(self): Metode untuk menampilkan isi keranjang belanja.

Contoh Penggunaan Kelas

- Membuat objek keranjang dari kelas KeranjangBelanja.
- Menggunakan loop while True untuk membuat menu interaktif yang terus berulang.
- Meminta pengguna untuk memilih aksi dengan memasukkan angka 1, 2, 3, atau 4.
- Pilihan 1: Meminta nama barang dari pengguna dan menambahkannya ke keranjang.
- Pilihan 2: Menghapus barang terakhir dari keranjang.
- Pilihan 3: Menampilkan isi keranjang.
- Pilihan 4: Keluar dari program.

Dalam kasus kode untuk keranjang belanja, pendekatan menggunakan stack (tumpukan) digunakan untuk melacak barang-barang yang ditambahkan ke dalam keranjang dan

menghapus barang terakhir yang ditambahkan. Di bawah ini adalah penjelasan bagaimana stack digunakan dalam konteks ini:

1. Stack diwakili oleh list (self.barang) yang digunakan untuk menyimpan barang-barang dalam keranjang.
2. Setiap kali fungsi tambah_barang dipanggil, nama barang ditambahkan ke akhir list (representasi stack).
3. Fungsi hapus_barang_terakhir menghapus elemen terakhir dari list (elemen terakhir yang ditambahkan ke dalam keranjang).
4. Fungsi lihat_keranjang mencetak isi keranjang dengan menelusuri list dari elemen terakhir ke elemen pertama (menerapkan prinsip LIFO)
5. Dalam loop, pengguna diberikan opsi untuk menambah barang, menghapus barang terakhir, melihat isi keranjang, atau keluar dari program. Setiap kali barang ditambahkan, itu menjadi elemen terakhir di dalam stack. Ketika pengguna memilih untuk menghapus barang terakhir, elemen terakhir dihapus dari stack (LIFO).