# Create a menu driven program for arrays having access to:  
# 1. Print the array   
# 2. Insert value at end   
# 3. Insert value at front  
# 4. Insert value at a given index location   
# 5. Insert value after a given value  
# 6. Search value in array  
  
  
  
**import** numpy **as** np  
**class** **menu**:  
 **def** **\_\_init\_\_**(self,size):  
 self.arr = np.zeros(size)  
 self.N = 0  
 self.size = size  
 **def** **traverse**(self):  
 **if** self.N == 0:  
 print("\nEmpty Array")  
 **else**:  
 **for** i **in** range(0,self.N):  
 print(self.arr[i])  
 **def** **insert\_front**(self,val):  
 **if** self.N==self.size:  
 print("Array overflow")  
 **else**:  
 **for** i **in** range(self.N-1,-1,-1):  
 self.arr[i+1]=self.arr[i]  
 self.arr[0]=val  
 self.N=self.N+1  
 print("Element inserted at the front")  
 self.traverse()  
 **def** **insert\_end**(self,val):  
 **if** self.N==self.size:  
 print("Array oveflow")  
 **else**:  
 self.arr[self.N]=val  
 self.N=self.N+1  
 print("Element inserted at the end")  
 self.traverse()  
 **def** **insert\_pos**(self,val,pos):  
 **if** self.N==self.size:  
 print("Array overflow")  
 **else**:  
 **if** pos<0 **or** pos>self.N:  
 print("Invalid input")  
 **elif** pos==self.N:  
 self.arr[pos]=val  
 self.N=self.N+1  
 print("Element inserted at the given position")  
 **else**:  
 **for** i **in** range(self.N-1,pos,-1):   
 self.arr[i+1]=self.arr[i]  
 self.arr[pos]=val  
 self.N=self.N+1  
 print("Element inserted at the given position")  
 self.traverse()  
 **def** **insert\_after\_value**(self, giv\_val, val):  
 indices = np.where(self.arr == giv\_val)  
 index = indices[0][0]  
 **if** self.N==self.size:  
 print("Array overflow")  
 **else**:  
 **for** i **in** range(self.N-1,index,-1):  
 self.arr[i+1] = self.arr[i]  
 self.arr[index+1] = val   
 self.N = self.N+1  
 print("Value replaced successfully")  
 self.traverse()  
 **def** **search\_val**(self, val):  
 indeces = np.where(self.arr == val)  
 index = indeces[0]  
 condition = False  
 **if** self.N==self.size:  
 print("Array overflow")  
 **else**:  
 **for** i **in** self.arr:  
 **if** i == val:  
 condition = True  
 print(f"{i} found at index {index}")  
 **break**  
 **if**(condition==False):  
 print("Element not found")  
  
  
  
  
  
# 2. Delete value at end  
  
# 3. Delete value at front  
  
# 4. Delete value at a given index location  
  
# 5. Delete value after a given value  
  
  
  
 **def** **delete\_end**(self):  
 **if** self.N == 0:  
 print("Array underflow")  
 **else**:  
 self.N=self.N-1  
 print("Element deleted at end")  
 self.traverse()  
 **def** **delete\_front**(self):  
 **if** self.N==0:  
 print("Array underflow")  
 **else**:  
 **for** i **in** range(0,self.N-1):  
 self.arr[i]=self.arr[i+1]  
 self.N=self.N-1  
 print("Element deleted at front")  
 self.traverse()  
 **def** **delete\_index**(self,dex):  
 **if** self.N==0:  
 print("Array underflow")  
 **else**:   
 **for** i **in** range(dex+1,self.N):  
 self.arr[i-1]=self.arr[i]  
 print("Element deleted at index")  
 self.N-=1  
 self.traverse()  
 **def** **delete\_givval**(self,giv\_val):  
 indices = np.where(self.arr == giv\_val)  
 index = indices[0][0]  
 **if** self.N==0:  
 print("Array underflow")  
 **else**:  
 **for** i **in** range(index+2,self.N):  
 self.arr[i-1]=self.arr[i]  
 np.delete(self.arr, index+1)  
 self.N-=1  
 print("Element deleted")  
 self.traverse()  
   
   
   
# Bubble sort  
  
# Insertion sort  
  
# Selection sort  
  
 **def** **bubble\_sort**(self):  
 **for** i **in** range(0,self.N-1):  
 **for** j **in** range(0,self.N-i-1):  
 **if** self.arr[j]>self.arr[j+1]:  
 temp = self.arr[j+1]  
 self.arr[j+1] = self.arr[j]  
 self.arr[j] = temp  
 self.traverse()  
   
 **def** **insertion\_sort**(self):  
 **for** j **in** range(1,self.N):  
 key = self.arr[j]  
 i = j - 1  
 **while** i>=0 **and** self.arr[i]>key:  
 self.arr[i+1]=self.arr[i]  
 i = i-1  
 self.arr[i+1] = key  
 self.traverse()  
   
 **def** **selection\_sort**(self):  
 **for** i **in** range(0,self.N-1):  
 min = i  
 **for** j **in** range(i+1,self.N):  
 **if** self.arr[j]<self.arr[min]:  
 min = j  
 **if** self.arr[i]>self.arr[min]:  
 temp = self.arr[i]  
 self.arr[i] = self.arr[min]  
 self.arr[min] = temp  
 self.traverse()  
  
  
  
  
  
size = 10  
p = menu(size)  
**while**(1):  
 print("\n============Array Menu============\n")  
 print("1. Print the elements in the array")  
 print("2. Insert value at the end")  
 print("3. Insert values at the front")  
 print("4. Insert values at a position")  
 print("5. Insert values at a given value")  
 print("6. Search values in array")  
 print("7. Delete values at the end")  
 print("8. Delete values at the front")  
 print("9. Delete values at a given index")  
 print("10. Delete values after a given value")  
 print("11. Bubble sort")  
 print("12. Insertion sort")  
 print("13. Selection sort")  
 print("14. Exit")  
   
  
  
 ch=int(input("\nEnter your choice : "))  
  
 **if**(ch==1):  
 p.traverse()  
 **elif**(ch==2):  
 val = int(input("Enter your value : "))  
 p.insert\_end(val)  
 **elif**(ch==3):  
 val = int(input("Enter your value : "))  
 p.insert\_front(val)  
 **elif**(ch==4):  
 val = int(input("Enter your value : "))  
 pos = int(input("Enter the index where you want to enter your value : "))  
 p.insert\_pos(val,pos)  
 **elif**(ch==5):  
 giv\_val = int(input("Enter the value after which you want your new value : "))  
 val = int(input("Enter the new value : "))  
 p.insert\_after\_value(giv\_val,val)  
 **elif**(ch==6):  
 val = int(input("Enter the element you want to search : "))  
 p.search\_val(val)  
 **elif**(ch==7):  
 p.delete\_end()  
 **elif**(ch==8):  
 p.delete\_front()  
 **elif**(ch==9):  
 dex = int(input("Enter index of the value to be deleted : "))  
 p.delete\_index(dex)  
 **elif**(ch==10):  
 giv\_val = int(input("Enter the value after which you want to delete the element : "))  
 p.delete\_givval(giv\_val)  
 **elif**(ch==11):  
 p.bubble\_sort()  
 **elif**(ch==12):  
 p.insertion\_sort()  
 **elif**(ch==13):  
 p.selection\_sort()  
 **elif**(ch==14):  
 print("Exited successfully")  
 **break**  
 **else**:  
 print("Invalid input!!")

# Find transpose of a matrix.  
  
# Find the sum of two matrices.  
  
# Multiply two matrices.  
  
  
**import** numpy **as** np  
  
arr = np.array([[1,2,3],[3,4,5],[5,6,7]])  
arr1 = np.array([[1,2,3],[3,4,5],[5,6,7]])  
  
**def** **transpose**(arr):  
 trans = np.array([[0 **for** i **in** range(len(arr))] **for** j **in** range(len(arr[0]))])  
 **for** k **in** range(len(arr)):  
 **for** l **in** range(len(arr[0])):  
 trans[k][l] = arr[l][k]  
 print(f"\nThe transpose of the matrix \n{arr}\n is : \n{trans}")  
transpose(arr)  
  
**def** **sum\_matrix**(arr,arr1):  
 sum = arr + arr1  
 print(f"\nSum of matrix \n{arr}\n and \n{arr1}\n is : \n{sum}")  
sum\_matrix(arr,arr1)  
  
**def** **mul\_matrix**(arr,arr1):  
 n = len(arr)  
 m = len(arr1)  
 **if** len(arr[0]) != len(arr1[0]):  
 print("Matrices cannot be multiplied!")  
 **else**:  
 multiply = np.array([[0 **for** i **in** range(len(arr))] **for** j **in** range(len(arr[0]))])  
 **for** i **in** range(len(arr)):  
 **for** j **in** range(len(arr[0])):  
 **for** k **in** range(len(arr1)):  
 multiply[i][j] = multiply[i][j] + arr[i][k] \* arr1[k][j]  
 print(f"\nMultiplication of matrix \n{arr}\n and \n{arr1}\n is : \n{multiply}")  
mul\_matrix(arr,arr1)