CS1010S Tutorial 6

Wang Zexin

National University of Singapore
Year 2 QF

wang.zexin@u.nus.edu

March 13, 2017

Today's Agenda

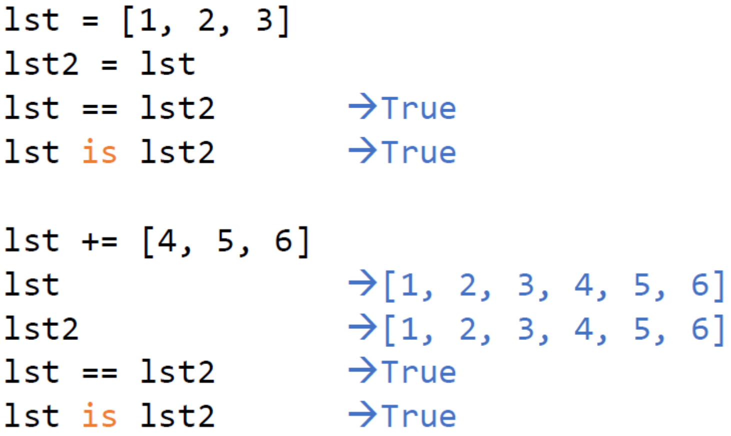
Recap - List

Why do you use list?

- Mutable
- list1 + list2
- list.append(element) → also touch on **list comprehension**
- list.extend(element)
- del a[index]
- len, min, max, in, * scalar (sequence operations)
- list.copy() → **shallow copy**, often tested in finals
- list.insert(position, element)
- list.pop(position)
- list.remove(element)
- list.clear()
- list.reverse()

Recap - Box and pointer diagram

- Most important when you are using lists than tuples
- Below is a screenshot of your lecture notes
- Why is it so? Explain using box and pointer diagram



```
lst = [1, 2, 3]
lst2 = lst
lst == lst2         →True
lst is lst2         →True

lst += [4, 5, 6]
lst                 →[1, 2, 3, 4, 5, 6]
lst2                →[1, 2, 3, 4, 5, 6]
lst == lst2         →True
lst is lst2         →True
```

Recap - Searching

- Linear search
    - Time: $O(n)$
- Binary search
    - only when array is sorted
    - Time: $O(\log n)$

Recap - Sorting

- Selection sort
  - Time: $O(n^2)$
  - Stable
- Merge sort
  - Time: $O(n \log n)$
  - Space: $O(n)$
  - Stable
- list.sort(key = lambda x: f(x), reverse = True)
  - Sort based on the value of f(x)
  - Sort to be in reverse order
- Other sorting methods and many more ...
  - Bubble sort
  - Quick sort
  - Shell sort
  - Bucket sort
  - Heap sort

Question 1

Ben Bitdiddle is required to implement a function *at_least_n* which takes in a list of integers and an integer n, and returns the original list with all the integers smaller than n removed.

Question 1 Part A

```python
def at_least_n(lst, n):
    for i in range(len(lst)):
        if lst[i] < n:
            lst.remove(lst[i])
    return lst
```

This is wrong as per hinted by part B.

Question 1 Part B

```
def at_least_n(lst, n):
    for i in lst:
        if i < n:
            lst.remove(i)
    return lst
```

This is wrong as per hinted by part C.
These two implementations are both wrong, because they are modifying the list while looping through it.

Question 1 Part C Discussion

Now your task is to give a correct implementation.

- First of all, you are supposed to remove the elements during the loop.
- Then how are you going to refrain from looping through the list itself?
- Recall the functionality of the method copy of list.
- When you are looping through a shallow copy of the list,
- feel free to modify the list itself!
- Another method is to adjust the index carefully for each deletion.

Question 1 Part C Zexin's solutions

```
def at_least_n(lst, n):
    lstcopy = lst.copy()
    for element in lstcopy:
        if element < n:
            lst.remove(element)
    return lst

def at_least_n(lst, n):
    i = 0
    while i < len(lst):
        if lst[i] < n:
            lst.pop(i)
        else:
            i += 1
    return lst
```

Question 1 Part C May's solutions

```
def at_least_n(lst, n):
    i = len(lst) - 1
    while i >= 0:
        if lst[i] < n:
            lst.pop(i)
        i -= 1
    return lst
```

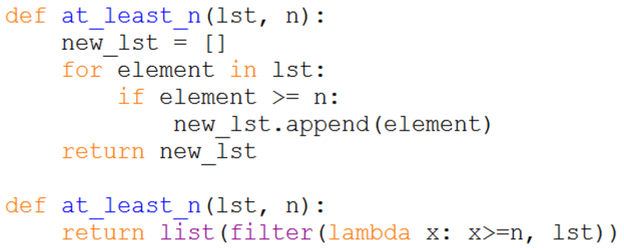
This solution very nicely made use of backward indices.

Question 1 Part D Discussion

Now you are supposed to generate a new list

- First step, generate a new empty list,
- Then loop through all the elements in the list,
- For each element, check if it is greater or equal to n
- Then append those into the new list.
- Then we are done.

Question 1 Part D Zexin's solutions



```python
def at_least_n(lst, n):
    new_lst = []
    for element in lst:
        if element >= n:
            new_lst.append(element)
    return new_lst

def at_least_n(lst, n):
    return list(filter(lambda x: x>=n, lst))
```

Question 2 Part A

Write a function *col_sum* which takes in a matrix and returns a list, where the *i*-th element is the sum of the elements in the *i*-th column of the matrix, using nested for loops. You can assume that the matrix will not be empty, and has exactly $m \times n$ elements, where m and n are positive integers.

Question 2 Part A Discussion

Your task now is to:

- Loop from the left most column to the right most one.
- Extract all the elements from the matrix in each column.
- Sum these elements up and add it to the list
- Note that solutions provided will be written in the simplest possible way

Question 2 Part B

Write a function *row_sum* which takes in a matrix and returns a list, where the i-th element is the sum of the elements in the i-th row of the matrix.

Question 2 Part B Discussion

Your task now is to:

- Loop from the upmost row to the bottom most one.
- Extract all the elements from the matrix in each column.
- Sum these elements up and add it to the list.
- Note that solutions provided will be written in the simplest possible way

Question 2 Part C

Write a function *transpose* which takes in a matrix and transposes it. Basically, this converts a $m \times n$ matrix into a $n \times m$ matrix.

Question 2 Part C Discussion

Your task now is to:

- Loop from the left most column to the right most one.
- Extract all the elements from the matrix in each column.
- List these elements together and add it to a new list
- Note that solutions provided will be written in the simplest possible way

Question 2 Zexin's solution

```python
def col_sum(matrix):
    m = len(matrix)
    n = len(matrix[0])
    return [sum([y[x] for y in matrix]) for x in range(n)]

def row_sum(matrix):
    return [sum(x) for x in matrix]

def transpose(matrix):
    m = len(matrix)
    n = len(matrix[0])
    return [[y[x] for y in matrix] for x in range(n)]
```

Question 3

Sort the list of integers $[5, 7, 4, 9, 8, 5, 6, 3]$ into ascending order on paper. Show how the contents of the list evolves in each step. (This question is meant to ensure that you have a high-level understanding of the common sort algorithms. You are not required to write any code.)

Question 3 Part A Solution - Insertion sort

[5, 7, 4, 9, 8, 5, 6, 3] start

[5, 7, 4, 9, 8, 5, 6, 3] insert 7

[4, 5, 7, 9, 8, 5, 6, 3] insert 4

[4, 5, 7, 9, 8, 5, 6, 3] insert 9

[4, 5, 7, 8, 9, 5, 6, 3] insert 8

[4, 5, 5, 7, 8, 9, 6, 3] insert 5

[4, 5, 5, 6, 7, 8, 9, 3] insert 6

[3, 4, 5, 5, 6, 7, 8, 9] insert 3

Question 3 Part B Solution - Selection sort

[5, 7, 4, 9, 8, 5, 6, 3] start

[3, 5, 7, 4, 9, 8, 5, 6] smallest=3

[3, 4, 5, 7, 9, 8, 5, 6] smallest=4

[3, 4, 5, 7, 9, 8, 5, 6] smallest=5

[3, 4, 5, 5, 7, 9, 8, 6] smallest=5

[3, 4, 5, 5, 6, 7, 9, 8] smallest=6

[3, 4, 5, 5, 6, 7, 9, 8] smallest=7

[3, 4, 5, 5, 6, 7, 8, 9] smallest=8

[3, 4, 5, 5, 6, 7, 8, 9] smallest=9

Question 3 Part C Solution - Bubble sort

$[5, 7, 4, 9, 8, 5, 6, 3 \|]$

$[5, 4, 7, 8, 5, 6, 3 \| 9]$

$[4, 5, 7, 5, 6, 3 \| 8, 9]$

$[4, 5, 5, 6, 3 \| 7, 8, 9]$

$[4, 5, 5, 3 \| 6, 7, 8, 9]$

$[4, 5, 3 \| 5, 6, 7, 8, 9]$

$[4, 3 \| 5, 5, 6, 7, 8, 9]$

$[3 \| 4, 5, 5, 6, 7, 8, 9]$

$[\| 3, 4, 5, 5, 6, 7, 8, 9]$

Question 3 Part D Solution - Merge sort

[5, 7, 4, 9, 8, 5, 6, 3] start

[5, 7, 4, 9] [8, 5, 6, 3] split

[5, 7] [4, 9] [8, 5] [6, 3] split

[5, 7] [4, 9] [5, 8] [3, 6] merge

[4, 5, 7, 9] [3, 5, 6, 8] merge

[3, 4, 5, 5, 6, 7, 8, 9] merge

Question 4 Part A

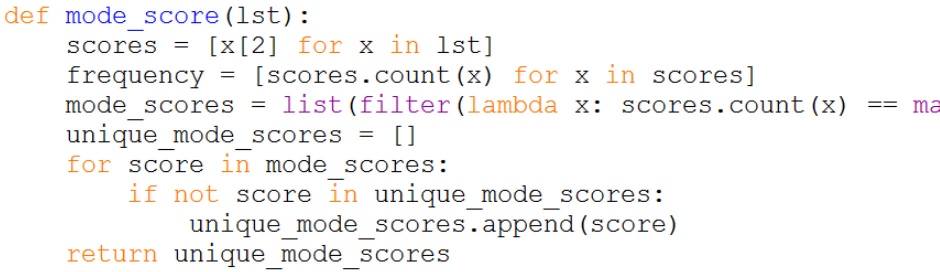
Write a function *mode_score* that takes a list of students and returns a list of the mode scores (scores that appear the most number of times). If there is only one score with the highest frequency, then this list would only have one element.

Question 4 Part A Discussion

In steps, these are what you should do:

- Extract all the scores from the students list.
- Find the frequency of each individual scores.
- Find the maximum frequency.
- Find those scores corresponding to the maximum frequency.
- Remove all the duplicates from the list obtained above.

Question 4 Part A Zexin's solution


```
def mode_score(lst):
    scores = [x[2] for x in lst]
    frequency = [scores.count(x) for x in scores]
    mode_scores = list(filter(lambda x: scores.count(x) == ma
    unique_mode_scores = []
    for score in mode_scores:
        if not score in unique_mode_scores:
            unique_mode_scores.append(score)
    return unique_mode_scores
```

Question 4 Part B

Write a function *top_k* that takes a list of students and an integer k and returns a list of the names of the k students with the highest scores in alphabetical order. If there are students in the range $(k+1; \ldots k+i)$ who have the same score as the kth student, include them in the list as well.

Question 4 Part B Discussion

In steps, these are what you should do:

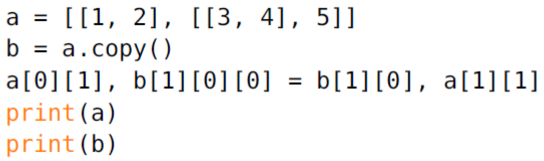
- Sort the list by alphabetical order. (why do this first?)
- Sort the list by reverse score order.
- Find the cutoff point starting from k.

Question 4 Part B Zexin's solution

```
def top_k(lst, k):
    lst.sort(key = lambda x: x[0])
    lst.sort(key = lambda x: x[2], reverse = True)
    while k < len(lst) and (lst[k-1][2] == lst[k][2]):
        k = k + 1
    return lst[:k]
```

Extra stuff: One final question

If time permits, we will go through this.



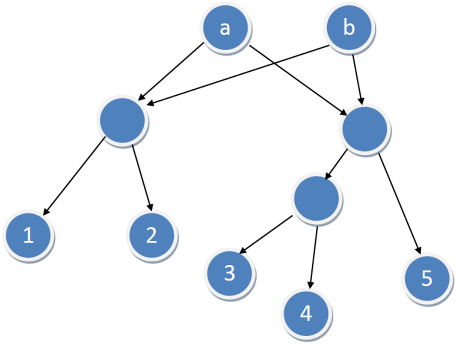
```
a = [[1, 2], [[3, 4], 5]]
b = a.copy()
a[0][1], b[1][0][0] = b[1][0], a[1][1]
print(a)
print(b)
```

- This is a problem in which you need to use box-and-pointer diagram.

Extra stuff: One final question

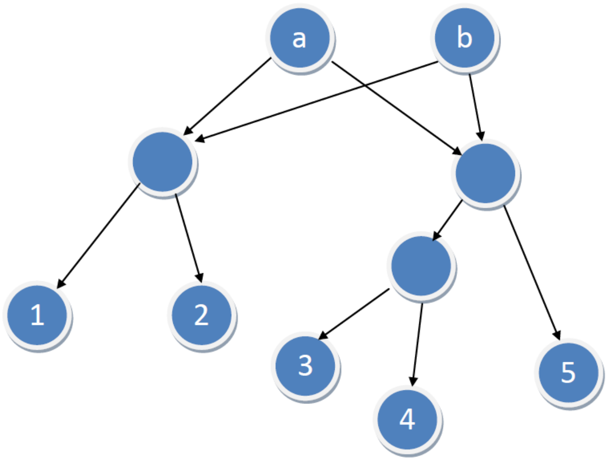
```python
a = [[1, 2], [[3, 4], 5]]
b = a.copy()
a[0][1], b[1][0][0] = b[1][0], a[1][1]
print(a)
print(b)
```

Craft out the box and pointer diagram



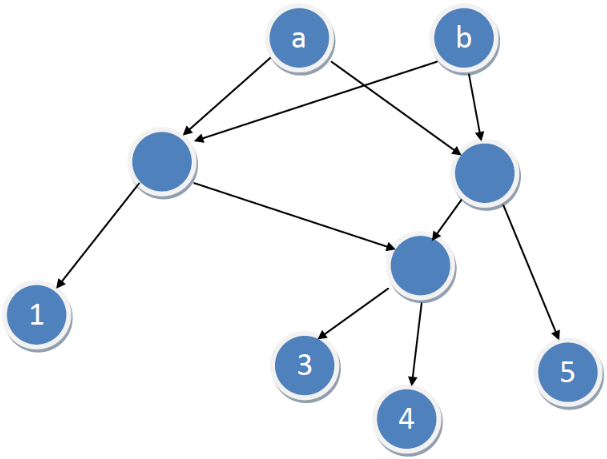
Extra stuff: One final question

Start



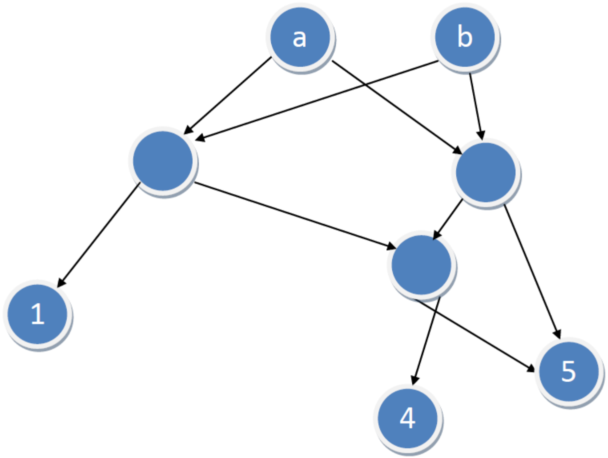
Extra stuff: One final question

$$a[0][1] = b[1][0]$$

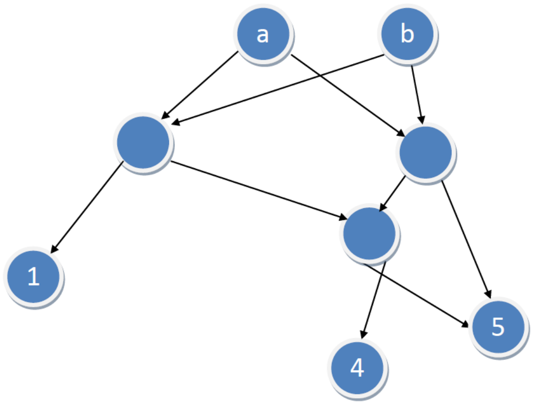


Extra stuff: One final question

$$b[1][0][0] = a[1][1]$$



Extra stuff: One final question



Hence the solution is:

$$a = [[1, [5, 4]], [[5, 4], 5]]$$
$$b = [[1, [5, 4]], [[5, 4], 5]]$$

Feedback & more

- Slides + relevant material available at:

  `https://github.com/wangzexin/Teaching`

- After the tutorial, if you have further questions:

  `wang.zexin@u.nus.edu`

Thank You

wang.zexin@u.nus.edu