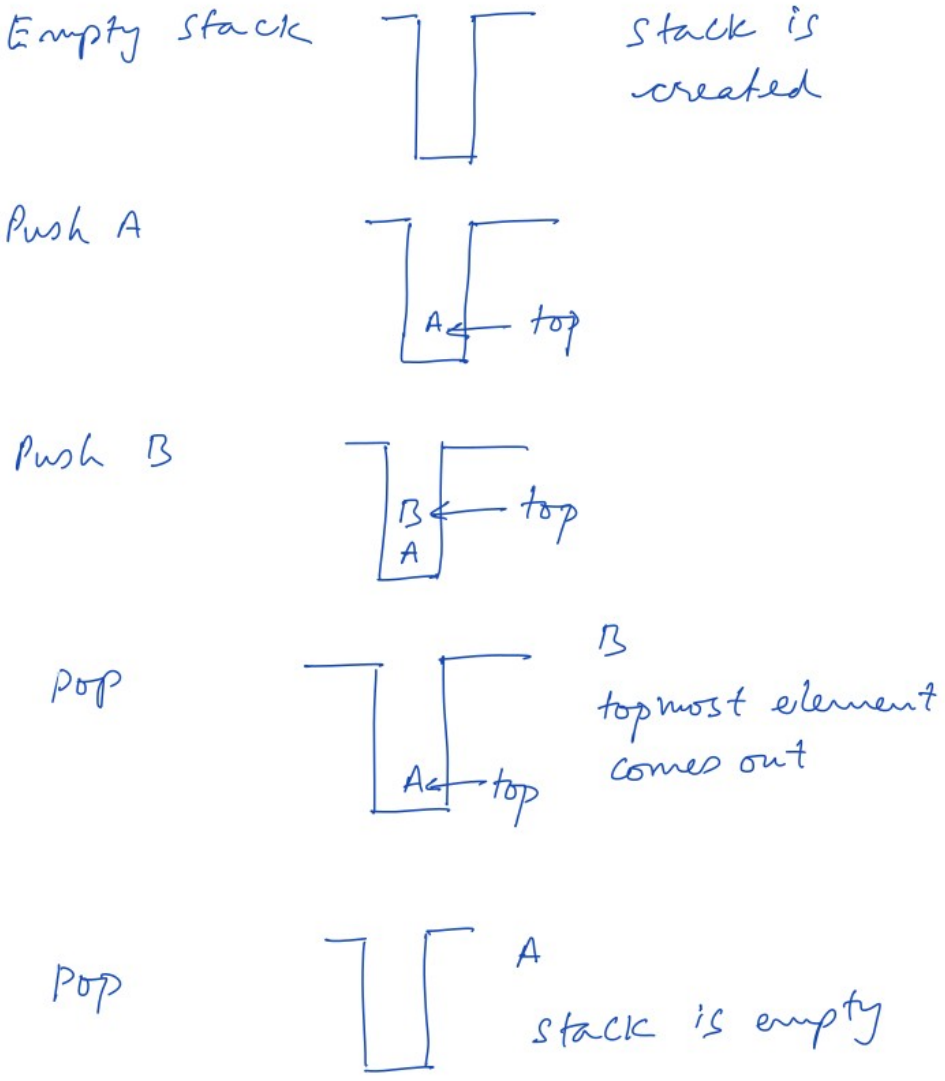
More data structures

IC152 Feb 2021

Stacks and queues

- Stacksand queues are dynamic data structures
- Delete operation is pre-specified
- Stack: last-in,first-out (LIFO)
- Queue: first-in first-out (FIFO)

Stacks


Empty stack
stack is created
Push A
A
top
Push B
B
A
top
Pop
B
topmost element comes out
A
top
Pop
A
stack is empty

```
In [41]: s2 = Stack()

In [42]: s2.push(1)

In [43]: s2.push(2)

In [44]: s2.push(3)

In [45]: print(s2)
[1, 2, 3*]

In [46]: s2.pop()
Out[46]: 3

In [47]: print(s2)
[1, 2*]
```

Create stack object

Push items into stack

Print stack. * indicates the top

Pop the stack. Returns the topmost element.

Stack after popping

```
"""
Created on Tue May 18 06:17:10 2021

@author: paddy IIT Mandi
"""

class Stack:

    # create empty stack
    def __init__(self):
        # init with empty list
        self.contents = []

#    def __str__(self):
#        return str(self.contents)

    def __str__(self):
        foo = str(self.contents)
        foo = foo[0:-1]+'*'+']'
        return(foo)

    # pushes x onto the stack
    def push(self,x):
        # append to list
        self.contents.append(x)

    def pop(self):
        # use the pop() method of list object
        return self.contents.pop()

    def peek(self):
        # return the last element
        return str(self.contents[-1])


# main pgm
s = Stack()
s.push('Monday')
s.push('Tuesday')
print('Popping: ' + s.pop())
print('Popping: ' + s.pop())
s.push('January')
s.push('March')
s.push('Friday')
print('Top is now: ' + s.peek())
print(s)

```

Python list can be used to make a stack.

The class Stack defined here is a wrapper around the list.

The wrapper is used here to give a more familiar interface to the stack.

```
Popping: Tuesday
Popping: Monday
Top is now: Friday
['January', 'March', 'Friday'*]
```

Output

Based on the application, a stack can have overflow and underflow.

Pushing onto a stack that is already full

Popping from an empty stack

Think: Python lists grow as needed. How will you implement overflow in the stack defined in the previous slide?

HW: Implement `isEmpty()`, `len()`, `isOnTop(x)` for the `Stack` class.

Applications of stacks

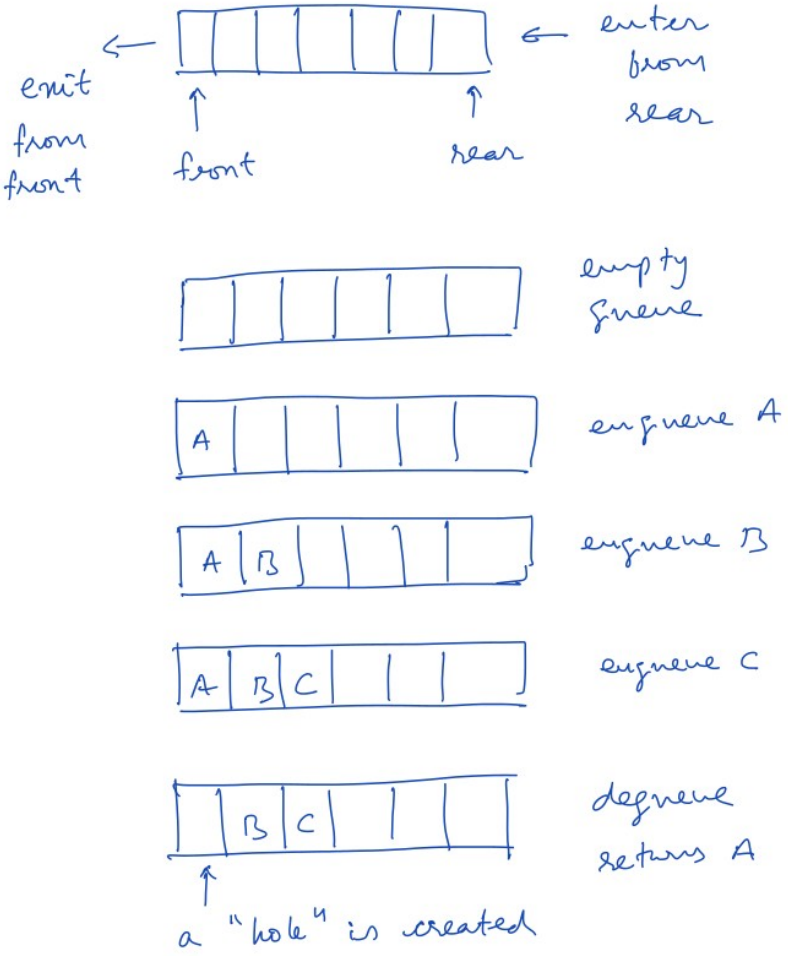
- Undo in text editors
- Uses in various other algorithms
- Function calls in programs

```
 8  def f():
 9      x = 5
10      g()
11      print('in f',x)
12
13  def g():
14      x = 7
15      h()
16
17  def h():
18      print('in h',x)
19
20
21  x = 3
22  f()
23  print('in main',x)
```

Where to go after executing h( )?
Need to come back to line 15.
Use a stack to keep track.

Queues

- Using a standard list is inefficient for queues
  - Removal from front requires movement of data to fill up the space created



- Python provides `collections.deque`
- Generalization of stacks and queues
- Provides many methods, in addition to basic stack and queue operations
- `dequeue` can also be used as a stack
- See more here: https://docs.python.org/3/library/collections.html#collections.deque

```
 8 from collections import deque
 9
10 queue = deque()
11 queue.append("Terry")
12 queue.append("Graham")
13 queue.popleft()
14 queue.popleft()
15
```

Using deque as a queue (FIFO)

Applications of queues

- Job scheduling in operating systems
- As auxilary data structures in various algorithms