y3yerpp1c

July 25, 2024

1 stack implement using singly linked list

2 add at begining

in this head will be last added element

[6]:
```
class node:
    def __init__(self,data):
        self.data=data
        self.next=None
class stack:
    def __init__(self):
        self.head=None
    def isempty(self):
        if (self.head==None):
            return True
        else:
            return False
    def push(self,data):
        if(self.head==None):
            self.head=node(data)
        else:
            newnode=node(data)
            newnode.next=self.head #############
            self.head=newnode

    def pop(self):
        if(self.isempty()):
            return None
        else:
            poppednode=self.head        #for pop data
            self.head=self.head.next    #for make new head
            poppednode.next=None        #for pop next of popped element
            return poppednode.data
    def peek(self):
        if (self.isempty()):
            return None
        else:
```

```
            return self.head.data
    def display(self):
        iternode=self.head
        if (self.isempty()):
            print("underflow")
        else:
            while(iternode!=None):
                print(iternode.data,"--->",end=" ")
                iternode=iternode.next
            return



mystack=stack()
mystack.push(10)
mystack.push(20)
mystack.push(30)
mystack.push(40)
mystack.display()
print("\nTop element is ",mystack.peek())
mystack.pop()
mystack.display()
print("\nTop element is ",mystack.peek())
```

```
40 ---> 30 ---> 20 ---> 10 --->
Top element is  40
30 ---> 20 ---> 10 --->
Top element is  30
```

[ ]:

2.1 add at end

in this head will be first element

[2]:
```
## not completed

#stack implementation using linedlist(ending)
class node:
    def __init__(self,data):
        self.data=data
        self.next=None
class stack:
    def __init__(self):
        self.head=None
    def isempty(self):
        if(self.head==None):
```

```
            return True
        else:
            return False
    def push(self,data):
        if(self.head==None):
            self.head=node(data)
        else:
            newnode=node(data)
            self.head.next=newnode
            #newnode.next=self.head
            newnode.next=None
            temp=newnode
            return

    def pop(self):
        temp=self.head
        if self.isempty():
            return None
        else:
            poppednode=self.head
            self.head=temp
            poppednode.next=None
            return poppednode.data
    def display(self):
        iternode=self.head
        if self.isempty():
            print("underrflow")
        else:
            while(iternode!=None):
                print(iternode.data,"--->",end=" ")
                iternode=iternode.next
            return

mystack=stack()
mystack.push(11)
mystack.push(22)
mystack.push(33)
mystack.push(44)

mystack.display()
```

```
11 ---> 44 --->
```

3 stack using doubly linked list

[3]:
```
#stack implementation using Doublylinkedlist
class node:
    def __init__(self,data):
        self.data=data
        self.previous=None
        self.next=None
class stack:
    def __init__(self):
        self.head=None
    def isempty(self):
        if self.head==None:
            return True
        else:
            return False
    def push(self,data):
        if self.head==None:
            self.head=node(data)
        else:
            newnode=node(data)
            self.head.prev=newnode
            newnode.next=self.head
            self.head=newnode
    def pop(self):
        if self.isempty():
            return None
        elif self.head.next is None:
            temp=self.head.data
            self.head=None
            return temp
        else:
            temp=self.head.data
            self.head=self.head.next
            self.head.previous=None
            return temp

    def peek(self):
        if self.isempty():
            return None
        else:
            return self.head.data
    def display(self):
        iternode=self.head
        if self.isempty():
            print("Underflow")
        else:
```

```
            while(iternode!=None):
                print(iternode.data,"-->",end=" ")
                iternode=iternode.next
            return
mystack=stack()
mystack.push(10)
mystack.push(20)
mystack.push(30)
mystack.push(40)
mystack.display()
print("\nTop element is",mystack.peek())
mystack.pop()
mystack.pop()
mystack.display()
print("\nTop element is",mystack.peek())
```

```
40 --> 30 --> 20 --> 10 -->
Top element is 40
20 --> 10 -->
Top element is 20
```

[ ]: