KUMARAGURU college of technology
character is life

KCED
Learning Augmented

PYTHON PROGRAMMING

Data Structure: List

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Overview of this lecture

- Introduction
- Creating lists
- Accessing list items
- Modifying lists
- List Methods
- List Function
- List Aliasing/Cloning
- List and Strings
- List and Functions
- List Comprehension
- List Processing

Elements of different kind

Elements of same kind

Introduction

- List is an object that contains multiple data items
- List is a sequence of values
- Every item in a list is called as an element
- List can contain collection of same data type or different data types.
- The list is one of the most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets.
- List are mutable, that is, their elements can be changed.

creating a list

- A list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.
- It can have any number of items and they may be of different types (integer, float, string etc.).

Method -1 without constructor

```
# empty list
my_list = []
# list of integers
my_list = [1, 2, 3]
# list with mixed datatypes
my_list = [1, "Hello", 3.4]
# nested list
my_list = [“welcome", [8, 4, 6]]
```

Method-2 using list constructor

```
# empty list
my_list = list()
# list of integers
my_list = list([1, 2, 3])
```

Creating lists – Using list constructor

- Create an empty list
  - L1 = list();
- Create a list using inbuilt range() function
  - L2 = list(range(0,6))
- Create a list of integers
  - L3 = list([10,20,40])
- Create a list of strings
  - L4 = list(["Apple", "Banana", "Grapes"])
- Create a list of characters
  - L5 = list("abcd")
  - L6 = list(['a','b','c','d'])
- Create a list of mixed type
  - L7 = list([10, "apple", 30.5])
- Create a nested list
- L8 = list(["welcome",[10,"apple",30],[1,2,3]])

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

Creating lists – without using list constructor

- Create an empty list
  - L1 = []
- Create a list of integers
  - L2 = [10,20,40]
- Create a list of strings
  - L3 = ["Apple", "Banana", "Grapes"]
- Create a list of characters
  - L4 = ['a','b','c','d']
- Create a list of mixed type
  - L5 = [10, "apple", 30.5]
- Create a nested list
  - L6 = ["welcome",[10,"apple",30],[1,2,3]]

LEARNING AUGMENTED

Accessing the elements of a list

- Indexing -Accessing individual elements using index operator [].
- Slicing – part of a list or subset of a list can be accessed using slicing.
- Traversing using loop. (for /while loop)

Accessing the elements of a list - index

- The index operator [] is used to access the elements of a list.
- The index starts from 0.
- **Example:**
  - >>> L1 = [10, 20, 30, 40, 50]
  - >>> L1
  - [10, 20, 30, 40, 50]
  - >>> L1[0]
  - 10

Items	**10**	**20**	**30**	**40**	**50**
Index	0	1	2	3	4

Accessing the elements of a list – Negative index

- The list elements can also be accessed from the backward direction using negative index.
- Negative index starts from -1 referring to the last item, and -2 to the second last item and so on.
- **Example:**
  - >>> L1 = [10, 20, 30, 40, 50]
  - >>> L1
  - [10, 20, 30, 40, 50]
  - >>> L1[0]
  - 10
  - >>> L1[-1]
  - 50
  - >>> L1[-5]
  - 10

Items	**10**	**20**	**30**	**40**	**50**
Negative Index	-5	-4	-3	-2	-1

Accessing the elements of a nested list

- **Example:**

```
>>> L1 = list([“hi",[10,"apple",30],[1,2,3]])
>>> L1[0]
‘hi'
>>> L1[0][0]
‘h'
>>> L1[1]
[10, 'apple', 30]
>>> L1[1][2]
30
>>> L1[1][1][3]
'l'
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Accessing the elements using slicing operator

- slicing operator can be used to access the part of list or subset of list.

Syntax:

List_object [start_index :end_index: step_size]

start_index - specifies starting index position of subset of list. It is optional parameter, if not specified, it takes the value 0 by default.

end_index - specifies end index position of subset list. Result excludes this index value. i.e if the end_index is 7,it access the value up to 6 and excludes the value at the position 7. It is optional parameter, if it is not specified, it takes the end of list by default.

step_size - an integer value which specifies increment step count for index position. It is optional parameter. If it is not specified, it increments the index value by 1.

KUMARAGURU college of technology character is life

Consider the list : marks= [50,60,90.5,80,100,65.5,89.5]

Example	Output	Explanation
marks[0:7]	[50, 60, 90.5, 80, 100, 65.5, 89.5]	Extracts elements from 0th index position to 6th index position.
marks[0:1]	[50]	Extracts 0th position element
marks[1:6:2]	[60, 80, 65.5]	Extracts the elements from 1st index position to 5th index position, where step size is 2. So it retrieve elements at position 1,3,5
marks[:]	[50, 60, 90.5, 80, 100, 65.5, 89.5]	If the start and end position is not specified, it retrieves the entire list
marks[2:]	[90.5, 80, 100, 65.5, 89.5]	If end position is not specified, it will take up to last element. Here 2 to 6
marks[-1:]	[89.5]	Extracts last element
marks[::-1]	[89.5, 65.5, 100, 80, 90.5, 60, 50]	Displays list in reverse order. It is the best technique to reverse the list.
marks[::5]	[50, 65.5]	Extracts the elements at 0th position and 6th position as step size is 5

Accessing a list: Traversing a list using loops

Using for loop:

Example 1:

```
>>> #create a list
>>> subjects=['Tamil','English','Physics','Chemistry','Mathematics','Computer Science']
>>> for item in subjects:    # loop iterates over the sequence(list)
        print(item)          # print the element
```

Output:

```
Tamil
English
Physics
Chemistry
Mathematics
Computer Science
```

Accessing a list: Traversing a list using loops

Using while loop:

Example 1:

```
>>> #create a list
>>> subjects=['Tamil','English','Physics','Chemistry','Mathematics','Computer Science']
>>> index=0   # intialization of index
>>> while(index<len(subjects)):  # loop iterates until index reaches last element
        print(subjects[index])  # print the element at postion specified by index variable
        index +=1               # update loop variable(index)
```

```
Tamil
English
Physics
Chemistry
Mathematics
Computer Science
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Modify a list

- Python provides way to perform the following operations
  - Change a data item in a list
  - Add one or more data item(s) to a list
  - Delete one or more data item(s) from a list

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Modify a list - Change a data item in a list

- lists are mutable datatype i.e changeable.
- List elements can be modified after it is defined.

A data item in a list can be changed by using "= operator"

Example:

- >>> marks=[90,60,80]
- >>> print(marks)
- [90, 60, 80]
- >>> marks[1]=100
- >>> print(marks)
- [90, 100, 80]

Modify a list - Add one or more data item(s) to a list

```
>>> marks=[90,100,80]
>>> marks
[90, 100, 80, 50]
```

- append() method
  - Adds one item to the end of a list

```
>>> marks.append(50)
>>> marks
[90, 100, 80, 50]
```

  - An entire list can also be appended to an existing list. It gets added as a nested list.

```
>>> L1=[45,67,89]
>>> marks.append(L1)
>>> marks
[90, 100, 80, 50, [45, 67, 89]]
```

- extend() method - Adds several items to the end of a list

```
>>> marks.extend([20,40,55])
>>> marks
[90, 100, 80, 50, 20, 40, 55]
```

- insert() method - Insert one item at a desired location of a list

```
>>> marks.insert(2,88)
>>> marks
[90, 100, 88, 80, 50, 20, 40, 55]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Modify a list - Delete one or more data item(s) from a list – remove() method

- remove() method - To remove the given item from the list

```
>>> marks = [90, 100, 80, 50, 20, 40, 55]
>>> marks.remove(80)
>>> marks
[90, 100, 50, 20, 40, 55]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

Modify a list - Delete one or more data item(s) from a list – pop() method

- pop() method
  - To remove an item at a given index and return it..
  - If index is not mentioned, the last item of the list is deleted and returned

```
>>> marks
[90, 100, 50, 20, 40, 55]
>>> marks.pop()
55
>>> marks
[90, 100, 50, 20, 40]

>>> marks.pop(2)
50
>>> marks
[90, 100, 20, 40]
```

LEARNING AUGMENTED

Modify a list - Delete one or more data item(s) from a list – del keyword, clear() method, slicing technique

- del keyword

Used to delete a data item from the list

```
>>> marks
[90, 100, 20, 40]
>>> del marks[2]
>>> marks
[90, 100, 40]
```

It can even delete the list entirely

```
>>> del marks
>>> marks
Traceback (most recent call last):
  File "<pyshell#67>", line 1, in <module>
    marks
NameError: name 'marks' is not defined
```

clear() method

- Used to empty a given list

```
>>> marks.clear()
>>> marks
[]
```

delete items in a list by assigning an empty list to a slice of elements.

```
marks=[100,20,30]
>>> marks[1:2]=[]
>>> print(marks)
[100, 30]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

List operations

- Concatenation = +
- Repetition= *
- Membership = in
- Identity = is
- Slicing [::]
- indexing

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Concatenation operator (+)

- Join or combine two lists
  - One list is appended at the end of the other list
  - Cannot concatenate a list with another data type, such as a number
- The += augmented assignment operator can also be used to concatenate lists
- Example:

```
>>> L1=[1,2,3]
>>> L2=[10,20,30]
>>> L1+L2
[1, 2, 3, 10, 20, 30]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Repetition operator (*)

- Makes multiple copies of a list and joins them together
  - The ***** symbol is a repetition operator when applied to a sequence and an integer
  - General format: *list* * *n*
- Iterate over a list using a for loop
  - Format: for *x* in *list*:
- **Example:**

```
>>> L1=[1,2,3]
>>> L1*3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Membership operator (in)

- Used to determine whether an element is in a list.
- It returns True if the element is present, and False if not present
- **Example**

```
>>> L1=[1,2,3,4,5]
>>> 3 in L1
True
>>> 30 in L1
False
```

Identity operator (is)

- To check if two variables refer to the same object.
- In python, even if two lists contains the same elements at a given time, they are not identical, but they are equivalent.
- If two objects are identical, they are also equivalent
- If two objects are equivalent, then it is not necessary that they will also be identical.
- When assignment operator is used for copying the list, then both the lists share the same id and hence they are identical

- Example:

```
>>> L1=[1,2,3,4]
>>> L2=[1,2,3,4]
>>> L1 is L2
False
>>> L2=L1
>>> L1 is L2
True
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

Python Built-in Functions for List

LEARNING AUGMENTED

Function	Description
all()	Returns True if all elements of the List are true (or if the list is empty).
any()	Returns True if any element of the List is true. If the list is empty, returns False.
len()	Returns the length (the number of items) in the list.
list()	Typecast (convert) an iterable (tuple, string, set, dictionary) object to a list.
max()	Returns the largest item in the list.
min()	Returns the smallest item in the list
sorted()	Returns a new sorted list (does not sort the list itself).
sum()	Returns the sum of all elements in the list.
reversed()	Returns an iterator that accesses the given sequence in the reverse order.

Examples

1. all()

```
>>> games=['cricket','foot ball','hockey','basket ball']  # create a list
>>> all(games) # return true if all element in list is true
True
```

2. any()

```
>>> scores=[0,50,100,10,20] # create a list
>>> any(scores) #Returns True if any element of the List is true
True
>>> scores=[0]
>>> any(scores)
False
```

3. len()

```
>>> len(games) # return the no.of items in a list
4
```

The length of the list games is 4.

4. list()

```
>>> name="Kalam" # create a string
>>> list(name)
['K', 'a', 'l', 'a', 'm']
```

Converts string to a list of characters. Any sequence datatype can be converted to list using list function.

5. max()

```
>>> scores
[0, 50, 100, 10, 20]
>>> max(scores) # returns the largest value in a list
100
```

The maximum value in a list is 100

6. min()

```
>>> min(scores) # returns the smallest value in a list
0
```

The minimum value in a list is 0

7. sorted()

```
>>> sorted_scores=sorted(scores) #Returns a new sorted list
>>> sorted_scores
[0, 10, 20, 50, 100]
>>> scores   # original list
[0, 50, 100, 10, 20]
>>> sorted(scores,reverse=True) # sort the elements in descending order
[100, 50, 20, 10, 0]
```

It doesn't sort the original list.

8. sum()

```
>>> sum(scores) #returns the sum of all elements in the list
180
```

9. reversed()

```
>>> games
['cricket', 'foot ball', 'hockey', 'basket ball']
>>> reversed(games) #Returns an iterator that accesses the list in the reverse order
<list_reverseiterator object at 0x023076D0>
>>> list(reversed(games)) # to access the list
['basket ball', 'hockey', 'foot ball', 'cricket']
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Python Built-in Methods for List

Function	Description
append()	Appends an element to the end of the list
extend()	Insert all elements of a one list to the another list/ inserts multiple elements
insert()	Insert an item at the defined index
remove()	Removes an item from the list
pop()	Removes and returns an element at the given index
clear()	Removes all items from the list
copy()	Returns a shallow copy of the list
index()	Returns the index of the first matched item
count()	Returns the number of occurrences of an element in a list
sort()	Sort elements in a list in ascending order. The list itself is modified.
reverse()	Reverse the order of items in the list

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

index() method

- It searches and find the given item in a list and returns the position of the first occurrence of the item in the list.
- **Example**

```
>>> L1=[34,56,23,78,22,56,67,23,56,89]
>>> L1.index(23)
2
>>> L1.index(22)
4
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

count() method

- Count the number of times a data items has occurred in a list and returns it.
- **Example**

```
>>> L1=[34,56,23,78,22,56,67,23,56,89]
>>> L1.count(56)
3
>>> L1.count(67)
1
>>> L1.count(100)
0
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

copy() method

- Returns a copy of the list, and it doesn't modify the original list when any changes are made to the copied list
- Called as shallow copy
- Example

```
>>> L1=[2,4,6,8]
>>> L2=L1.copy()
>>> L2
[2, 4, 6, 8]
>>> L2.append(9)
>>> L2
[2, 4, 6, 8, 9]
>>> L1
[2, 4, 6, 8]
```

- Note: If the original list must also to be modified, then use "=" operator and copy the list
- Example

```
>>> L1=[5,7,8,9]
>>> L2=L1
>>> L2
[5, 7, 8, 9]
>>> L1.append(10)
>>> L1
[5, 7, 8, 9, 10]
>>> L2
[5, 7, 8, 9, 10]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

sort() method

- Used the sort the elements of a list in a specific order, either ascending or descending.
- It doesn't return anything.
- It performs sorting and updates the list
- Sort in ascending order:
  - list.sort()
- Sort in descending order:
  - list.sort(reverse=True)

- Example:

```
>>> L1=[34,6,13,98,65,2]
>>> L1.sort()
>>> L1
[2, 6, 13, 34, 65, 98]
>>> L1.sort(reverse=True)
>>> L1
[98, 65, 34, 13, 6, 2]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

reverse() Method

- Used to reverse the elements in the list
- It doesn't return anything.
- It reverses the elements and updates the list
- **Example**

```
>>> L1=[34,6,13,98,65,2]
>>> L1.reverse()
>>> L1
[2, 65, 98, 13, 6, 34]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

List Aliasing

- An alias is a second name for a piece of data Often easier (and more useful) than making a second copy.
- If the data is immutable, aliases don't matter, Because the data can't change.
- But if data can change, aliases can result in a lot of hard-to-find bugs.
- Aliasing happens whenever one variable's value is assigned to another variable.

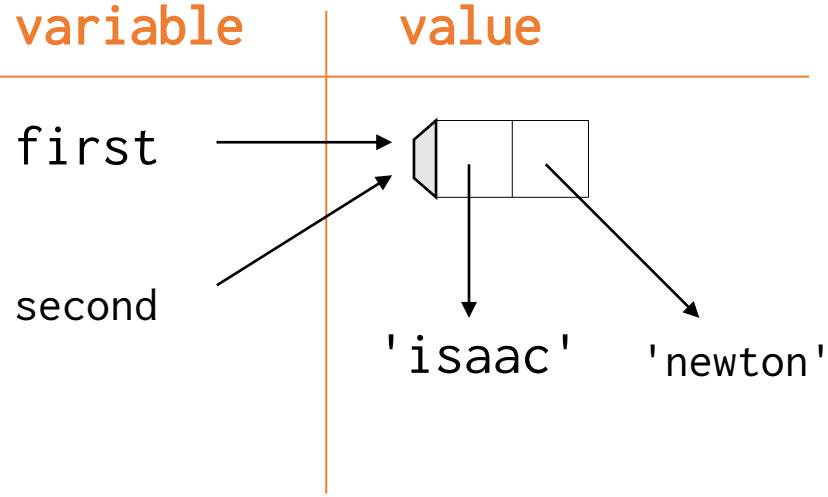
But lists are mutable

```
first = ['isaac']
second = first
first = first.append('newton')
print(first)
['isaac', 'newton']
print(second)
['isaac', 'newton']
```

Didn't explicitly

modify second

A *side effect*

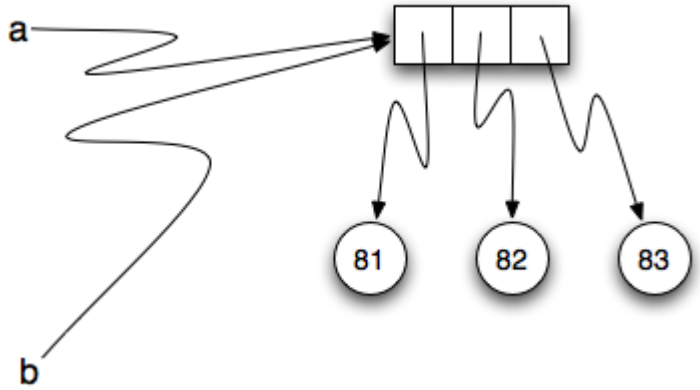


No Aliasing	Aliasing
`first = []` `second = []`	`first = []` `second = first`

Create a list aliasing

```
a = [81, 82, 83]
b = a
print(a is b)
True
```

In this case, the reference diagram looks like this:



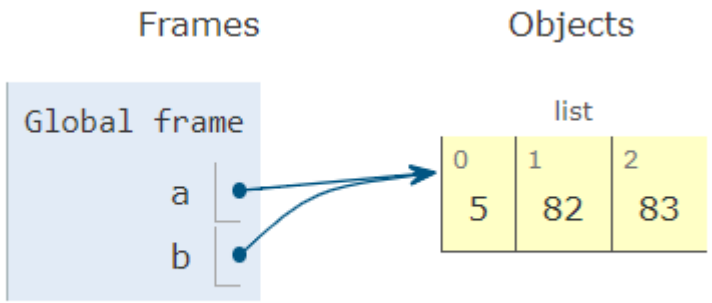
Because the same list has two different names, a and b, we say that it is aliased. Changes made with one alias affect the other.

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

Example

```
1  a = [81, 82, 83]
2  b = [81, 82, 83]
3
4  print(a == b)
5  print(a is b)
6
7  b = a
8  print(a == b)
9  print(a is b)
10
11 b[0] = 5
12 print(a)
```



Program output:

```
True
False
True
True
[5, 82, 83]
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

List Cloning?

- If we want to modify a list and also keep a copy of the original, we need to be able to make a copy of the list itself, not just the reference.
- This process is sometimes called **cloning**, to avoid the ambiguity of the word copy.
- The easiest way to clone a list is to use the slice operator.
- Taking any slice of a creates a new list.
- In this case the slice happens to consist of the whole list.

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

List Cloning?

- To copy a list, following methods can be used.

Slice operator:

- Two operands of slice operator are index of start and end of slice.
- If not explicitly used, both default to start end of sequence.
- We can take advantage of this feature.

```
>>> L1 = [1,2,3,4]
>>> L2 = L1[:]
>>> L1
[1, 2, 3, 4]
>>> L2
[1, 2, 3, 4]
```

```
>>> id(L1)
185117025160
>>> id(L2)
185117171592
```

KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Another method is to use built-in **list() method**.

```
>>> L1 =[ 1,2,3,4]
>>> L2 = list(L1)
>>> L1
[1, 2, 3, 4]
>>> L2
[1, 2, 3, 4]
>>> id(L1)
185117295816
>>> id(L2)
185117209352
```

List Cloning-using copy

- In addition to the list copy() method, the copy module can be used
- The copy module of Python's standard library contains functions for shallow and deep copy of objects.
- While deep copy is nested copying, in shallow copy, inner list copied by reference only

```
>>> import copy
>>> L1 = [1,2,3,4]
>>> L2 = copy.copy(L1)
>>> L1
[1, 2, 3, 4]
>>> L2
[1, 2, 3, 4]

>>> id(L1)
185117025160
>>> id(L2)
185117295880

>>> L3=copy.deepcopy(L1)
>>> L3
[1, 2, 3, 4]
>>> id(L3)
185117304328
```

Copy()

Example

```
>>> L1=[2,4,6,8]
>>> L2=L1.copy()
>>> L2
[2, 4, 6, 8]
>>> L2.append(9)
>>> L2
[2, 4, 6, 8, 9]
>>> L1
[2, 4, 6, 8]
```

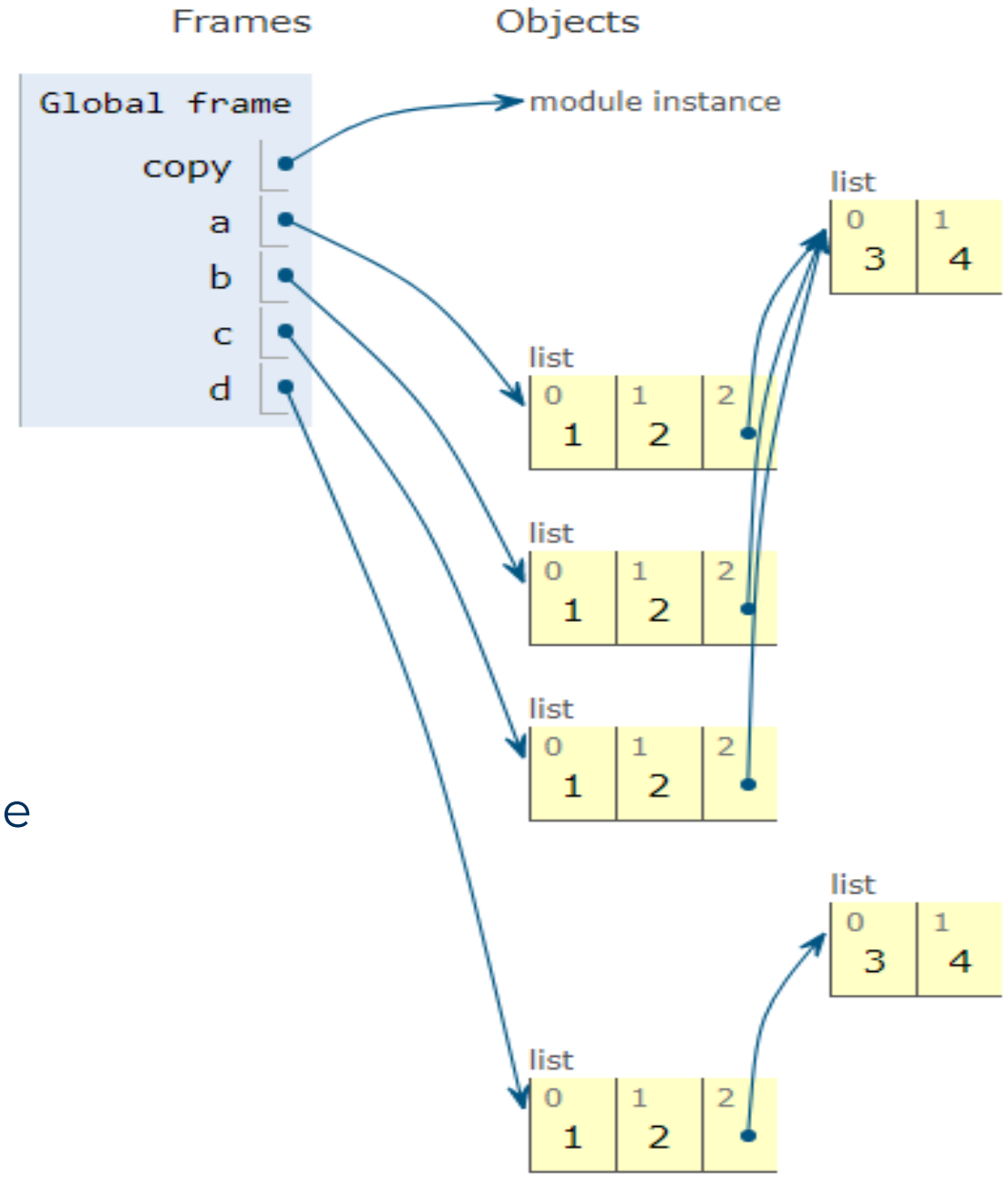
KUMARAGURU college of technology
character is life

KCED
Learning Augmented

List Cloning

LEARNING AUGMENTED

```
import copy
a=[1,2,[3,4]]
# using copy() of list object
b=a.copy()
# using copy() from copy module
c=copy.copy(a)
# using deepcopy() from copy module
d=copy.deepcopy(a)
```



KUMARAGURU college of technology
character is life

KCED
Learning Augmented

LEARNING AUGMENTED

Example Programs on Lists

Computer shop has list of products available in stock. When a user request any product, your program has to respond the availability of the product.

```
#Program to search an element in a list
products=['keyboard','mouse','monitor','processor','speaker','web camera']  # list
item=input("Enter the item to check the availability:") # get the item to be searched
if item in products: # check if the item is available
    print("Yes!!!!",item, "is available")
else:
    print("Sorry!!!!",item, "is not available")
```

Output 1:

Enter the item to check the availability: speaker
Yes!!!! speaker is available

Let's try with other input

Output 2:

Enter the item to check the availability: laptop
Sorry!!!! laptop is not available

1. The marks of “computer science” subject scored by the I BE A2 section is stored in a list in the rollno order i.e 0th index element represents marks of student with rollno1 and 1st index element represents rollno 2,.. so on.

Write a program to find the first, last, total and average marks of the class. Also find the Rollno of student who scored 1st mark. Then rearrange the marks (highest to lowest) and store it in a new list.

Program

```
#Mark list manipulation


cs_mark = [90,80,91.5,80.5,85,60,70,80,90,50,70,85,95,80,82]
highest_mark=max(cs_mark) #find the highest mark
lowest_mark=min(cs_mark)  # find the lowest mark
first_rollno=cs_mark.index(highest_mark) # find the student who got highest mark
total_mark=sum(cs_mark) # find the total marks
avg_mark=total_mark/len(cs_mark) # find the avg mark of a class
sorted_marks=sorted(cs_mark, reverse=True) # sort the list is descending order
print("Highest mark is: ",highest_mark)
print("Lowest mark is: ",lowest_mark)
print("Roll No: ",first_rollno+1, "got first mark")
print("Total mark is: ",total_mark)
print("Average mark is: ",round(avg_mark,2))
print("The original list:\n",cs_mark)
print("The sorted list is: \n",sorted_marks)
```

Output:

```
Highest mark is:  95
Lowest mark is:  50
Roll No:  13 got first mark
Total mark is:  1189.0
Average mark is:  79.27
The original list:
[90, 80, 91.5, 80.5, 85, 60, 70, 80, 90, 50, 70, 85, 95, 80, 82]
The sorted list is:
[95, 91.5, 90, 90, 85, 85, 82, 80.5, 80, 80, 80, 70, 70, 60, 50]
```

Write a function check_duplicate (list) which returns True, if a list contains duplicate words and False if all the words in the list are unique.

```
# program to check duplicate elements in a list
#function to check duplicate elements
def check_duplicate(list1):
    for i in list1:
        if list1.count(i)==1:
            continue
        else:
            return True
    return False

list1=input("Enter the sentence: ").split() # list input at runtime
duplicate_present=check_duplicate(list1) # function calling
if duplicate_present:
    print("The sentence contains duplicate words")
else:
    print("The sentence doesn't contains duplicate words")
```

Output 1:

Enter the sentence: Betty bought some butter but the butter was bitter so Betty bought some better butter to make the bitter butter better
The sentence contains duplicate words

Write a program to reverse the list elements. Get the list elements at runtime.

Without using built-in function	Using built-in function

L,rev_list=[],[]
#creating list at runtime
n=int(input("Enter the no.of elements: "))
for i in range(n):
  element=eval(input("Enter element of any type: "))
  L.append(element)
print("The original list is:",L)
for index in range(n-1,-1,-1):
  rev_list.append(L[index])
print("The reversed list is:", rev_list)
	L=[]
	#creating list at runtime
	n=int(input("Enter the no.of elements: "))
	for i in range(n):
	  element=eval(input("Enter element of any type: "))
	  L.append(element)
	print("The original list is:",L)
	rev_list=list(reversed(L))
	print("The reversed list is:", rev_list)

Note: you can use L.reverse() which modifies the original list itself

Output:

```
Enter the no.of elements: 5
Enter element of any type: "Python"
Enter element of any type: "Version"
Enter element of any type: 3.7
Enter element of any type: "Easy to learn"
Enter element of any type: 100
The original list is: ['Python', 'Version', 3.7, 'Easy to learn', 100]
The reversed list is: [100, 'Easy to learn', 3.7, 'Version', 'Python']
```

Summary

- A list is a ordered sequence of elements or items of any data type
- List is a mutable data structure i.e., you can change the elements of a list in place
- List elements can be accessed using indexing.
- We can traverse the list using for and while loops.
- List concatenation and repetition is possible by using the operators ‘+’ and ‘*’ respectively.
- The built-in methods append(),extend(),insert() are used to insert elements in the existing list.
- The keyword del and built-in methods pop(),remove() are used to delete the elements from list.
- The built-in functions len(),min(),max(),sorted(),reversed() are applicable for list objects.
- Two lists are said to be identical if they refers to the same list object.

Tips

Blue light is basically the rays that are emitted by the digital screen, which may cause severe health problems. Flux software for PCs, Twilight app for Smartphones can act as blue light filters.

Microcosm
The Co-existing Architecture

Kumaraguru Eco Campus

Annually, the **4200 trees** inside the Kumaraguru Campus, on an average, releases **10,92,000 pounds** of oxygen inhaled by **42,000 humans**.

Wellness 360

Find ways to structure and optimize your time for when and where you learn best and keep your **learning on track**

KUMARAGURU college of technology

KCED
Learning Augmented