# Bài 2: Kiểu dữ liệu ngăn xếp

**Giải Chuyên đề Tin học 12 Bài 2: Kiểu dữ liệu ngăn xếp**  
**Khởi động trang 9 Chuyên đề Tin học 12**: Theo em những kiểu dữ liệu sau có thể được dùng để thiết lập dữ liệu ngăn xếp không? Tại sao?  
a) Sử dụng kiểu mảng có chiều dài cố định N, với số tự nhiên N khá lớn.  
b) Sử dụng kiểu dữ liệu danh sách liên kết (đã học ở chương trình Tin học 11-Định hướng Khoa học máy tính).  
c) Sử dụng kiểu dữ liệu list của Python.  
**Lời giải:**  
Những kiểu dữ liệu sau có thể được dùng để thiết lập dữ liệu ngăn xếp:  
a) Sử dụng kiểu mảng có chiều dài cố định N, với số tự nhiên N khá lớn.  
c) Sử dụng kiểu dữ liệu list của Python.  
**1. Biểu diễn ngăn xếp băng mảng 1 chiều**  
**Hoạt động 1 trang 9 Chuyên đề Tin học 12**: Dùng kiểu dữ liệu mảng để biểu diễn ngăn xếp  
Quan sát, trao đổi, thảo luận để tìm hiểu cách biểu diễn ngăn xếp bằng mảng 1 chiều. Trả lời các câu hỏi sau:  
- Có thể biểu diễn ngăn xếp bằng mảng 1 chiều được không?  
- Cần có các biến nào để thực hiện các phép toán cơ bản trên ngăn xếp?  
**Lời giải:**  
- Có thể biểu diễn ngăn xếp bằng mảng 1 chiều.  
- Cần có các biến nào để thực hiện các phép toán cơ bản trên ngăn xếp:  
+ Biến S là danh sách rỗng.  
+ Biến x là phần tử của mảng.  
**Câu hỏi 1 trang 10 Chuyên đề Tin học 12**: Dãy các số 1, 2, 3, 4, 5, 6 lần lượt được đưa vào ngăn xếp S bằng lệnh push(). Người thực hiện làm như sau: Cứ thực hiện push(S,x) hai lần thì lại pop(S) một lần. Dãy số kết quả thu được bao gồm những số nào?  
**Lời giải:**  
Dãy các số 1, 2, 3, 4, 5, 6 lần lượt được đưa vào ngăn xếp S bằng lệnh push(). Người thực hiện làm như sau: Cứ thực hiện push(S,x) hai lần thì lại pop(S) một lần. Dãy số kết quả thu được bao gồm những số: 1,3,5.  
**Câu hỏi 2 trang 10 Chuyên đề Tin học 12**: Giả sử chúng ta lần lượt thực hiện dãy các lệnh sau (ngăn xếp S ban đầu là rỗng). push(S,1); push(S,2); pop(S); push(S,3); pop(S); pop(S).  
Dãy các phần tử lần lượt được đưa ra khỏi ngăn xếp là các số nào?  
**Lời giải:**  
Giả sử chúng ta lần lượt thực hiện dãy các lệnh sau (ngăn xếp S ban đầu là rỗng). push(S,1); push(S,2); pop(S); push(S,3); pop(S); pop(S).  
Dãy các phần tử lần lượt được đưa ra khỏi ngăn xếp là các số: 2,3,1.  
**2. Các phép toán của kiểu dữ liệu ngăn xếp**  
**Hoạt động 2 trang 10 Chuyên đề Tin học 12**: Tìm hiểu các hàm cơ bản của ngăn xếp  
Đọc, trao đổi để biết các hàm cơ bản của ngăn xếp được cài đặt bằng danh sách (kiểu list của Python).  
**Lời giải:**  
- Hàm Stack() dùng để tạo ngăn xếp rỗng.  
- Hàm Push(S,x) dùng để thêm x vào đỉnh của ngăn xếp, thêm x vào cuối danh sách bằng S bằng hàm append():  
- Hàm Pop dùng để lấy ra phần tử tại đỉnh của top.  
- Hàm Top trả về phần tử tại đỉnh của Top.  
**Câu hỏi 1 trang 12 Chuyên đề Tin học 12**: Sửa lại hàm pop(S) và top(S) trong hoạt động trên như sau: Nếu ngăn xếp rỗng thì thông báo: “Ngăn xếp rỗng không thể thực hiện được lệnh này”.  
**Lời giải:**  
Sửa lại hàm pop(S):  
def pop(S):  
 if isEmptyStack(S):  
 raise ValueError(“Ngăn xếp rỗng không thể thực hiện được lệnh này”)  
 else:  
 return S.pop()  
Sửa lại hàm top(S):  
def top(S):  
 if isEmptyStack(S):  
 raise ValueError(“Ngăn xếp rỗng không thể thực hiện được lệnh này”)  
 else:  
 return S[len(S)-1]  
**Câu hỏi 2 trang 12 Chuyên đề Tin học 12**: Vì sao các hàm cơ bản trên ngăn xếp S được cài đặt bằng danh sách (kiểu list của Python) không cần sử dụng biến top và biến bottom?  
**Lời giải:**  
Vì đỉnh (top) của ngăn xếp S luôn là phần tử cuối cùng của danh sách S. Do vậy không cần biến top.  
Vì đáy (bottom) của ngăn xếp S luôn là phần tử đầu tiên của danh sách S. Do vậy không cần biến bottom.  
**Luyện tập 1 trang 12 Chuyên đề Tin học 12**: Viết hàm length(S) trả về số phần tử của ngăn xếp S.  
**Lời giải:**  
def length(S):  
 count = 0   
while S:   
S.pop()   
count += 1  
return count  
**Luyện tập 2 trang 12 Chuyên đề Tin học 12**: Giả sử dãy số ban đầu là 2, 7, 6, 1 và S là ngăn xếp rỗng. Chúng ta lần lượt thực hiện các thao tác push(S,x), pop(S) với dãy số trên từ trái sang phải. Kết quả các số lần lượt được đưa ra khỏi ngăn xếp là 6, 7, 1, 2. Hãy viết các lệnh theo trình tự đã thực hiện.  
**Lời giải:**  
Các lệng theo trình tự là: push(S,2); push(S,7); pop(S); push(S,6); pop(S); push(S,1); push(S,7); push(S,6); pop(S); pop(S); pop(S); pop(S).  
**Vận dụng 1 trang 12 Chuyên đề Tin học 12**: Xâu kí tự được gọi là biểu thức nếu nó là rỗng hoặc chỉ chứa các kí tự “(“ và “)”  
Ví dụ: "((()())())". Xâu biểu thức được gọi là đúng nếu vị trí các dáu ngoặc được sắp xếp hợp lí theo tự nhiên. Ví dụ các xâu sau là biểu thức đúng:   
()  
(()())  
Ví dụ các xâu biểu thức sau là sai:   
((())  
))()()  
Có thể định nghĩa khái niệm biểu thức đúng bằng đệ quy như sau:   
- Xâu rỗng là đúng.   
- Nếu xâu A, B đúng thì xâu AB đúng.   
- Nếu xâu A là đúng thì xâu (A) đúng.   
Cho trước xâu biểu thức A, viết chương trình kiểm tra xem A có là biểu thức đúng hay không. Yêu cầu sử dụng kiểu dữ liệu ngăn xếp.  
**Lời giải:**  
def is\_valid\_expression(expression):  
 # Khởi tạo ngăn xếp rỗng  
 stack = []  
 # Tạo một từ điển để ghép các dấu ngoặc đóng với dấu ngoặc mở tương ứng  
 matching\_parentheses = {')': '(', '}': '{', ']': '['}  
   
 # Duyệt qua từng ký tự trong biểu thức  
 for char in expression:  
 if char in matching\_parentheses.values():  
 stack.append(char)  
 elif char in matching\_parentheses.keys():  
 if not stack or stack.pop() != matching\_parentheses[char]:  
 return False  
 return not stack  
**Vận dụng 2 trang 12 Chuyên đề Tin học 12**: Ngăn xếp S được cài đặt bằng mảng T có N phân tử, phần tử đầu tiên có chỉ số 0. Hãy viết các hàm cơ bản trên ngăn xếp S.  
Lưu ý:   
- Biến topldx cho biết đỉnh top của ngăn xếp.   
- Ngăn xếp là rỗng thì topldx = -1. Khi topldx = N-1 thì ngăn xếp bị tràn (overflow), không thể thêm phần tử mới vào ngăn xếp S.   
- Viết hàm stackOverflow(S) trả về True nếu ngăn xếp S bị tràn; ngược lại trả về False. Hàm stackOverflow(S) sẽ tạo ngoại lệ ValueError(). Sử dụng hàm stackOverflow(S) để kiểm tra ngăn xếp S chưa bị tràn trước khi gọi hàm push(S, x)  
**Lời giải:**  
def push(S, x):  
 try:  
 if not S.stackOverflow():  
 S.topldx += 1  
 S.T[S.topldx] = x  
 except ValueError as e:  
 print(e)  
def pop(S):  
 if S.isEmpty():  
 raise IndexError(“Ngăn xếp rỗng!")  
 item = S.T[S.topldx]  
 S.T[S.topldx] = None   
 S.topldx -= 1  
 return item  
def isEmptyStack(S):  
 return S.topldx == -1  
def isFull(S):  
 return S.topldx == self.size - 1