Structures: Lists: Complex

CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I
Fall 2020

Jonathan Hudson, Ph.D
Instructor
Department of Computer Science
University of Calgary

Tuesday, September 8, 2020

UNIVERSITY OF CALGARY

List Operations and Methods

Operation	Example	Description
Indexing	`names[0]`	Access a list element
Membership	`if 'Alice' in names` `if 'Alice' not in names`	Query whether or not an item is in the list
Length	`len(names)`	Get the number of items in a list
Append	`names.append('Alice')`	Add an item to the end of the list
Insert	`names.insert(0, 'Alice')`	Insert an item at certain position
Sort	`names.sort()`	Sort the list
Reverse	`names.reverse()`	Reverse the items in the list
Count	`names.count('Alice')`	Count the number of occurence of an item

Search/Remove List

Searching For Elements

- Use `in` to check if an item is present in a list

```
data = [1,2,3,4,5]
2 in data  evaluates to True
8 in data  evaluates to False
```

- Use `index` to determine where it is in the list

```
data = [11,12,13,14]
data.index(12)  evaluates to 1
data.index(8)  results in a ValueError
```

Removing Elements

- How can we remove an item from a list?
  - Use the remove method
    - Removes the first occurrence of the item
    - Subsequent identical items remain in the list
    - Item must exist or a ValueError will occur

```
x = [1,2,1,3,4,2,1]
x.remove(1)
print(x)
```

Removing Elements

- What if we want to remove all occurrences of an item from a list?

- Use a while loop:

```
while x in myList:
    myList.remove(x)
```

Removing Elements

- What if we know the index of the item we want to remove?
  - Use *pop(index)*
  - With no parameters: Removes last item
  - With one parameter: Removes item at the index specified
  - Returns the item that is removed

```
myList = [1,2,3,4]

myList.pop()
print(myList)

myList.pop(0)
print(myList)

myList.pop(myList.index(2))
print(myList)
```

```
[1, 2, 3]
[2, 3]
[3]
```

Sorting a List

Sorting

- Sorting is the process of ordering elements of a list in ascending or descending order.

`[4, 2, 1, 3, 0]` → **Unordered** list

`[0, 1, 2, 3, 4]` → Ordered list in **ascending order**

`[4, 3, 2, 1, 0]` → Ordered list in **descending order**

- How do we sort the list?

Sorting

- Sorting is an important task
  - Needed when working with large data sets
  - Frequently occurs as part of other algorithms
- Sorting has been studied extensively
  - Many algorithms, some of which are quite complex

Sorting - Selection Sort

General idea (ascending order): The list is initially considered entirely unordered.

- Select the smallest element in the unordered portion of list
- Remove the element from unordered portion of the list and place it at the end of the ordered portion of the list.
- Repeat until no elements remain in the unordered portion of the list.

https://www.youtube.com/watch?v=xWBP4lzkoyM

Lets implement this!

You can download another solution from D2L: *21_Selection.py*

Sorting - Bubble Sort

General idea (ascending order)

- go through list from beginning to end
  - compare adjacent elements
  - swap if previous element is larger than current element
- repeat until no swaps are performed

https://www.youtube.com/watch?v=nmhjrI-aW5o

- You can download a solution from D2L: *23_Bubble.py*

Sorting in Python

- Python makes sorting a list easy
  - Use the sorted function
    - Takes one parameter which is an unsorted list
    - Returns a new list sorted into increasing order
  - Use the *sort(order)* method
    - Order is a Boolean parameter. Default is True for ascending order. False sorts in descending order.
    - Invoked on a list using dot notation
    - Modifies the list

```
list=[0,1,6,2,10]
list.sort()
print(list)
```



```
[0, 1, 2, 6, 10]
```

List Example

Practice Example

- Compute the median of a list of values entered by the user
  - User will enter an unknown number of values
  - A blank line will be used to indicate that no additional values will be entered
  - If the list has an odd number of elements
    - Median is the middle value
  - If the list has an even number of elements
    - Median is average of the two middle values

Practice Example Design

- read values from user and store in a list (using append)
- sort list (put numbers in ascending order)
- if list length is odd, display middle value (index = `len(list)/2`)
- if list length is even, display the average of two middle values (index len(list)/2 and len(list)/2 – 1)

Lets code this!

2D Lists

2D Lists

- A list of lists (images,movies,tables,matrices -> all 2D data)
- [does not have to be rectangular]

A matrix

1	2	3
4	5	6
7	8	9

A table

T01	Sandeep Zechariah
T02	Hooman Khosravi
T03	Kanishka Singh
T04	Khobaib Zaamout

- Format:

<list name> = [[<value 1>, <value 2>, ... , <value n>],
[<value 1>, <value 2>, ... , <value n>],
...
[<value 1>, <value 2>, ... , <value n>]]

rows

columns

Accessing 2D Lists

```
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matrix1)
print(matrix1[0])
print(matrix1[1])
print(matrix1[2])
```

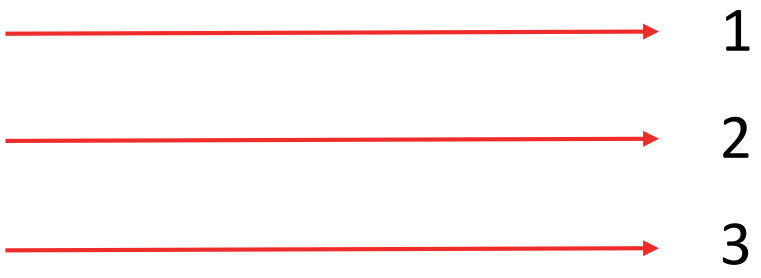
```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

```
row = matrix1[0]
print(row[0])
print(row[1])
print(row[2])
```

```
1
2
3
```

Accessing 2D Lists

```
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matrix1[0][0])
print(matrix1[0][1])
print(matrix1[0][2])
```



2D-List Creation

- Creating the following matrix programmatically:

```
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

2D-List Creation

- Creating the following matrix programmatically:

```
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
matrix1 = []
for i in range (1, 10, 3):
        row = [i , i + 1, i + 2]
        matrix1.append(row)

print(matrix1)
```

```
matrix2 = []
for i in range (0, 9, 3):
        matrix2.append([])
        for j in range (i, i + 3, 1):
                matrix2[i//3].append(j+1)

print(matrix2)
```

```
matrix3=[]

ROWS=3
COLS=3
counter = 1

for row in range(ROWS):
    matrix3.append([])
    for col in range(COLS):
        matrix3[row].append(counter)
        counter+=1

print(matrix3)
```

2D-List Printing

- Using *print (matrix)*
- Using loops:

```
for row in matrix:
    print(row)
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

Print one row per iteration

```
for row in matrix:
    output = ''
    for num in row:
        output += str(num) + ' '
    print (output)
```

Print one row per iteration

```
1 2 3
4 5 6
7 8 9
```

2D List: Example

Example: Boggle

- Generate a random board for Boggle
  - 4x4 board
  - Store the board in a 2D list
  - Each space on the board contains one randomly selected letter
  - Display the board
  - Sample Board:

S	N	K	O
V	R	E	R
I	D	I	N
N	E	G	U

Example: Boggle

```
from pprint import pprint
from random import choice

NUM_ROWS = 4
NUM_COLS = 4

board = [] # Create a new, empty board
for row in range(NUM_ROWS): # Add the correct number of rows to the board
  board.append([""]* NUM_COLS) # Append a row of size NUM_COLS

pprint(board) #pretty print the board

# Set each element in the board to a random letter
for row in range(NUM_ROWS):
  for col in range(NUM_COLS):
    board[row][col] = choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

pprint(board) # Pretty print the board
```

Tracing

Trace The Code 1:

```
def f1(list1) :
      list2 = list1
      for index in range(len(list1)):
            list2[index] = list1[index]+1
      print(list1)
      print(list2)

f1([1,2,3])
```

```
[2, 3, 4]
[2, 3, 4]
```

Trace The Code 2:

```
def f2(list1) :
      list2 = list1[:]
      for index in range(len(list1)):
            list2[index] = list1[index]+1
      print(list1)
      print(list2)

f2([1,2,3])
```

```
[1, 2, 3]
[2, 3, 4]
```

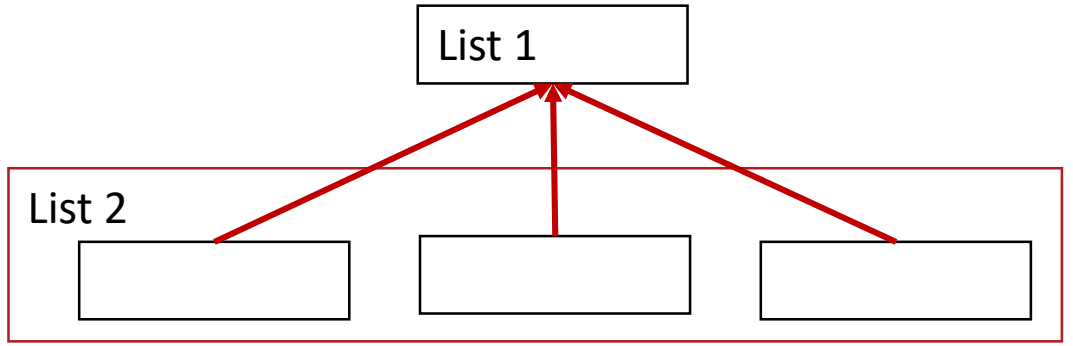
Trace The Code 3:

```
def f3(list1) :
      list2 = list1*2
      for index in range(len(list2)):
            list2[index] += 1
      print(list1)
      print(list2)

f3([1,2,3])
```

```
[1, 2, 3]
[2, 3, 4, 2, 3, 4]
```

Trace The Code 4:




```
def f4(list1) :
    list2 = [list1]*3
    for index in range(len(list2)):
        innerList = list2[index]
        for innerIndex in range(len(innerList)) :
            innerList[innerIndex] += 1
    print(list1)
    print(list2)

f4([1,2,3])
```

```
[4, 5, 6]
[[4, 5, 6], [4, 5, 6],[4, 5, 6]]
```

Trace The Code 5:

```
def f5(list1) :
    list2 = [list1]*2
    for index in range(len(list2)):
        innerList = list2[index]
        innerList = innerList[:]
        for innerIndex in range(len(innerList)) :
            innerList[innerIndex] += 1
        list2[index] = innerList
    print(list1)
    print(list2)

f5([1,2,3])
```

```
[1, 2, 3]
[[2, 3, 4], [2, 3, 4]]
```

Onward to … dictionaries.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY