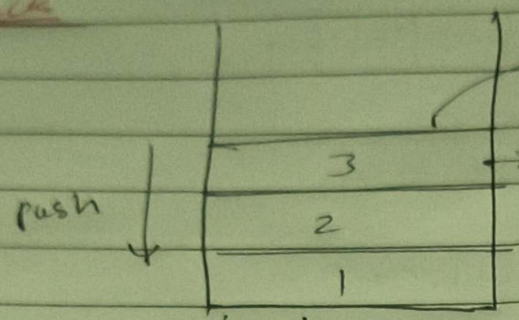
23/12/25
Day 32

Stack Intro & Implementation - Python

Stack



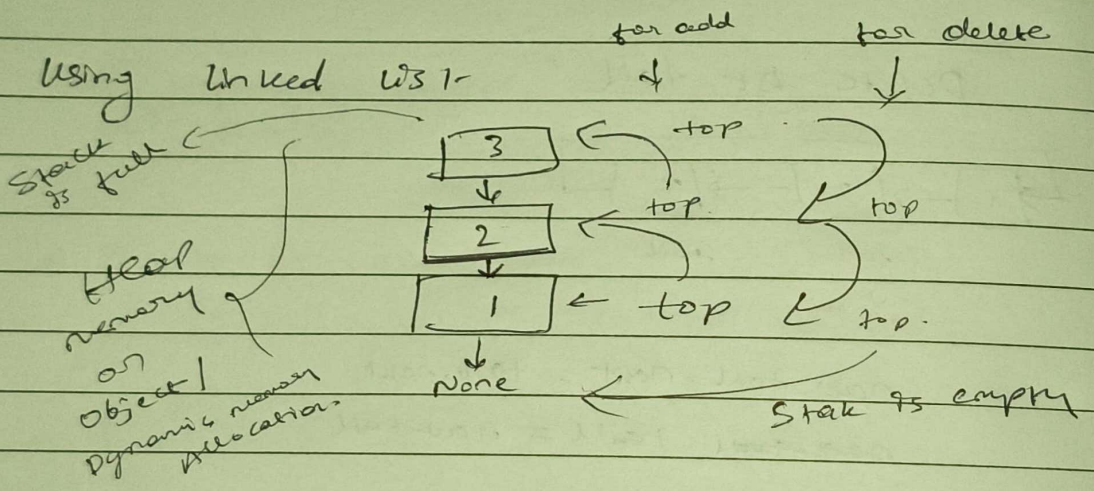
→ An Array

↳ linear

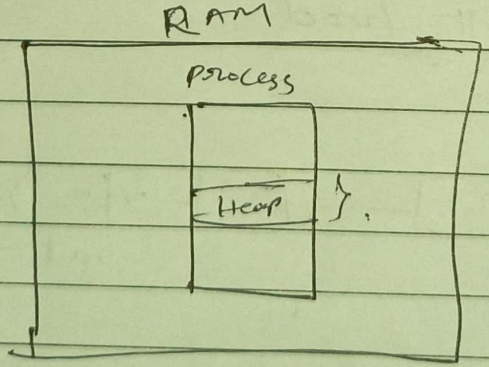
Array:- Fixed size

Also using Linked list. We can make Stack

Using Linked list:-



How many memory we can create



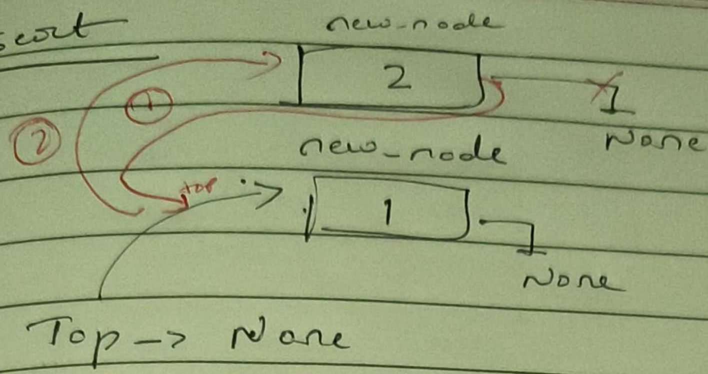
Artificial limit

max = 100

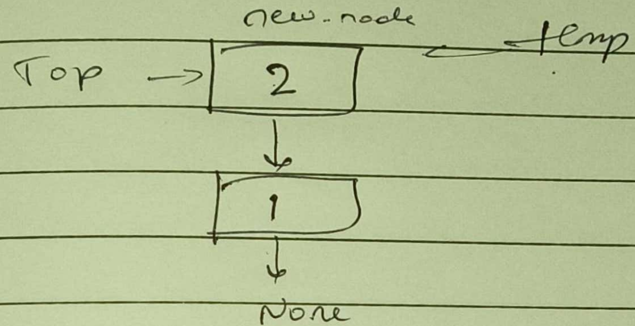
push -> counter += 1

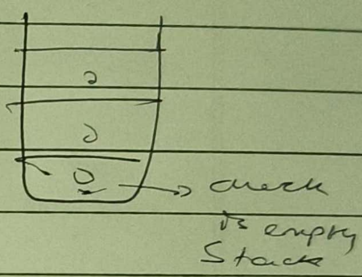
Pop -> counter -= 1

* Insert



* pop / delete :





```python
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None

class Stack:

    def __init__(self):
        self.top = None


    def push(self,data):
        new_node = Node(data)

        new_node.next = self.top
        self.top = new_node

    def print_stack(self):
        current_node = self.top

        while current_node:
            print(current_node.data, end = " -> ")
            current_node = current_node.next

        print("Node")
```

```
33  class Stack:

47      def pop(self) -> int:
48          # Case 1: Stack is empty
49          if (self.top == None):
50              print("Stack is empty! can't perform pop operation")
51              return -100
52
53          data = self.top.data
54          self.top = self.top.next
55          self.count = self.count - 1
56          print(f"Popped the item from stack {data}")
57          return data
58
59      def peek(self) -> int:
60          # Case 1: Stack is empty
61          if (self.top == None):
62              print("Stack is empty! can't perform pop operation")
63              return -100
64
65          return self.top.data
66
67      def get_count(self) -> int:
68          print(f"There are {self.count} items in the stack")
69          return self.count
70
71      def print_all_values(self):
72          if (self.top == None):
73              print("Stack is empty")
74              return
```

```
33  class Stack:
71      def print_all_values(self):
72          if (self.top == None):
73              print("Stack is empty")
74              return
75
76          current_node = self.top
77          print("Elements in the stack are as below")
78          while( current_node is not None):
79              print(f" {current_node.data} ")
80              current_node = current_node.next
81
82  if __name__ == "__main__":
83
84      stack = Stack()
85
86      stack.pop()
87      count_items = stack.get_count()
88      stack.print_all_values()
89
90      stack.push(1)
91      stack.get_count()
92      stack.print_all_values()
93      stack.pop()
94
95      stack.push(1)
96      stack.push(2)
97      stack.push(3)
98      stack.push(4)
99      stack.push(5)
```