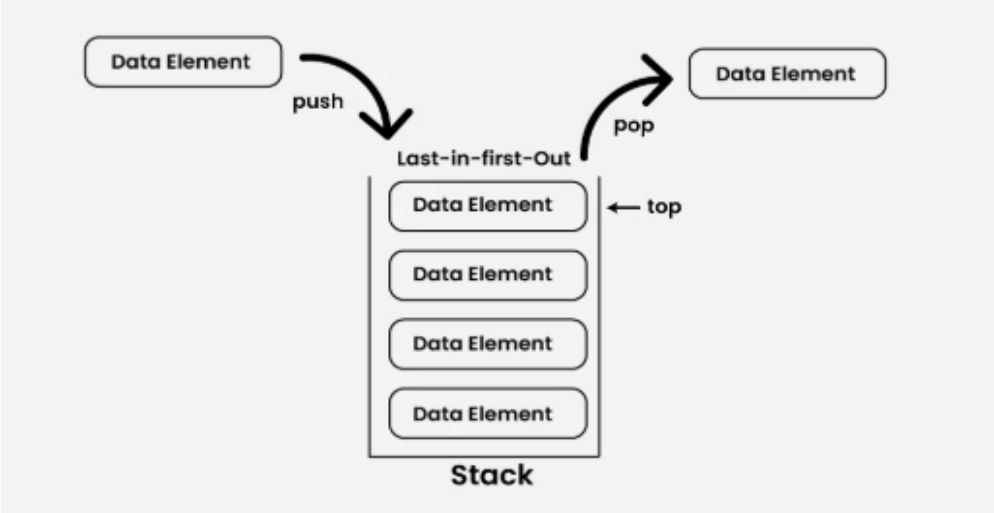
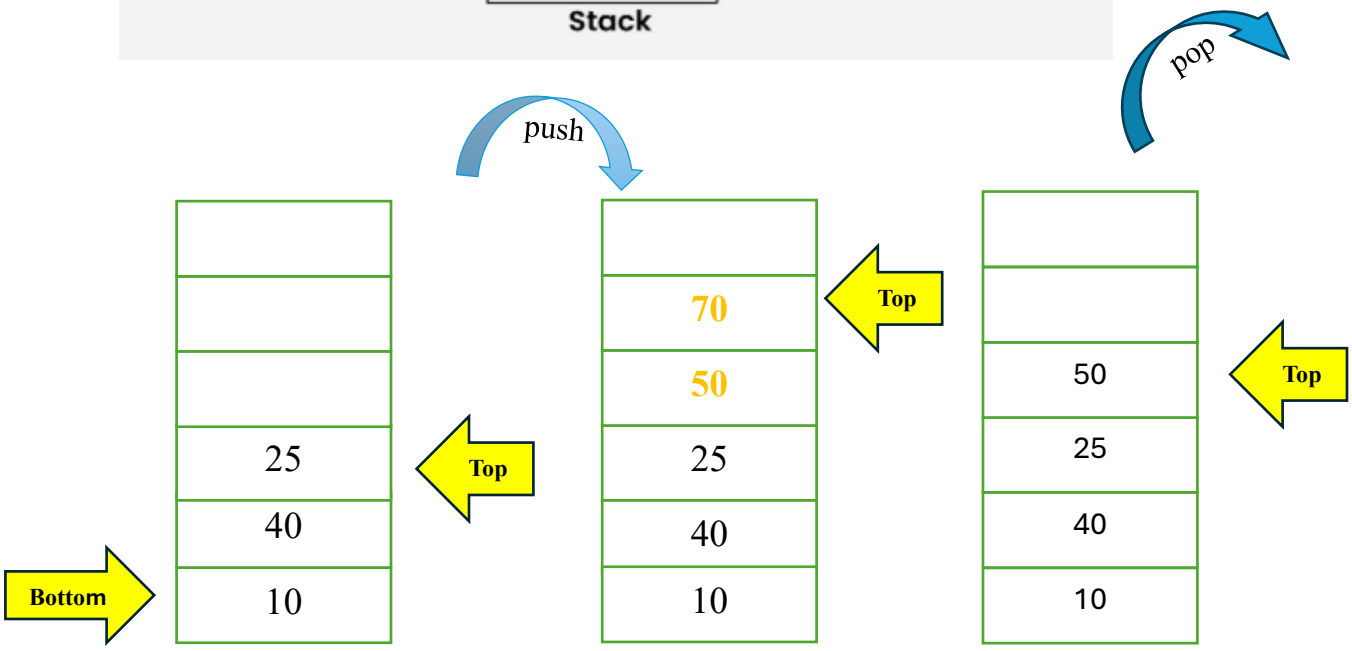
BÀI 1

KIỂU DỮ LIỆU NGĂN XẾP

1. Cơ chế hoạt động

Ngăn xếp hay còn gọi là Stack là cấu trúc dữ liệu tuyến tính tuân theo nguyên tắc LIFO (Last In First Out), do đó các phần tử được thêm vào sau cùng thì sẽ là phần tử lấy ra đầu tiên.

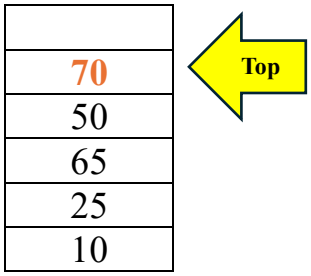




2. Biểu diễn ngăn xếp bằng mảng một chiều

Biểu diễn ngăn xếp bằng mảng một chiều là cách tổ chức và quản lý dữ liệu của ngăn xếp (stack) bằng cách sử dụng một mảng một chiều. Trong Python, các

thao tác cơ bản của danh sách tương tự như các thao tác của ngăn xếp. Do đó, ngăn xếp được biểu diễn bằng kiểu list của Python.



Ngăn xếp	**10**	**25**	**65**	**50**	**70**	
Chỉ số mảng	0	1	2	3	4	

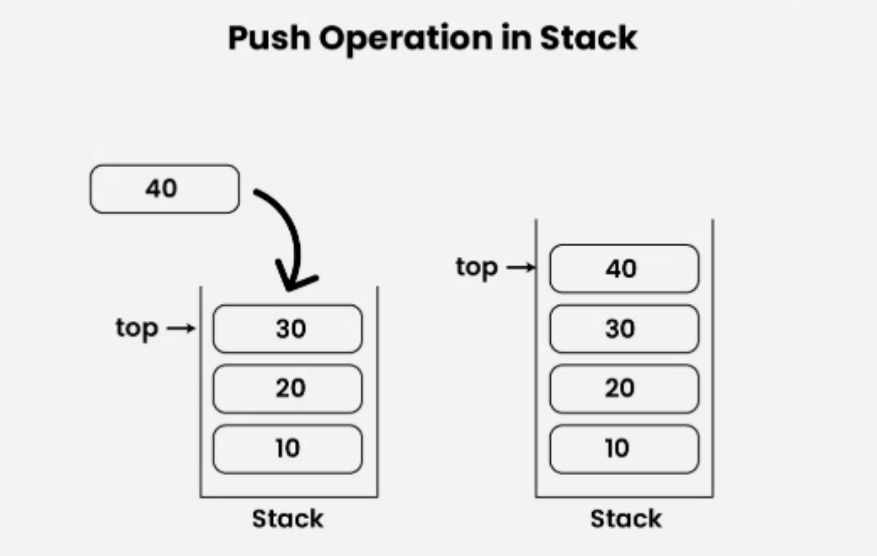
3. Các thao tác cơ bản của ngăn xếp

Khi thao tác với ngăn xếp, chúng ta thường thực hiện các thao tác nhất định:

- push() : Để chèn một phần tử vào ngăn xếp
- pop() : Để xóa một phần tử khỏi ngăn xếp
- top() : Trả về phần tử trên cùng của ngăn xếp.
- isEmpty() : Trả về true nếu ngăn xếp rỗng, nếu không trả về false.
- isFull() : Trả về true nếu ngăn xếp đầy, nếu không trả về false.

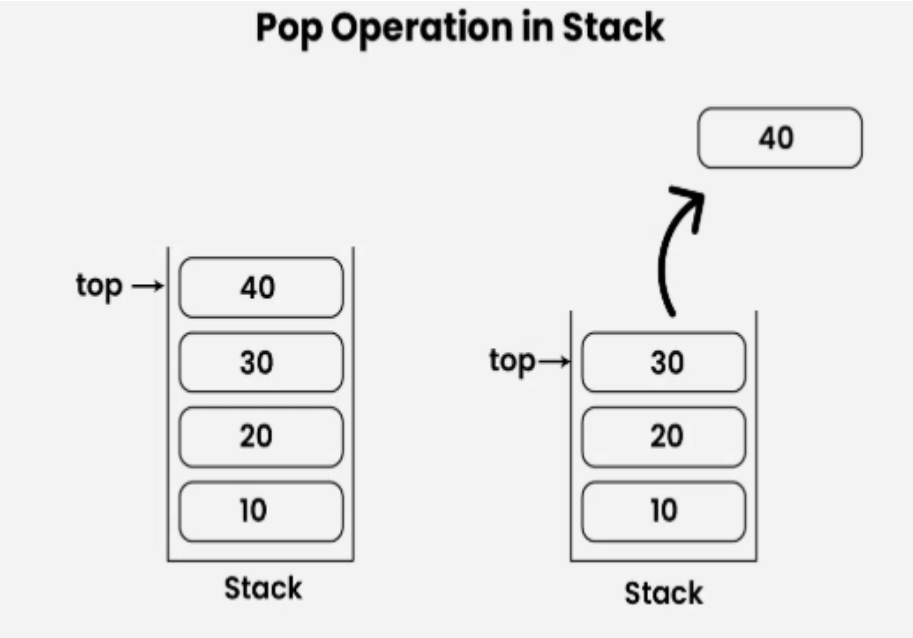
Thuật toán Push trong cấu trúc dữ liệu ngăn xếp

- Trước khi đẩy phần tử vào ngăn xếp, chúng ta kiểm tra xem ngăn xếp đã **đầy** hay chưa .
- Nếu ngăn xếp đầy **(top == capacity-1)** thì **ngăn xếp sẽ tràn** và chúng ta không thể chèn phần tử vào ngăn xếp.
- Nếu không, chúng ta tăng giá trị của top lên 1 **(top = top + 1)** và giá trị mới được chèn vào **vị trí top** .
- Các phần tử có thể được đẩy vào ngăn xếp cho đến khi đạt đến **sức chứa** của ngăn xếp.



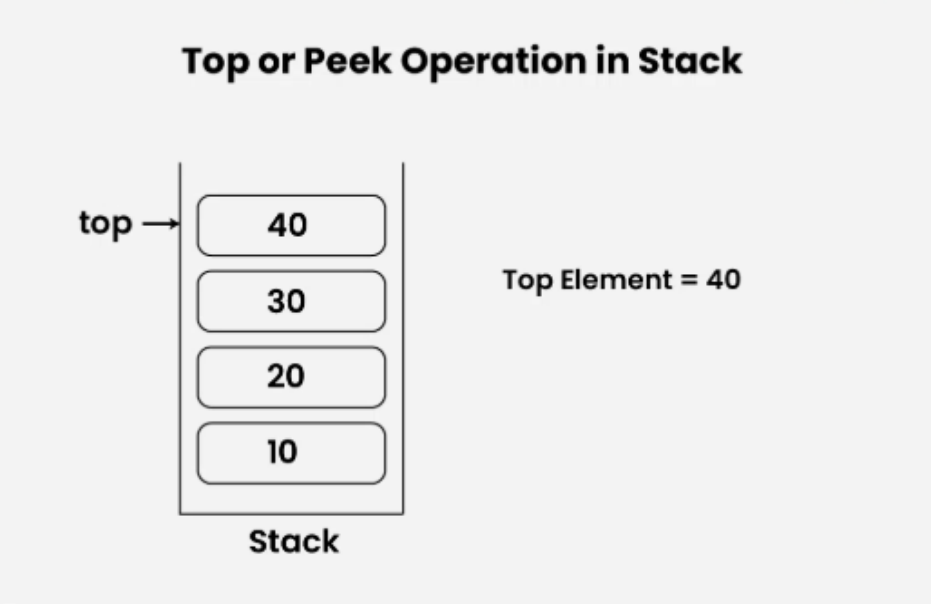
Thuật toán Pop trong cấu trúc dữ liệu ngăn xếp

- Trước khi lấy phần tử ra khỏi ngăn xếp, chúng ta kiểm tra xem ngăn xếp có **rỗng** hay không .
- Nếu ngăn xếp rỗng (top == -1), thì **Stack sẽ tràn** và chúng ta không thể xóa bất kỳ phần tử nào khỏi ngăn xếp.
- Nếu không, chúng ta lưu trữ giá trị ở top, giảm giá trị của top đi 1 **(top = top – 1)** và trả về giá trị top đã lưu trữ.



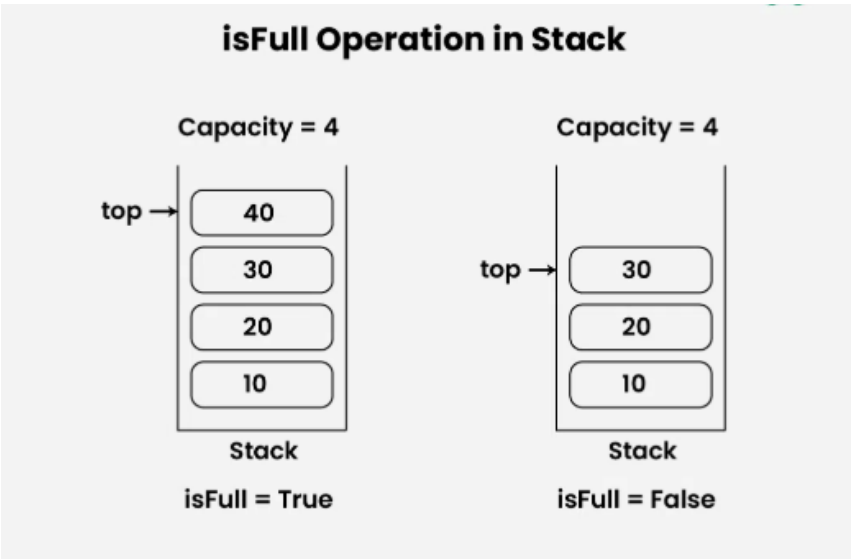
Thuật toán Top trong cấu trúc dữ liệu ngăn xếp

- Trước khi trả về phần tử trên cùng của ngăn xếp, chúng ta kiểm tra xem ngăn xếp có rỗng không.
- Nếu ngăn xếp rỗng (top == -1), chúng ta chỉ cần in ra "Ngăn xếp rỗng".
- Nếu không, chúng ta sẽ trả về phần tử được lưu trữ tại **index = top** .



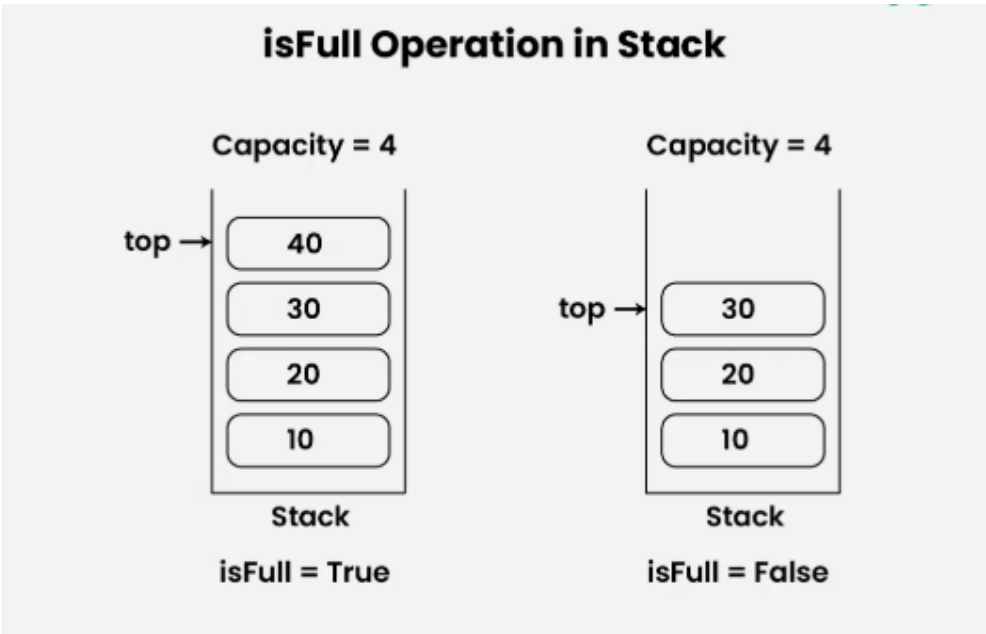
Thuật toán isEmpty trong cấu trúc dữ liệu ngăn xếp

- Kiểm tra giá trị của **top** trong ngăn xếp.
- Nếu **(top == -1)** thì ngăn xếp **trống** nên trả về **true** .
- Nếu không, ngăn xếp không rỗng nên trả về **false** .



Thuật toán isFull trong cấu trúc dữ liệu ngăn xếp

- Kiểm tra giá trị của **top** trong ngăn xếp.
- Nếu **(top == capacity-1),** thì ngăn xếp đã **đầy** nên trả về **true** .
- Nếu không, ngăn xếp chưa đầy nên trả về **false** .



4. Cài đặt ngăn xếp bằng ngôn ngữ lập trình Python

```
def push(stack, value):
    stack.append(value)
    print(f"{value} đã được thêm vào stack.")
def pop(stack):
    if is_empty(stack):
        print("Stack rỗng, không thể pop phần tử.")
    else:
        value = stack.pop()
        print(f"Đã xóa {value} khỏi stack.")
        return value
def top(stack):
    if is_empty(stack):
        print("Stack rỗng, không có phần tử trên cùng.")
    else:
        print(f"Phần tử trên cùng của stack là: {stack[-1]}")
        return stack[-1]
```

```
def is_empty(stack):
    return len(stack) == 0
def display(stack):
    if is_empty(stack):
        print("Stack rỗng.")
    else:
        print("Stack hiện tại:", stack)
def main():
    stack = []
    while True:
        print("Chọn thao tác:")
        print("1. Thêm phần tử (push)")
        print("2. Xóa phần tử (pop)")
        print("3. Xem phần tử trên cùng (top)")
        print("4. Hiển thị stack")
        print("5. Thoát")
        choice = input("Nhập lựa chọn (1-5): ")
        if choice == "1":
            value = input("Nhập giá trị cần thêm: ")
            push(stack, value)
        elif choice == "2":
            pop(stack)
        elif choice == "3":
            peek(stack)
        elif choice == "4":
            display(stack)
        elif choice == "5":
            print("Kết thúc chương trình.")
            break
        else:
            print("Lựa chọn không hợp lệ. Vui lòng thử lại.")
if __name__ == "__main__":
    main()
```

5. **Một số ví dụ mẫu**

Bài 1. Cho mảng Array = [4, 7, 16, 23, 9, 25, 10], hãy viết chương trình in ra ngăn xếp gồm các phần tử là số chính phương của mảng Array

```
import math
def is_perfect_square(num):
    if num < 0:
        return False
    root = int(math.sqrt(num))
    return root * root == num
def create_stack_from_array(array):
    stack = []
    for num in array:
        if is_perfect_square(num):
            stack.append(num)  # Thêm phần tử vào ngăn xếp
    return stack
array = [4, 7, 16, 23, 9, 25, 10]
stack = create_stack_from_array(array)
print("Ngăn xếp các số chính phương:", stack)
```

Bài 2. Cho mảng A = [3, 6, 8, 15], B = [4, 9, 12, 20]. Hãy viết chương trình in ra ngăn xếp và phần tử đầu tiên được thêm vào ngăn xếp, biết ngăn xếp gồm các phần tử là số chẵn của mảng A và mảng B

```
def create_even_stack(array1, array2):
    stack = []
    for num in array1 + array2:  # Gộp hai mảng lại và duyệt từng phần tử
        if num % 2 == 0:        # Kiểm tra số chẵn
            stack.append(num)   # Thêm phần tử vào ngăn xếp
    return stack
array1 = [3, 6, 8, 15]
array2 = [4, 9, 12, 20]
even_stack = create_even_stack(array1, array2)
first_element_in_stack = even_stack[0] if even_stack else None
print("Ngăn xếp các số chẵn từ hai mảng:", even_stack)
print("Phần tử đầu tiên của ngăn xếp:", first_element_in_stack)
```