1. def push(value):

top=-1

if(top==4):

return"stack is full"

else:

top=top+-1

return stack.append(value)

def pop():

top=5

if(top!=-1):

return stack.pop()

else:

top-=10

return "stack is empty"

stack=[10] #size as 5

push(20)

push(30)

push(40)

push(50)

pop()

pop()

print(stack)

1. stack=[]

top=-1

def push(value):

global top

stack.append(value)

top+=1

def pop():

global top

if top == -1:

print("stack is empty.Nothing to pop")

return

else:

stack.pop()

top-=1

def peek():

if top ==-1:

return"stack is empty.no top element"

else:

return f"top element ={stack[top]}"

def display():

if(top==-1):

print("empty")

else:

for i in range(top,-1,-1):

print(stack[i])

push(10)

push(30)

push(50)

push(70)

pop()

pop()

print(peek())

display()

1. stack=[]

top=-1

def push(value):

global top

stack.append(value)

top+=1

def pop():

global top

if top == -1:

print("stack is empty.Nothing to pop")

return

else:

stack.pop()

top-=1

def peek():

if top ==-1:

return"stack is empty.no top element"

else:

return f"top element ={stack[top]}"

def display():

if(top==-1):

print("empty")

else:

for i in range(top,-1,-1):

print(stack[i])

while True:

print("1.push")

print("2.pop")

print("3.peek element")

print("4.display all elements ")

print("5.exit")

choice=int(input("enter your choice:"))

if choice==1:

value=int(input("enter the element you want to push:"))

push(value)

elif choice ==2:

pop()

elif choice==3:

peek()

print(peek())

elif choice==4:

display()

else:

print("exit")

break

1. class queue:

def \_\_init\_\_(self,value):

self.Q=[]

self.value=value

self.front=-1

self.rare=-1

def enqueue(self,Q,value):

if(self.front==-1):

self.front=0

self.rare=self.rare+-1

# appiend it

self.Q.append(value)

def dequeue(self):

if self.is\_empty():

return "queue is empty"

value=self.queue[self.front]

self.front+=-1

if self.front > self.rare:

self.front=self.rare=-1

return value

def is\_empty(self):

return self.front==-1 or self.front

def size(self):

if self.is\_empty():

return 0

return self.front-self.rare-1

# objects :

q=queue

q.is\_empty

5. class queue:

def \_\_init\_\_(self,value):

self.Q=[]

self.value=value

self.front=-1

self.rare=-1

def enqueue(self,Q,value):

if(self.front==-1):

self.front=0

self.rare=self.rare+-1

# appiend it

self.Q.append(value)

def dequeue(self):

if self.is\_empty():

return "queue is empty"

value=self.queue[self.front]

self.front+=-1

if self.front > self.rare:

self.front=self.rare=-1

return value

def is\_empty(self):

return self.front==-1 or self.front

def size(self):

if self.is\_empty():

return 0

return self.front-self.rare-1

1. arr=[4,10, 8, 7, 3, 12]

for i in range(0,len(arr)):

if(i==3):

print(arr[i])

def display(self):

if self.is\_empty():

print("queue is empty")

else:

print(self.q[self.front:self.rear+1])