**STACK ADT USING ARRAY**

class Stack:

def \_\_init\_\_(self, capacity):

"""Initialize the stack with a specific capacity."""

self.capacity = capacity

self.stack = [None] \* capacity # Fixed size array to store stack elements

self.top = -1 # Represents the index of the top element

def push(self, item):

"""Add an item to the top of the stack."""

if self.is\_full():

raise Exception("Stack overflow: cannot push, stack is full")

self.top += 1

self.stack[self.top] = item

print(f"Pushed {item} onto stack.")

def pop(self):

"""Remove and return the top item from the stack."""

if self.is\_empty():

raise Exception("Stack underflow: cannot pop, stack is empty")

item = self.stack[self.top]

self.stack[self.top] = None # Optional: Clear the reference

self.top -= 1

print(f"Popped {item} from stack.")

return item

def peek(self):

"""Return the top item from the stack without removing it."""

if self.is\_empty():

raise Exception("Stack is empty: cannot peek")

return self.stack[self.top]

def is\_empty(self):

"""Check if the stack is empty."""

return self.top == -1

def is\_full(self):

"""Check if the stack is full."""

return self.top == self.capacity - 1

def size(self):

"""Return the number of items in the stack."""

return self.top + 1

def display(self):

"""Display the current state of the stack."""

if self.is\_empty():

print("Stack is empty.")

else:

print("Stack from top to bottom:")

for i in range(self.top, -1, -1):

print(self.stack[i])

# Example usage:

if \_\_name\_\_ == "\_\_main\_\_":

stack\_capacity = 5

stack = Stack(stack\_capacity)

# Push elements onto the stack

stack.push(10)

stack.push(20)

stack.push(30)

stack.push(40)

stack.push(50)

# Display the stack

stack.display()

# Peek at the top element

print(f"Top element is: {stack.peek()}")

# Pop elements from the stack

stack.pop()

stack.pop()

# Display the stack again

stack.display()

# Check if the stack is empty

print(f"Is the stack empty? {'Yes' if stack.is\_empty() else 'No'}")

# Get the current size of the stack

print(f"Current stack size: {stack.size()}")