# Programming for Data Science (CSE3041)

Ramesh Ragala

VIT Chennai Campus

July 22, 2020

## Introduction to List

### Problem - 1

A farmer with a fox, a goose, and a sack of corn needs to cross a river. Now he is on the east side of the river and wants to go to west side. The farmer has a rowboat, but there is room for only the farmer and one of his three items. Unfortunately, both the fox and the goose are hungry. The fox cannot be left alone with the goose, or the fox will eat the goose. Likewise, the goose cannot be left alone with the sack of corn, or the goose will eat the corn. Given a sequence of moves find if all the three items fox, goose and corn are safe. The input sequence indicate the item carried by the farmer along with him in the boat. 'F' - Fox, 'C' - Corn, 'G' - Goose, N-Nothing. As he is now on the eastern side the first move is to west and direction alternates for each step.

# Introduction to List

## Pseudocode

```
READ items_ carried
SET east as G, C, F
SET west as empty
SET from_ Dir = east and to_Dir = west
FOR each item in items_ carried
    IF from_ Dir == east THEN
        remove item from east and add to west
        IF east or west contains 'C' and 'G' or 'G' and 'F' THEN
            PRINT 'NOT SAFE'
            BREAK
    ELSE
        remove item from west and add to east
        IF east or west contains 'C' and 'G' or 'G' and 'F' THEN
            PRINT 'NOT SAFE'
            BREAK
    END IF
    IF from_ Dir == east THEN
        SET from_ Dir = west
        SET to_ Dir = east
    ELSE
        SET from_ Dir = east
        SET to_ Dir = west
    END IF
END FOR
PRINT 'SAFE'
```

## Introduction to List

**Problem**

While going for a Shopping . . . .!!!???

**Problem**

Imagine you have the scores in “Python Programming” for 100 students. If you want to find the average score in Python. . . ?

## Introduction to List

**Simple Statistics**

- Many programs deal with large collections of similar information.
    - Words in a document
    - Students in a course
    - Data from an experiment
    - Customers of a business
    - Graphics objects drawn on the screen
    - Cards in a deck

## Introduction to List

**List**

## Introduction to List

**Examples**

- Apple, Banana, Berry, Mango $\rightarrow$ fruits family
- Football, Basketball, Throwball, Tennis, Hockey $\rightarrow$ different kinds of balls
- Sunrise, Sugar, Cheese, Butter, Pickle, Soap, Washing Powder, Oil $\rightarrow$ grocery
- Agra, Delhi, Kashmir, Jaipur, Kolkata $\rightarrow$ places

## Introduction to List

### Introduction

- Contains multiple values that are logically related
- List is a type of mutable sequence in Python
- Each element of a list is assigned a number - index / position
- Can do indexing, slicing, adding, multiplying, and checking for membership
- Built-in functions for finding length of a sequence and for finding its
- largest and smallest elements

## Introduction to List

**What is a List?**

- Most versatile data type in Python
- Comma-separated items can be collected in square brackets
- Good thing is..
  - THE ITEMS IN THE LIST NEED NOT BE OF SAME TYPE

## Introduction to List

### Creating a list

- Creating an **EMPTY list**
  - listname = []
- Example:
  - L1 = []
  - MyList = []
  - Books = []

- Creating a **list with items**
  listname = $[item1, item2, ...]$
- Example:
  - Temp = $[100, 99.8, 103, 102]$
  - S = $['15BIT001', 'Achu', 99.9]$
  - L2 = $[1, 2, 3, 4, 5, 6, 7]$
  - Course = $['Python', 'C', 'C++', 'Java']$

## Introduction to List

### Accessing Values

- Using index or indices
  - $>>> L1 = [1, 2, 3, 4, 5, 6]$
  - $>>> print(L1[3])$ # indexing
  - $>>> 4$
  - $>>> print(L1[2 : 5])$ # slicing
  - $>>> [3, 4, 5]$

## Introduction to List

### Updating Elements

- Update an element in list using index
  - $>>> L1 = [1, 2, 3, 4, 5, 6]$
  - $>>> L1[2] = 111$
  - $>>> L1$
  - $[1, 2, 111, 4, 5, 6] \rightarrow$ Inserted a given element in specified position in List

### Deleting Elements

- Delete an element in the list using Index:
  - >>>L1 = [1,2,3,4,5,6]
  - >>>**del L1[4]**
  - L1 $\rightarrow$ check the output
  - $[1, 2, 3, 4, 6] \rightarrow$ deleted an element in specified position

## Introduction to List

**Basic Operations in List:**

- >>>len([1,2,3]) → gives the length of the list
- **3**
- >>> [1,2,3] + [4,5,6] → Concatenation of two list
- **[1,2,3,4,5,6]**
- >>> ['VIT!'] * 4 → Repetition of the elements in list
- **['VIT!', 'VIT!', 'VIT!', 'VIT!']**
- >>> str([1,2]) + "34" → equals to **"[1,2]" + "34"**
- **'[1,2]34'** → check the indices
- >>> [1,2] + list("34") → same as **[1,2]+["3","4"]**
- **[1,2,'3','4']**

**List Iteration**

- >>> 3 in [1,2,3] → membership → **True**
- >>> for i in [1,2,3]:
  print(i,end=' ') → check result

## Introduction to List

**List Comprehensions:**

- >>> res = []
- >>> for i in 'SPAM':
  res.append(i * 4)
- >>> res → check output → **['SSSS', 'PPPP', 'AAAA', 'MMMM']**
- **another way to achieve same result**
- >>> res = [i * 4 for i in 'SPAM']
- >>> res → check output → **['SSSS', 'PPPP', 'AAAA', 'MMMM']**
- expression is functionally **equivalent** to a **for loop** that builds up a list of results manually
- list comprehensions are **simpler** to code and likely **faster to** run.

## Introduction to List

**List Methods:**

- **list.append(item)** → adds a single item to the end of the list. → It does not return the new list, just modifies the original.
- **list.insert(index,item)** → it inserts the item at the given index, shifting elements to right.
- **list.extend(list2)** → it adds the items in **list2** to the end of the list. → Equal to + between two lists.
- **list.index(item)** → It searches for a given item form the start of the list and returns its index. → It returns ValueError if item not found.
- **list.remove(item)** → it searches for first occurrence of the given item and removes it. → It returns ValueError if item not found.
- **list.sort()** → sort the list in place.
- **list.reverse()** → it reverse the items in list in place.
- **list.pop(index)** → it removes and returns the element at given index. → It returns the rightmost element if the index is omitted. → It is opposite to append().

# Introduction to List

## Map

- Map is built-in function applies a function to items in a sequence and collects all the results in a new list.
- Example
  - >>> list(map(abs,[-2,-1,0,1,2])) → check output → **[2,1,0,1,2]** → Map a function across a sequence.

## Indexing and Slicing

- >>> L = ['spam', 'Spam', 'SPAM!']
- >>> L[2] → check output → **'SPAM!'** → The offset starts at zero
- >>> L[-2] → check output → **'Spam'** → Negative: count from the right
- >>> L[1:] → check output → **['Spam', 'SPAM!']** → Slicing fetches sections

# Introduction to List

## Matrix

- two-dimensional list-based array
- Example:
  - >>> matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- With one **index** → get an entire row
- With two **index** → get an item within the row
- Examples:
  - >>> matrix[1] → check result
  - [4, 5, 6]
  - >>> matrix[1][1] → check result
  - 5
  - >>> matrix[2][0] → check result
  - 7
  - >>> matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
  - >>> matrix[1][1] → check result → 5

## Introduction to List

### Multiplication of Matrices

Write a python code to multiply the matrix multiplication for a given matrices

### Python code for matrix multiplication

```
X = [[12,7,3],[4  ,5,6],[7  ,8,9]]
Y = [[5,8,1,2],[6,7,3,0],[4,5,9,1]]
result = [[0,0,0,0],[0,0,0,0],[0,0,0,0]]
# iterate through rows of X
for i in range(len(X)):
  # iterate through columns of Y
  for j in range(len(Y[0])):
     # iterate through rows of Y
     for k in range(len(Y)):
        result[i][j] += X[i][k] * Y[k][j]
for r in result:
  print(r)
```

# Introduction to List

## Zip and Enumerate

- >>> list10 = ['a1', 'a2', 'a3']
- >>> for i, a in enumerate(list10):
  print(i, a) → check output
- >>> alist = ['a1', 'a2', 'a3']
- >>> blist = ['b1', 'b2', 'b3']
- >>> for a, b in zip(alist, blist):
  print(a, b) → check output
- >>> alist = ['a1', 'a2', 'a3']
- >>> blist = ['b1', 'b2', 'b3']
- >>> for i, (a, b) in enumerate(zip(alist, blist)):
  print (i,a, b) → check output

## Introduction to List

### Multiplication of Matrices

Write a python code to multiply the matrix multiplication for a given matrices using List comprehensions

### Python code for matrix multiplication using List Comprehension

```
# Program to multiply two matrices using list comprehension
# 3x3 matrix
X = [[12,7,3],[4 ,5,6],[7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],[6,7,3,0],[4,5,9,1]]
# result is 3x4
result = [[sum(a*b for a,b in zip(X_row,Y_col))
                for Y_col in zip(*Y)]
                        for X_row in X]
for r in result:
    print(r)
```

## Introduction to List

### Insertion, Deletion and Replacement

- >>> L = [1, 2, 3]
- >>> L[1:2] = [4, 5] → **Replacement/insertion**
- >>> L → check output → [1, 4, 5, 3]
- >>> L[1:1] = [6, 7] → **Insertion only (replace nothing)**
- >>> L → Check output → [1, 6, 7, 4, 5, 3]
- >>> L[1:2] = [] → **Deletion (insert nothing)**
- >>> L → check Output → [1, 7, 4, 5, 3]
- >>> **L = [1]**
- L[:0] = [2,3,4] → **Insert all at :0, an empty slice at front**
- >>> L → check output → [2, 3, 4, 1]
- >>> L[len(L):] = [5, 6, 7] → **Insert all at len(L):, an empty slice at end**
- >>> L → check output → [2, 3, 4, 1, 5, 6, 7]

# Introduction to List

## List method calls

- >>> L = ['eat','more','SPAM!']
- >>> L.append('please') → Append method call: add item at end
- >>> L
- ['eat','more','SPAM!','please']
- >>> L.sort() → # Sort list items ('S' < 'e')
- >>> L
- ['SPAM!','eat','more','please']
- >>> L = ['abc', 'ABD', 'aBe']
- >>> L.sort() → # Sort with mixed case
- >>> L
- ['ABD','aBe','abc']
- >>> L = ['abc', 'ABD', 'aBe']
- >>> L.sort(key=str.lower) → # Normalize to lowercase
- >>> L
- ['abc','ABD','aBe']
- >>> L.sort(key=str.lower, reverse=True) → # Change sort order
- >>> L
- ['aBe','ABD','abc']

## Introduction to List

### Other common list methods

- $>>>$ L = [1, 2]
- $>>>$ L.extend([3, 4, 5]) # Add many items at end (like in-place +)
- $>>>$ L
- [1, 2, 3, 4, 5]
- $>>>$ L.pop() $\rightarrow$ # Delete and return last item 5
- $>>>$ L [1, 2, 3, 4]
- $>>>$ L.reverse() $\rightarrow$ # In-place reversal method
- $>>>$ L [4, 3, 2, 1]
- $>>>$ list(reversed(L)) $\rightarrow$ # Reversal built-in with a result (iterator)
- [1, 2, 3, 4]

## Introduction to List

**Other common list methods**

- >>> L = ['spam', 'eggs', 'ham']
- >>> L.index('eggs') → # Index of an object (search/find)
- 1
- >>> L.insert(1, 'toast') → # Insert at position
- >>> L
- ['spam', 'toast', 'eggs', 'ham']
- >>> L.remove('eggs') → # Delete by value
- >>> L
- ['spam', 'toast', 'ham']
- >>> L.pop(1) → # Delete by position 'toast'
- >>> L
- ['spam', 'ham']
- >>> L.count('spam') →# Number of occurrences 1
- 1

# Introduction to List

## Other common list methods

- >>> L = ['spam', 'eggs', 'ham', 'toast']
- >>> del L[0] →# Delete one item
- >>> L ['eggs', 'ham', 'toast']
- >>> del L[1:] →# Delete an entire section
- >>> L →# Same as L[1:] = []
- [*'eggs'*]
- >>> L = ['Already', 'got', 'one']
- >>> L[1:] = []
- >>> L
- [*'Already'*]
- >>> L[0] = []
- >>> L
- [[]]

## Introduction to List

**Hence...**

- A list is a sequence of items stored as a single object.
- Items in a list can be accessed by indexing, and sub lists can be accessed by slicing.
- Lists are mutable; individual items or entire slices can be replaced through assignment statements.
- Lists support a number of convenient and frequently used methods.
- Lists will grow and shrink as needed.

# Introduction to List

## Strings and Lists

- >>> S = 'spammy'
- >>> L = list(S)
- >>> L
- ['s','p','a','m','m','y']
- >>> L[3] = 'x' →# Works for lists, not strings
- >>> L[4] = 'x'
- >>> L
- ['s','p','a','x','x','y']
- >>> S = ''.join(L) →#uses '' for joining elements of list
- >>> S
- 'spaxxy'
- >>> 'SPAM'.join(['eggs', 'sausage', 'ham', 'toast'])
- 'eggsSPAMsausageSPAMhamSPAMtoast' →# uses 'SPAM' for joining elements of list
- >>> line = 'aaa bbb ccc'
- >>> cols = line.split()
- >>> cols
- ['aaa','bbb','ccc']

## Introduction to List

**Statistics using List**

### Statistics

Find the mean, standard deviation and median of a set of numbers

# Introduction to List

## Common List Literal and Operations

| Operation | Interpretation |
|---|---|
| L[] | An empty list |
| L[123,'abc',12.3,{}] | Four Items index 0..3 |
| L['Bob',14.0, ['Dev','mgr']] | Nested sublists |
| L=list('spam') | List of an iterable's items, list of successive integers |
| L = list(range(-4,4)) | |
| L[i] | index, index of index, slice, length |
| L[i][j] | |
| L[i:j] | |
| len(L) | |
| L1 + L2 | Concatenate, Repeat |
| L * 3 | |
| for x in L: print(x) | Iteration, membership |
| 3 in L | |
| L.append(4) | Methods, growing |
| L.extend([5,6,7]) | |

## Introduction to List

| Operation | Interpretation |
|---|---|
| L.extend([5,6,7]) | |
| L.insert(i,x) | |
| L.index(x) | Methods:Searching |
| L.count() | |
| L.sort() | Methods:Sorting,reversing |
| L.reverse() | copying(3,3+),clearing(3,3+) |
| L.copy() | |
| L.clear() | |
| L.pop(i) | Methods, Statements:shrinking |
| del L[i] | |
| del L[i:j] | |
| L[i:j] = [ ] | |
| L[i = 3 | Index assignment, slice assignment |
| L[i:j] = [4,5,6 ] | |
| L = [x**2 for x in range(5)] | List Comprehension and Maps |
| list(map(ord,'spam')) | |

## Introduction to List

Exercise - 1

Given a positions of coins of player1 and player2 in a 3X3 Tic Tac Toc board, write a program to determine if the board position is a solution and if so identify the winner of the game. In a Tic Tac Toc problem, if the coins in a row or column or along a diagonal is of the same player then he has won the game. Assume that player1 uses ‘1’ as his coin and player2 uses ’2’ as his coin. ’0’ in the board represent empty cell.

Exercise - 2

In a supermarket there are two sections S1 and S2. The sales details of item1 to itemn of section1 and item1 to itemp of section2 are maintained in a sorted order. Write a program to merge the elements of the two sorted lists to form the consolidated list.

## Introduction to List

### Exercise - 3

Watson gives Sherlock an list of N numbers. Then he asks him to determine if there exists an element in the list such that the sum of the elements on its left is equal to the sum of the elements on its right. If there are no elements to the left/right, then the sum is considered to be zero.

### Exercise - 4

Sunny and Johnny together have M dollars they want to spend on ice cream. The parlor offers N flavors, and they want to choose two flavors so that they end up spending the whole amount.
You are given the cost of these flavors. The cost of the ith flavor is denoted by ci. You have to display the indices of the two flavors whose sum is M.

## Introduction to List

### Exercise - 5

Given a list of integer values, find the fraction of count of positive numbers, negative numbers and zeroes to the total numbers. Print the value of the fractions correct to 3 decimal places.

### Exercise - 6

Given N integers, count the number of pairs of integers whose difference is K.

## Introduction to List

### Python code for matrix Addition

```
import sys
# Get dimension of first matrix
number_Of_Row1 = int(input())
number_Of_Col1 = int(input())

# Get dimension of second matrix
number_Of_Row2 = int(input())
number_Of_Col2 = int(input())
if number_Of_Row1 != number_Of_Row2 or number_Of_Col1 != number_Of_Col2:
    print('Addition not possible')
    sys.exit()
matrix1 = []
matrix2 = []

#Read first matrix
for counter1 in range (0,number_Of_Row1):
    #insert a list for a row in the matrix
    matrix1.append([])
    for counter2 in range (0, number_Of_Col1):
        matrix1[counter1].append(int(input()))

#Read second matrix
for counter1 in range (0,number_Of_Row2):
    #insert a list for a row in the matrix
    matrix2.append([])
    for counter2 in range (0, number_Of_Col2):
        matrix2[counter1].append(int(input()))
```

## Python code for matrix Addition(cont..)

```
#Add matrices and form third matrix
matrix3 = []
for counter1 in range (0,number_Of_Row2):
    #insert a list for a row in the matrix
    matrix3.append([])
    for counter2 in range (0, number_Of_Col2):
        matrix3[counter1].append(matrix1[counter1][counter2]+\
                                 matrix2[counter1][counter2])
for counter1 in range (0,number_Of_Row2):
    for counter2 in range (0, number_Of_Col2):
        print(matrix3[counter1][counter2],end = ' ')
    print()
```

# Introduction to List

## Python code for Merging Lists

```
count1 = int(input())
count2 = int(input())
list1 = []
list2 = []
#Get the elements in list1
for counter in range (0,count1):
    list1.append(int(input()))

#Get the elements in list2
for counter in range (0,count2):
    list2.append(int(input()))

consolidated_List = []
#initialize two pointers to first element in list
pointer1 = 0
pointer2 = 0
list3=[]
#add elements into the resultant list till both the pointers point to a
# valid position in list
while pointer1< count1 and pointer2 < count2:
                 if (list1[pointer1]<=list2[pointer2]):
                     list3.append(list1[pointer1])
                     pointer1+=1
                 else:
                    list3.append(list2[pointer1])
                    pointer2+=1
```

# Introduction to List

## Python code for Merging Lists (cont..)

```
#when elements from one list is added and elements are remaining in the
# Other list
#Elements are remaining in list1
while pointer1 < count1:
                list3.append(list1[pointer1])
                pointer1+=1

#Elements are remaining in list2
while pointer2 < count2:
                list3.append(list2[pointer2])
                pointer2+=1
for element in list3:
                print(element)
```

## Python code for Tic Tac Toe Problem

```
#Get input for board position
board = []
winner = 0
for row in range(0,3):
    #create a list to represent a row
    board.append([])
    for col in range(0,3):
        #enter '0' if the position is empty, '1' if it contains
        #the coin of player1 and '2' if it contains coin of player2
        board[row].append(int(input()))

#scan row by row to check if the coins in the row are of same player
for row1 in range(0,3):
    if (board[row1][0]==1 and board[row1][1]==1 and board[row1][2]==1):
        winner = 1
    elif (board[row1][0]==2 and board[row1][1]==2 and board[row1][2]==2):
        winner = 2

#scan column by column to check if the coins in the column are of same player
for col1 in range(0,3):
    if (board[0][col1]==1 and board[1][col1]==1 and board[2][col1]==1):
        winner = 1
    elif (board[0][col1]==2 and board[1][col1]==2 and board[2][col1]==2):
        winner = 2
```

Introduction to List

## Python code for Tic Tac Toe Problem(cont..)

```
#scan diagonal1 to check if the coins in the column are of same player
for col1 in range(0,3):
    if (board[0][0]==1 and board[1][1]==1 and board[2][2]==1):
        winner = 1
    elif (board[0][0]==2 and board[1][1]==2 and board[2][2]==2):
        winner = 2

#scan diagonal2 to check if the coins in the column are of same player
for col1 in range(0,3):
    if (board[0][2]==1 and board[1][1]==1 and board[2][0]==1):
        winner = 1
    elif (board[0][2]==2 and board[1][1]==2 and board[2][0]==2):
        winner = 2

print(winner)
'''
for row1 in range(0,3):
    for col1 in range(0,3):
        print(board[row1][col1])'''
```

## Assignments

- Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with 'R' first. Hint: this can be done by making 2 lists and sorting each of them before combining them.
- Given a list of strings, return the count of the number of strings where the string length is 2 or more and the first and last chars of the string are the same. Note: python does not have a $++$ operator, but $+=$ works.
- Given a list of numbers, return a list where all adjacent $==$ elements have been reduced to a single element, so [1, 2, 2, 3] returns [1, 2, 3]. You may create a new list or modify the passed in list.
- Given two lists sorted in increasing order, create and return a merged list of all the elements in sorted order. You may modify the passed in lists. Ideally, the solution should work in "linear" time, making a single pass of both lists.
- Given a string s, return a string made of the first 2 and the last 2 chars of the original string, so 'spring' yields 'spng'. However, if the string length is less than 2, return instead the empty string.

## Assignments

- Given a string s, return a string where all occurrences of its first character have been changed to '*', except do not change the first character itself. Example: 'babble' yields 'ba**le'. Assume that the string is length 1 or more.
- verbing: Given a string, if its length is at least 3, add 'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged.
- not_bad: Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'...'bad' substring with 'good'. Return the resulting string. Example: 'This dinner is not that bad!' yields: This dinner is good!

Thank you