lab_cipher Feedback

Average completion time looks to be ~2 hours

Pretty happy with outcome of lab overall

Some people don’t like the 20 minute lecture.

Avg: 18 / 15
lab_cipher review

Plaintext:  
Ciphertext:  

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
</tr>
</tbody>
</table>
Course Pacing Feedback

Most of class actually overqualified for class

Everyone thinks pacing is fine

Everyone thinks content is either fine or too basic

mp_racing was too hard
Learning Objectives

Discuss implementation strategies for lists

Debate which strategy is better (and how to judge ‘better’)

Introduce the concept of asymptotic efficiency
List Implementations

1. Linked list

2.
```python
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

class linkedList:
    def __init__(self, head=None):
        self.head = head
```
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

class linkedList:
    def __init__(self, head=None):
        self.head = head

    def add(self, data):
        temp = self.head
        self.head = Node(data, temp)

if __name__ == '__main__':
    ll = linkedList()
    for i in range(3):
        ll.add(i)
class linkedList:

def index(self, pos):
    curr = self.head
    i = 0
    while(curr and i < pos):
        curr = curr.next
        i+=1
    if i == pos:
        return curr
Linked List

Insert(X, 2)

Head

C -> S -> 2 -> 7 -> 7 -> None
class linkedList:
    def insert(self, data, pos=0):
        if (pos == 0):
            self.add(data)
        else:
            prev = self.index(pos-1)
            temp = prev.next
            prev.next = Node(data, temp)
Linked List

Remove (7)

Head

C -> S -> 2 -> 7 -> 7 -> None
class linkedList:
    def remove(self, data):
        prev = None
        curr = self.head
        while(curr):
            if curr.data == data:
                if prev == None:
                    self.head = self.head.next
                    break
                else:
                    prev.next = curr.next
                    break
            else:
                prev = curr
                curr = curr.next
Linked List

Head

Head

C

S

2

7

7

None

Pop(4)
class linkedList:
    def pop(self, pos):
        if(pos == 0):
            temp = self.head.data
            self.head = self.head.next
            return temp
        else:
            prev = self.index(pos-1)
            temp = prev.next
            prev.next = temp.next
            return temp.data
Exercise 1

```python
ll = linkedList()
for i in range(3):
    ll.add(i)

ll.insert(6,1)
ll.insert(3,1)
ll.insert(10,3)

ll.remove(6)
ll.remove(2)
ll.remove(12)

ll.pop(2)
ll.pop(0)
```

List Functions

Index:

Add:

Insert:

Remove:

Pop:
List Implementations

1. Linked List

2. Arrays
Array
Creating an array list

class arrayList:
    def __init__(self, array=[]):
        self.array = array
def add(self, data):

def index(self, pos):

1 | def index(self, pos):
2 |     return self.array[pos]
3 |
def insert(self, data, pos=0):
def remove(self, data):
    self.array.remove(data)
Array Pop

def pop(self, pos):

1 2 3
Which implementation is better?
What do we care about when we write code?