# CS65: Introduction to Computer Science 

More on List

Md Alimoor Reza

## Topics

- List
- Quick recap
- List manipulation
- Appending items in a list
- List slicing
- Modifying items in a list
- Removing an item from a list
- Other methods/operations
- List of lists
- Understanding the dimensions
- Accessing elements
- Accessing with nested for loops


## List

- Sequence is an ordered group of elements (numbers, characters, etc)
- String is a type of sequence whose members are characters
- "Drake University"
-"cs65:introduction_to_computer_science!"
- List is another type of sequence whose members can be numbers, strings, or even another list!
- ["Drake University", "hello", "world"]
- [1, 2, 3, 4, 5]


## Inserting Items in a List

- Appending elements in an empty list with . append() method
- A method instructs an object to perform some action
- It is executed by a "." (dot) symbol followed the method name

```
# building list with append() function
num_list = []
num_list.append(2)
print("num_list: ", num_list)
Shell
>>> %Run lec11.py
    num_list: [2]
```


## Inserting Items in a List

- Appending elements in an empty list with .append() method

```
# building list with append() function
num_list = []
num_list.append(2)
print("num_list: ", num_list)
num_list.append(4)
print("num_list: '', num_list)
num_list.append(6)
print("num_list: ", num_list)
```

```
>>> %Run lec11.py
    num_list: [2]
    num_list: [2, 4]
    num_list: [2, 4, 6]
```


## Inserting Items in a List Iteratively

- Appending elements in an empty list with .append() method iteratively using for loop

```
# building list with append function using loop
num_list = []
for i in range(10, 100, 10):
        num_list.append(i)
print("num_list: ", num_list)
```

```
>>> %Run lec11.py
    num_list: [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

>>>

## Inserting Specific Items in a List Iteratively

- Appending odd numbers (from 1 to 20) in an empty list with .append() method iteratively using for loop

```
# building list with append function using loop
# insert only odd numbers
num_list = []
for num in range(1, 20):
    if (num % 2 != 0):
        num_list.append(num)
print("num_list: ", num_list)
```

>> \%Run lec11. py
num_list: $[1,3,5,7,9,11,13,15,17,19]$

## Exercise 1

- Appending multiples of 5 in an empty list with append() method iteratively using for loop
- Prompt the user to enter two numbers
- first number is the lower limit, eg, $\mathbf{1}$
- second number is the upper limit, eg, 15
- If user enters 1 and $\mathbf{1 5}$, you should append only 5, 10, 15

```
# building list with append function using loop
# insert only odd numbers
num_list = []
for num in range(1, 20):
    if (num % 2 != 0):
        num_list.append(num)
print("num_list: ", num_list)
```

>>> \%Run lec11. py
num_list: $[1,3,5,7,9,11,13,15,17,19]$

## Exercise: my solution

- Appending multiples of 5 in an empty list with append() function iteratively using for loop

```
# building list with append function using loop
# insert only multiples of 5
lower_limit = int(input("enter the lower limit: "))
upper_limit = int(input("enter the upper limit: "))
num_list = []
for num in range(lower_limit, upper_limit):
    if (num % 5 == 0):
        num_list.append(num)
print("num_list: ", num_list)
```

```
>>> %Run lec11.py
enter the lower limit:
enter the upper limit: 22
num_list: [5, 10, 15, 20]
>> %Run lec11.py
    enter the lower limit: 5
    enter the upper limit: 49
    num_list: [5, 10, 15, 20, 25, 30, 35, 40, 45]
```


## Quick Review: Indexing



## Quick Review: Accessing Items using Index

- Use variable_name[index] access an item in a list

```
num_list = [10, -1, -2, -3, 20, -4, 30]
print("Number at index 0 is ", num_list[0])
print("Number at index 1 is ", num_list[1])
print("Number at index 4 is ", num_list[4])
print("Number at last index is ", num_list[-1])
```

Shell
Python 3.7.9 (bundled)
>> \%Run lec11.py
Number at index 0 is 10
Number at index 1 is -1
Number at index 4 is 20
Number at last index is 30

## List Slicing

- Slice is a span of items that are taken from a list (or any sequence)
- Span is a list containing copies of elements from start up to, but not including, end
- Format: list_variable_name[start : end]
- start : starting index - if not specified 0 is used
- end : end index - if not specified len(list) is used


## List Slicing

- Format: list_variable_name[start : end]
- start : starting index - if not specified 0 is used
- end : end index - if not specified len(list) is used

```
num_list = [10, 11, 12, 13, 14, 15]
print("num_list ", num_list)
print("num_list[0] ", num_list[0])
print("num_list[:1] ", num_list[:1])
print("num_list[1:4] ", num_list[1:4])
print("num_list[1:] ", num_list[1:])
```

>> \%Run lec12.py
num_list $[10,11,12,13,14,15]$
num-list[0] 10
num_list[:1] [10]
num-list[1:4] [11, 12, 13]
num_list[1:] $[11,12,13,14,15]$

## Changing/Replacing Items in a List

- Changing specific items in a list

```
num_list = [10, -1, -2, -3, 20, -4, 30]
print("Before modification num_list is ", num_list)
num_list[1] = 0
num_list[2] = 0
num_list[3] = 0
num_list[4] = 40
```

```
>>> &Run lec12.py
    Before modification num_list is [10, -1, -2, -3, 20, -4, 30]
    After modification num_list is [10, 0, 0, 0, 40, -4, 30]
```


## Changing/Replacing Items in a List

- Replacing specific items in a list iteratively
- You must use index for loop
- value for loop would modify the variable, it won't modify the list
- Step 1: You should find the index/position of an item
- Step 2: Use the index to change/modify the value


## Recap: Value for loop vs Index for loop

- Syntax of value for loop
for var in [10, 20, 30, 40, 50]: print(var)
- There is another form called index for loop

$$
\begin{aligned}
& \text { my_list }=[10,20,30,40,50] \\
& \text { length }=\text { len }(\text { my_list }) \\
& \text { for } \mathbf{i} \text { in range }(\text { length }): \\
& \quad \operatorname{print}(\text { my_list }[\mathbf{i}])
\end{aligned}
$$

## Changing/Replacing Items in a List

- Replacing all negative numbers in a list with zeros (0s)

```
# modifying list based on a criteria
num_list = [10, -1, -2, -3, 20, -4, 30]
list_size = len(num_list)
print("Before modification num_list is ", num_list)
for i in range(list_size):
    if (num_list[i] < 0):
        #print("Found negative num at index: ", i)
        num_list[i] = 0
print("After modification num_list is ", num_list)
```


## Changing/Replacing Items in a List

- Code simplification
- Inserting the len(num_list) inside the range function

```
# modifying list based on a criteria
num_list = [10, -1, -2, -3, 20, -4, 30]
for i in range(len(num_list)):
    if (num_list[i] < 0):
        #print("Found negative num at index: ", i)
        num_list[i] = 0
```

```
>> %Run lec11.py
    Before modification num_list is [10, -1, -2, -3, 20, -4, 30]
    After modification num_\overline{l}ist is [10, 0, 0, 0, 20, 0, 30]
```


## Exercise 2

- Replacing all negative numbers in a list by making them positives

$$
\text { num_list }=[10,-1,-2,-3,20,-4,30]
$$



$$
\text { num_list }=[10,1,2,3,20,4,30]
$$

```
# modifying list based on a criteria
num_list = [10, -1, -2, -3, 20, -4, 30]
for i in range(len(num_list)):
    if (num_list[i] < 0):
```

        \#print("Found negative num at index: ", i)
    num_list[i] \(=0\)
    >>> skun lec11.py

## Removing Items from a List

- Remove a specific item from a list using value
- use .remove() method

```
# deleting an item from the list
num_list = [10, -1, -2, -3, 20, -4, 30]
print("Initial num_list is ", num_list)
num_list.remove(-1)
print("After removing -1 num_list is ", num_list)
num_list.remove(-2)
print("After removing -2 num_list is ", num_list)
```

>>> \%Run lec12.py
Initial num_list is $[10,-1,-2,-3,20,-4,30]$
After removing -1 num_list is $[10,-2,-3,20,-4,30]$
After removing -2 num_list is $[10,-3,20,-4,30]$

- Heads up! the size of the list will change! Be careful when you are removing items using for loop


## Removing Items from a List

- Remove a specific item from a list using value
- use .remove() method

```
num_list = [10, -2, -2, -2, 20, -4, 30]
num_list.remove(-2)
num_list.remove(-2)
num_list. remove(-2)
print("After removing all -2s num_list is ", num_list)
```

>>> \%Run lec12.py
After removing all -2 s num_list is $[10,20,-4,30]$

- Heads up! If the item is not in the list, you will get an error


## Removing Items from a List

- Remove a specific item from a list using index
- use .pop() method

```
# deleting an item from the list using .pop() method
num_list = [10, -1, -2, -3, 20, -4, 30]
print("Initial num_list is ", num_list)
num_list.pop(0)
print("After removing item at index 0 num_list is ", num_list)
num_list.pop(1)
print("After removing item at index 1 num_list is ", num_list)
```

```
>>> %Run Lec12.py
    Initial num_list is [10, -1, -2, -3, 20, -4, 30]
    After removing item at index 0 num_list is [ [1, -2, -3, 20, -4, 30]
    After removing item at index 1 num_list is [ [-1, -3, 20, -4, 30]
```


## Removing Items from a List

- Remove a specific item from a list using index
- use del keyword

```
# deleting an item from the list
num_list = [10, -1, -2, -3, 20, -4, 30]
print("Initial num_list is ", num_list)
del num_list[0]
print("After removing item at index 0 num_list is ", num_list)
del num_list[1]
print("After removing item at index 1 num_list is ", num_list)
```

```
>>> %Run lec12.py
    Initial num_list is [10, -1, -2, -3, 20, -4, 30]
    After removing item at index 0 num_list is [-1, -2, -3, 20, -4, 30]
    After removing item at index 1 num_list is [ [-1, -3, 20, -4, 30]
```


## Other Useful List Operations

- Sort the list items in ascending order

```
num_list = [10, 1, 2, 3, 20, 4, 30]
num_list.sort()
```

```
>>> %Run lec12.py
    Before sorting the items list is [10, 1, 2, 3, 20, 4, 30]
    After sorting the items list is [1, 2, 3, 4, 10, 20, 30]
```

- Sum of elements in the list

```
total = sum(num_list)
print("Sum of list items is ", total)
```

```
Sum of list items is 70
```


## Other Useful List Operations

| operation | meaning | result type |
| :--- | :--- | :--- |
| $x$ in $s$ | checks if an item in $s$ equals $x$ | bool |
| $x$ not in $s$ | checks if no items in s equal $x$ | bool |
| $s+t$ | concatenation (two sequences) | same seq. type |
| $s^{*} n$ (or: $\left.n^{*} s\right)$ | $n$ shallow copies of $s$, <br> concatenated | same seq. type |
| s.count(x) | find \# items in s equal to $x$ | int (\#matches) |
| s.index(x) | find index of first $x$ in $s$ <br> (if not found, crashes) | int |

## Other Useful List Operations

- list1 + list2 produces new list by concatenating list2 to end of list1
- min(list) finds the elements in the list with the smallest value
$\cdot \boldsymbol{m a x}(l i s t)$ finds the elements in the list with the largest value

| operation | meaning | returned value |
| :--- | :--- | :--- |
| s.append( x$)$ | add x as a single value at end of s. | None value |
| s.extend(t) | individually append each item of sequence t to <br> the end of s. | None value |
| s.insert(i,x) | make space (push other spots to the right), put x <br> value at location i. | None value |
| s.pop(i) | remove value at index i from sequence; return <br> the value that was there | item that was at <br> index i |
| s.remove( x$)$ | find first occurrence of x, remove it. | None |
| s.reverse() | reverse the ordering of items. | None |

## Poll

- Follow the link below and submit your answer


## https://tinyurl.com/yra6fbp3

## Understanding the 'List of Lists'

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
```

- Step 1: Understand each dimension

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
list_size_outer = len(num_list)
print("Size of the outer list ", list_size_outer)
list_size_inner0 = len(num_list[0])
print("Size of the first inner-list ", list_size_inner0)
list_size_inner1 = len(num_list[1])
print("Size of the second inner-list ", list_size_inner1)
```

```
>>> %Run lec12.py
    Size of the outer list 2
    Size of the first inner-list 3
    Size of the second inner-list 3
```


## Understanding the 'List of Lists'

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
```

- Step 2: Access element in each dimension
- outer dimension

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
print("1st element in outer-list ", num_list[0])
print("2nd element in outer-list ", num_list[1])
```

>>> \%Run lec12.py
1st element in outer-list $[1,2,3]$
2nd element in outer-list $[10,20,30]$

## Understanding the 'List of Lists'

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
```

- Step 2: Access element in each dimension - inner dimension

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
print("1st element in inner-list0 ", num_list[0] [0])
print("2nd element in inner-list0 ",' num_list[0] [1])
print("3rd element in inner-list0 "', num_list[0][2])
```

```
>>> %Run lec12.py
    1st element in inner-list0 1
    2nd element in inner-list0 2
    3rd element in inner-list0 3
```


## Recap: Nested for loops

- Putting one loop inside another
- The first loop is called the outer loop
- The second loop is called the inner loop
- Here is simpler version:

```
for i in range(3): for j in range(3):
\[
\operatorname{print}(" \mathrm{i}: ", \mathrm{i}, " \mathrm{j}: ", \mathrm{j})
\]
```


## Recap: Visualization of nested for loop



## List of Lists and Nested Index Loops

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
```

- Step 1: Create an index for each dimension
- Step 2: Nest loops
- Step 3: Access each element using indexing


## List of Lists and Nested Index Loops

```
# list of lists
num_list = [ [1, 2, 3], [10, 20, 30] ]
list_size_i = len(num_list)
for i in range(list_size_i):
    inner_list = num_list[i] # get one inner list
    list_size_j = len(inner_list) # find the size of the inner list
    for j in range(list_size_j):
        print("num_list[", i, "][", j, "]", num_list[i][j])
```

```
>>> %Run lec12.py
    num_list[ 0 ][ 0 ] 1
    num_list[ 0 ][ 1 ] 2
    num_list[ 0 ][ 2 ] 3
    num_list[ 1 ][ 0 ] 10
    num_list[ 1 ][ 1 ] 20
    num_list[ 1 ][ 2 ] 30
```


## List of Lists and Nested Index Loops

- Code simplification
- Inserting the len() inside the range function

```
num_list = [ [1, 2, 3], [10, 20, 30] ]
for i in range(len(num_list)):
    for j in range(len(num_list[i])):
        print("num_list[", i, "][", j, "]", num_list[i][j])
```

```
>>> %Run lec12.py
    num_list[ 0 ][ 0 ] 1
    num_list[ 0 ][ 1 ] 2
    num_list[ 0 ][ 2 ] 3
    num_list[ 1 ][ 0 ] 10
    num_list[ 1 ][ 1 ] 20
    num_list[ 1 ][ 2 ] 30
```


## Exercise 3

## - Find the summation of all the numbers in a list of lists

$$
\text { num_list }=[[1,2,3],[10,20,30]]
$$



```
num_list = [ [1, 2, 3], [10, 20, 30] ]
```

for $i$ in range(len(num_list)):
for $j$ in range(len(num_list[i])):
print("num_list[", i, "] [", j, "]", num_list[i][j])
>>> \%Run lec12.py
num_list[ 0 ][ 0 ] 1
num_list[ 0 ][ 1 ] 2
num list [ 0 ][ 2 ] 3

$\begin{array}{lllllll}\text { num_list } & 1 & ][ & 0 & 1 & 10 \\ \text { num_list } & 1 & ][ & 1 & ] & 20\end{array}$
num_list [ 11$]\left[\begin{array}{llll} & 2 & ]\end{array}\right.$

## Other Useful Slicing Operations

## operation

## meaning

## $s[i]=x \quad$ replace ith item of $s$ with $x$

$s[i: j]=\mathrm{replace}$ slice $i: j$ with sequence t .
(lengths needn't match!)
$s[i: j: k]=t \quad$ replace slice $i: j: k$ with sequence $t$. (lengths must match!)
del s[i] remove ith item from s.
del $s[i: j]$
remove slice i:j from s.
del $s[i: j: k] \quad$ remove slice $i: j: k$ from $s$.

