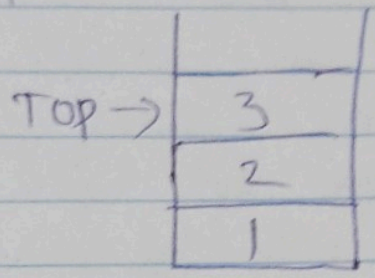
Stack

Theory – A stack is a linear data structre that stores items in a last-In/First-Out(LIFO) or First-In/last-out (FILO). In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.

Representation-



Operations of stack :-

1) push-

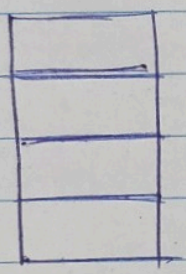
Algorithm:-

```
begin
  if stack.is full
        return
  end if
  else
     increment top
     stack[top]assign value
  end else
end procedure
```

Procedure :-

* After intializing and calling the class stack an empty list will be created and a top value - 1
* In stack push means to insert the data into the stack. Since we are using arrays (or) list to insert the stack. since we are going to use append function in python.
* To push data we are going to send to the function.
* Let us push values 1,2,3,4 to stack.
* It will follow these steps -

1) self.max will be 4 in this case



top ← -1

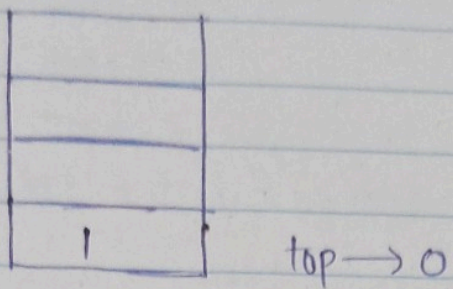
2) First we will always check if top+1 >= max which will give us if stack is full or not. if stack is full and still try to push element we need to get overflow. Hence

```
if self.top+1 >= self.max
    print("overflow")
```

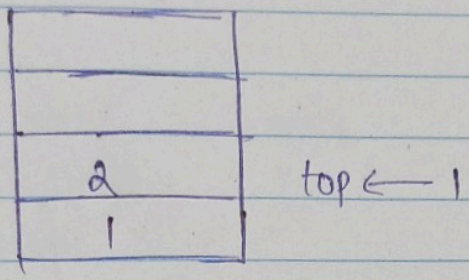
3) Since the stack is empty we will follow these steps -

```
else:
    self.top ← self.top+1
    self.stack.append(d)
```

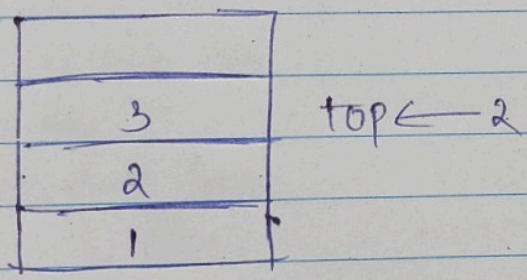
(i) Value 1 will be passed



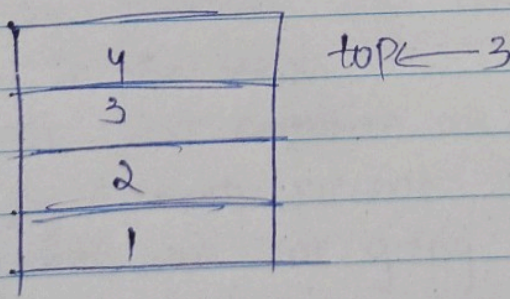
(ii) Value 2 will be passed



(iii) Value 3 will be passed



(iv) Value 4 will be passed



* Now let us try to push element 5 -

self.max = 4

top + 1 = 4

Since top + 1 >= self.max we will get overflow and the element will not be inserted

2) Pop -

Algorithm :-

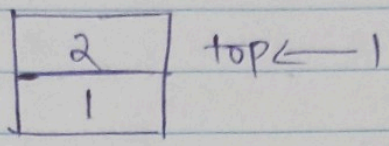
```
begin
    if stack is empty
        return
    end if
    else
        store value of stack[top]
        decrement top
        return value
    end else
end procedure
```

Procedure :-

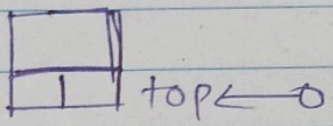
* In pop function we are going to delete the last element inserted in the stack.
* so the first condition we are going to check if there are any elements present in the stack. If there are no elements we are going to print underflow

Step 1- if self.top is -1:
print("underflow")

*Now let us take an existing stack-



*To delete the last element we will follow these steps-
1) d ← self.stack.pop()
2) self.top ← self.top - 1



*The element will be poped out and top value will decrement. The element will be returned.

3) Peek-
Algorithm :-

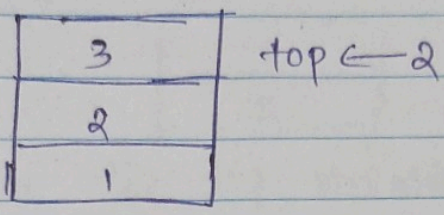
begin
return stack[top]
end procedure

Procedure :-

*In peek function we are just going to return the last element inserted in the stack.
*First we will check if stack is empty or not. If empty we need to get underflow.

Step. 1 – if self.top is – 1:
print("under flow")

*Now let us peek for this stack –

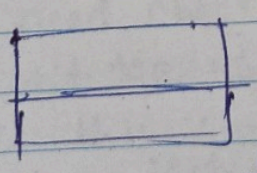


peek element ← self.stack[self.top]

4) Is Empty –
Algorithm:

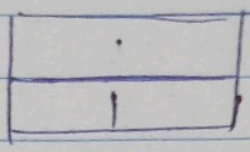
```
begin
    if: top < 0
        return true
    else: return false
end procedure
```

procedure –
*In IsEmpty function we can get to know if there are any elements inside the stack.
*Let us take an empty stack –



*So to check if the stack is empty all we need to do is check the top variable.

*If the top variable is less then 0 then the stack is empty.
*In this case the top is '-1' hence the stack is empty.
*After pushing an element ex- 1 we will get our stack RS-

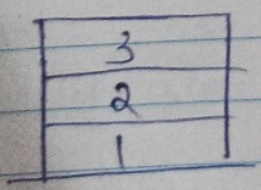


*Now since the top variable is not less than Zero we will get stack is not empty.

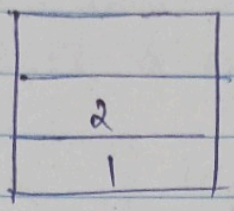
5) Is Full -
Algorithm:-

```
begin
    if top+1>=max
        return true
    else:
        return false
end procedure
```

procedure :-
*In Is Full function we can get to know if stack is completely filled with elements or not.
*For this we need check top variable is greater than or equal to max-self.top>=self.max then the stack is full.
*Let us take this stack -



* Now self.top+1 >= self.max which is 3>=3 is true the stack is full.
* Now for the same stack if we perform pop function we will get -



* Now self.top >= self.max which is 2>=3 is not true we will get the stack as not full

6) Display -

Algorithm -

```
begin
    if top < 0:
        return
    else:
        i ← self.top
        while i>=0
            print(stack[i])
            i ← i-1
end procedure :
```

Procedure:

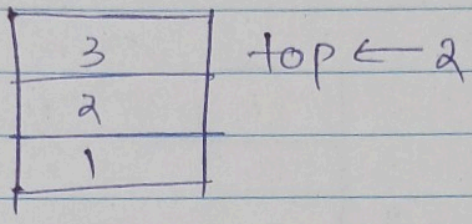
* In stack we are going to display the reverse order of the list since it follows LIFO.

* The first step is to check under flow condition-

Step-1

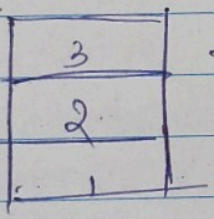
```
if self.top<0:
    print("under Flow")
```

* Now let us take a existing stack-



* Since Now the top is not less than zero it will not print underflow.
* Now to print this stack we will follow these steps-

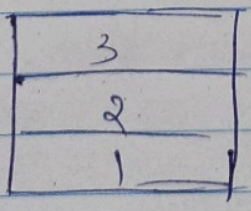
1) i ← self.top



2) while i>=0:

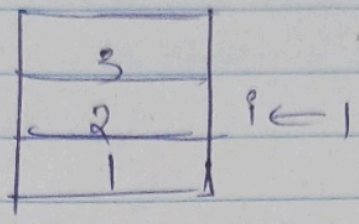
```
    print(self.stack[i])
    i ← i-1
```

1st Iteration-



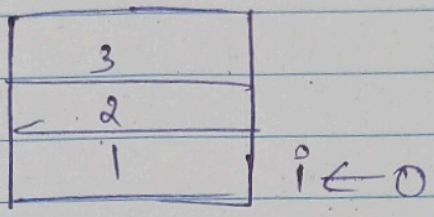
value -3 will be printed
and i ← i-1 => i ← 1

II nd Iteration –



Value – 2 will be printed
and i ← i-1 => i ← 0

III rd Iteration –



value – 1 will be printed
and i ← i-1 => i ← -1

Since i<0 while loop breaks and all elements in the stack will be printed.

Code :-

```
#include <stdio.h>
int stack[100], i, j, choice=0, n, loop=1;
void push();
void pop();
void show();
void main();
{
 printf("Enter the no. of elements in the stack");
 scanf("%d", &n);
 while(choice != 4)
{
printf("Choose one from the below option.....\n");
```

```
printf("\n1. push\n2. pop\n3. show\n4. exit");
printf("\n Enter your choice\n");
scanf("%d", &choice);
switch(choice)
{
  case 1:
  push();
  break;
  case 2:
  pop();
  break;
  case 3:
  show();
  break;
  case 4:
  printf("Exiting....");
  break;
}
 default
 {
  printf("please enter valid choice");
}
};
}
void push()
{
  int val;
  if(top==n)
  printf("\n overflow");
  else
{
```

```
printf("Enter the value");
scanf("%d",&val);
top=top+1
stack[top]=val;
}
}
Void pop()
{
if (top==-1)
printf("Under flow");
else
top=top-1;
}
void show()
{
for(i=top;i>0=0;i--)}
{
print("%d\n",stack[i]");
}
if (top==-1)
print("stack is empty");
}
```

Output -

```
Choose one from the below options....

1. push
2. pop
3. show
4. Exit
Enter your choice
3
stack is empty

Choose one from the below options....

1. push
2. pop
3. show
4. EXIT
Enter your choice
4
Existing.....
```