DATA 200

Paramdeep Saini, SJSU

Week 3 - DATA 200

- Questions from Week 1 & 2?
  - HW?
  - Classes/Objects/IS-A/HAS-A

How to Solve following?

- Web Browser can keep of visited page and when you press back button can provide you last visited page?
- Text Editor Undo function?

Stacks

- What is a Stack?
  - A stack is a data structure of ordered items such that items can be inserted and removed only at one end.
- Order
  - LIFO
- Examples of Stacks:
  - Pez Dispenser
  - Cafeteria Trays

Stacks Functions

Function	Description
Pop()	Look at the item on top of the stack and remove it
PUSH()	Place an item in stack
PEEK()	Check the top element

Stacks

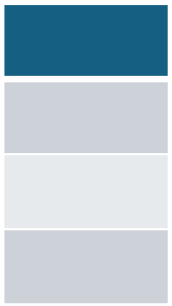
- A stack is a LIFO (Last-In/First-Out) data structure
- A stack is sometimes also called a pushdown store.
- What are some applications of stacks?
  - Program execution
  - Parsing
  - Evaluating postfix expressions

Stacks – Underflow

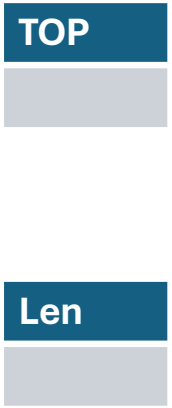
- If stack is empty and we try to pop the value? What should happen?
  - Underflow

Stacks – Example

s.push(L)
s.push(K)
Len(s)
s.push(J)
s.top()
s.pop()
s.push(L)
s.pop()



Stack S



Stacks – Quiz

- The following sequence of calls is executed on an empty stack:
  S.push(3), S.push(5), **S.pop()**.
  What is the return value of the last call?

- The following sequence of calls is executed on an empty stack:
  S.push(3), S.push(2), S.top(), **len(S)**.
  What is the return value of the last call?

- The following sequence of calls is executed on an empty stack:
  S.push(10), S.push(20), S.push(5), S.pop(), **S.top()**.
  What is the return value of the last call?

Python – List is a Stack?

- List support append method to add the element at the top
- List provide method to pop the element
- List is orders
- Can List be used as Stack?

Python – List is a Stack?

Stack	List
s.Push(k)	
s.Pop()	
s.top()	
s.is_empty	
Len(s)	

Do you still think list can be used as stack?

Implement MyStack Class

- Implement MyStack class and it support following operations:
  - Push
  - Pop
  - Len
  - top

HomeWork – Develop MyStack Container

Queues

- Another fundamental data structure is the ***queue***.
- It is a close "cousin" of the stack, as a queue is a collection of objects that are inserted and removed according to the ***first-in, first-out*** (***FIFO***) principle.
- That is, elements can be inserted at any time, but only the element that has been in the queue the longest can be next removed.
- E.g.
  - Ticket Queue
  - Call Center Queque

Queues

Syntax	Description
Q.enqueue(e)	Add element to the back of queue Q
Q.deuqueue()	Remove and return the first element from queue ; an error occurs if the queue is empty
Q.first()	Return a reference to the element at the front of queue , without removing it; an error occurs if the queue is empty
Q.is_empty()	Return True if queue does not contain any elements
Len(Q)	Return the number of elements in queue ; in Python, we implement this with the special method __len__

Queues

Operation	Return Value	Q First<-Last
Q.enqueue(10)	-	[10]
Q.enqueue[13]		[10,13]
Len(Q)	2	[10,13]
Q.dequeue()	10	[13]
Q.is_empty()	False	[13]
Q.dequeue()	13	[]
Q.is_empty()	True	[]
Q.Dequeue	Error	[]
Q.enqueue(17)		[17]
Q.enqueue(19)		[17,19]
Q.first()	17	[17,19]

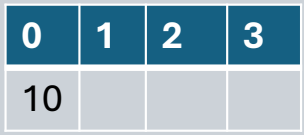
Quiz

- The following sequence of calls is executed on an empty queue:
  Q.enqueue(5), Q.enqueue(10), **Q.dequeue()**.
  What is the return value of the final call?
- The following sequence of calls is executed on an empty queue:
  Q.enqueue(10), Q.enqueue(15), Q.dequeue(), Q.enqueue(117), **len(Q)**
  What is the return value of the final call?
- The following sequence of calls is executed on an empty queue:
  Q.enqueue(7), Q.enqueue(8), Q.dequeue(), Q.enqueue(9), Q.enqueue(17), Q.dequeue(), Q.dequeue(), Q.enqueue(19), **Q.front()**.
  What is the return value of the final call?

Circular Queue

- A circular queue is another way of implementing a normal queue.
- Last element of queue is connected to first element of queue
- It is also called ***Ring Buffer***
- Operations on Circular Queue
  - getFront: First Item
  - getRear: Last Item
  - Enqueue: Insert Item
  - Dequeue: Remove Item which inserted first

Circular Queue

Step	Operation	Calcultion	
1	Front=0 Size=0 Capacity=0 Enqueue(10)	Rear=(front+size)%capacity Rear=(0+0)%4=0 Size=size+1	
2	Front=0 Size=1 Capacity=4 Enqueue(15)	Rear=(front+size)%capacity Rear=(0+1)%4=1 Size=Size=Size+1	
3	Front=0 Size=2 Capacity=4 Enqueue(16)	Rear=(front+size)%capacity Rear=(0+2)%4=2 Size=Size=Size+1	

Circular Queue

Step	Operation	Calcultion	
4	Front=0 Size=3 Capacity=4 Dequeue()	Front=(front+1)%capacity Front=(0+1)%4=1 Size=size-1	
5	Front=1 Size=2 Capacity=4 Enqueue(35)	Rear=(front+size)%capacity Rear=(1+2)%4=3 Size=Size+1	
6	Front=1 Size=3 Capacity=4 Enqueue(50)	Rear=(front+size)%capacity Rear=(1+3)%4=0 Size=Size+1	

Queue: MyQueue

Double Ended Queue (Dequeue)

- Queue-like data structure that supports insertion and deletion at both the front and the back of the queue

  A nice application of the deque is storing a web browser's history. Recently visited URLs are added to the front of the deque, and the URL at the back of the deque is removed after some specified number of insertions at the front.

  Another common application of the deque is storing a software application's list of undo operations.

DEQueues

Operation	Return Value	Q First<-Last
D.add_last(10)	-	[10]
D.add_first[13]		[13,10]
D.add_first(17)		[17,10,13]
D.first()		[17,10,13]
D.delete_last()	13	[17,10]
Len(D)	2	[17,10]
D.last()	10	[17,10]
D.is_empty()	False	[17,10]

Double Ended Queue (Dequeue)

Syntax	Description
D.add_first()	
D.add_last()	
D.delete_first()	
D.delete_last()	
D.is_empty()	
Len(D)	

Quiz

The following sequence of calls is executed on an empty deque:
D.add_last(15), D.add_first(12), D.add_last(13), **len(D)**?

Answer=?

The following sequence of calls is executed on an empty deque:
D.add_first(15), D.add_last(17), D.add_first(18),
D.delete_first(), **D.delete_first()**.
What is the return value of the final call?

Answer=?