```
# Step 1: initialize an empty stack
Stack = []                # stack = []

# Step 2: push four items (growth)
Stack. Append (“A”)          # push A -> stack becomes [‘A’]          (bottom->top)
Print (“After push A:”, stack)                # after  pushing item “A”

Stack. Append (“B”)          # push B -> stack becomes [‘A’,’B’]
Print ("After push B:’ , stack)

Stack. Append (“C”)          # push C -> stack becomes [‘A’,’B’,’C’]
Print ("After push C:”, stack)

stack.append (“D”)          # push D -> stack becomes [‘A’,’B’,’C’,’D’]        # (top item is 'D')
Print (“After push D:”, stack)

# Step 3: pop twice (shrink) second time shrink the item
popped1 = stack. Pop ()       # pop -> removes 'D' (the top), popped1 = 'D' because it is in rear or on the top
Print ("Popped 1:”, popped1, “-> stack now:”, stack)    # show the remained items and removed one
# after removing ‘D’ remained item on the top  is ‘C’
popped2 = stack. Pop ()       # pop -> removes 'C' (new top), popped2 = 'C'
Print ("Popped 2:", popped2, "-> stack now:", stack)     # we remains items where B is on the top

# Step 4: final state after popping elements or items
Print ("Final stack (bottom->top):", stack)  # expected ['A','B']
```

Summary

- Start: `[]` (empty)
- After `push A`: `['A']` (top = A)
- After `push B`: `['A', 'B']` (top = B)
- After `push C`: `['A', 'B', 'C']` (top = C)
- After `push D`: `['A', 'B', 'C', 'D']` (top = D) # **stack is at its largest**
- After `pop()` → popped `'D'`, stack becomes `['A', 'B', 'C']`
- After second `pop()` → popped `'C'`, stack becomes `['A', 'B']` #**final stack**