



SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF CSE (IoT & CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)



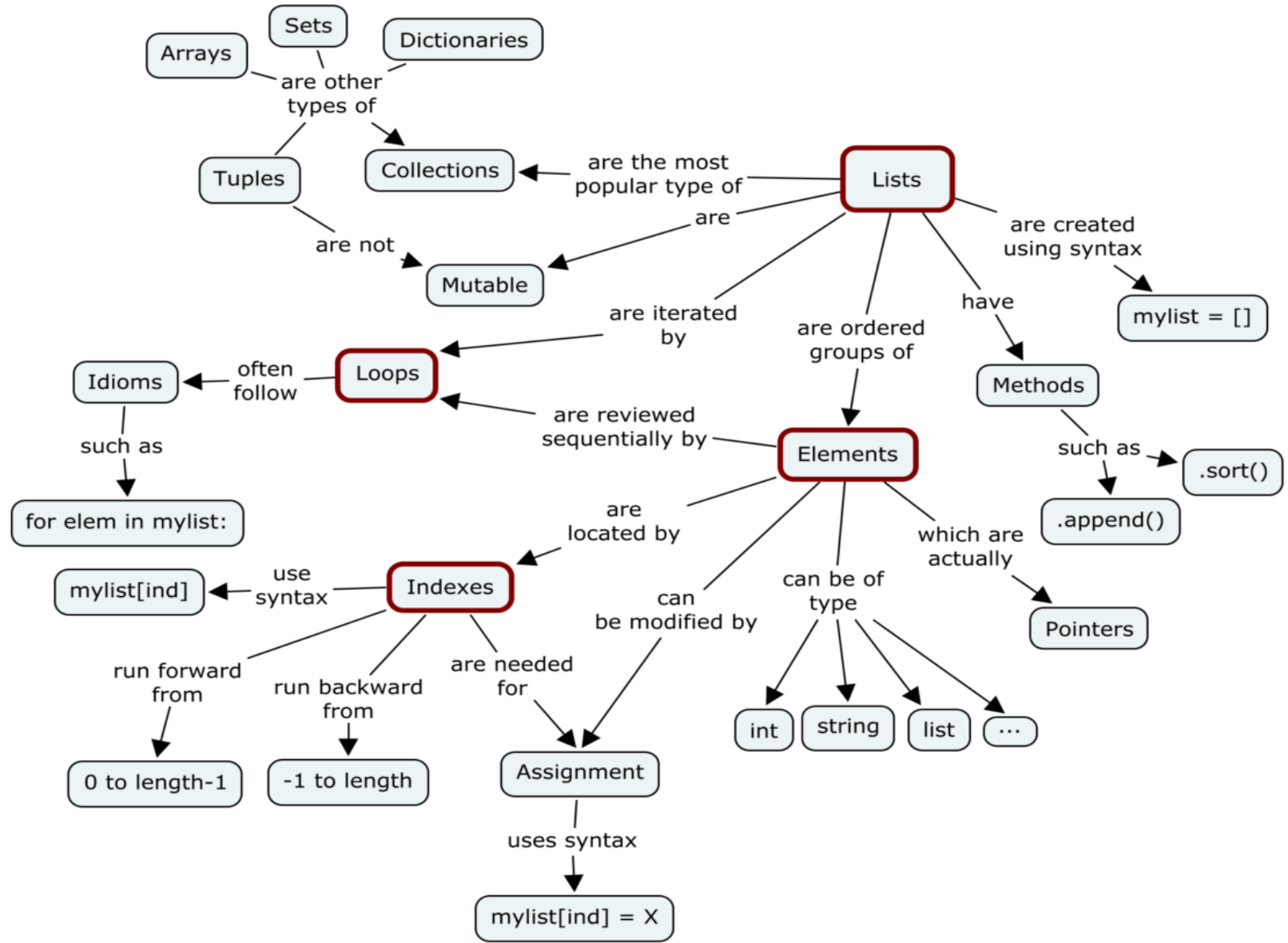
19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

- ❖ A readable, dynamic, pleasant, flexible, fast and powerful language

Session wise Agenda

- **session 1 - List (Operations, Slice, Methods)**
- Session 2 - List (Loop, Mutability)
- Session 3 - List (Aliasing, Cloning, Parameters)
- Session 4 - Tuples (Assignment, as return value)
- Session 5 - Dictionaries (operations and methods)
- Session 6 - Advance List processing, List Comprehension
- Session 7 - Simple Sort, Histogram
- Session 8 - Student Mark Statement
- Session 9 - Retail Bill preparation

Overview of Unit 4



List

- A list is a sequence of values. In a string, the values are characters; in a list, they can be any type.
- The values in a list are called elements or sometimes items.
- It can be written as a list of comma-separated items (values) between square brackets [].
- Items in the lists can be of different data types.
- Lists are used to store multiple items in a single variable.

Creating a List

```
# Creating a List
List = []
print("Blank List: ")
print(List)

# Creating a List of numbers
List = [10, 20, 14]
print("\nList of numbers: ")
print(List)
```

```
# Creating a List with
# mixed type of values
# (Having numbers and strings)
List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
print("\nList with the use of Mixed Values: ")
print(List)
```

Creating a List

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed data types
my_list = [1, "Hello", 3.4]
```



Contd..

Operations on list:

1. Indexing
2. Slicing
3. Concatenation
4. Repetitions
5. Updating
6. Membership
7. Comparison

Adding Elements to a List

```
# Creating a List
List = []
print("Initial blank List: ")
print(List)

# Addition of Elements
# in the List
List.append(1)
List.append(2)
List.append(4)
print("\nList after Addition of Three elements: ")
print(List)
```


Adding Elements to a List

```
# Adding elements to the List
# using Iterator
for i in range(1, 4):
    List.append(i)
print("\nList after Addition of elements from 1-3: ")
print(List)
```

```
# Creating a List
List = [1,2,3,4]
print("Initial List: ")
print(List)

# Addition of Element at
# specific Position
# (using Insert Method)
List.insert(3, 12)
List.insert(0, 'Geeks')
print("\nList after performing Insert Operation: ")
print(List)
```

Adding Elements to a List

```
# Creating a List
List = [1,2,3,4]
print("Initial List: ")
print(List)

# Addition of multiple elements
# to the List at the end
# (using Extend Method)
List.extend([8, 'Geeks', 'Always'])
print("\nList after performing Extend Operation: ")
print(List)
```

Adding Elements to a List

```
list = [1, 2, 3, 4, 5, 6]
```

```
print(list)
```

```
# It will assign value to the value to the second index
```

```
list[2] = 10
```

```
print(list)
```

```
# Adding multiple-element
```

```
list[1:3] = [89, 78]
```

```
print(list)
```

```
# It will add value at the end of the list
```

```
list[-1] = 25
```

```
print(list)
```

List Indexing

```
# List indexing  
  
my_list = ['p', 'r', 'o', 'b', 'e']  
  
print(my_list[0]) # p  
  
print(my_list[2]) # o  
  
print(my_list[4]) # e
```

List Indexing

Negative indexing

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

```
# Negative indexing in lists
my_list = ['p','r','o','b','e']

print(my_list[-1])

print(my_list[-5])
```

List Slicing

In Python, list slicing is a common practice and it is the most used technique for programmers to solve efficient problems. Consider a python list, In-order to access a range of elements in a list, you need to slice a list.

Two ways of slicing the List:

- Using :: Slice Operator
- Using slice()

List Slicing – Using :: operator

```
# List slicing in Python

my_list = ['p','r','o','g','r','a','m','i','z']

# includes element at index 2, 3, 4
# excludes element at index 5
print(my_list[2:5])

# elements beginning to 4th
print(my_list[:-5])

# elements 6th to end
print(my_list[5:])

# elements beginning to end
print(my_list[:])
```

List Slicing

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5
---	---	---	---	---	---

List[0] = 0

List[0:] = [0,1,2,3,4,5]

List[1] = 1

List[:] = [0,1,2,3,4,5]

List[2] = 2

List[2:4] = [2, 3]

List[3] = 3

List[1:3] = [1, 2]

List[4] = 4

List[:4] = [0, 1, 2, 3]

List[5] = 5

List Slicing

```
list = [1,2,3,4,5,6,7]
```

```
print(list[0])
```

```
print(list[1])
```

```
print(list[2])
```

```
print(list[3])
```

```
# Slicing the elements
```

```
print(list[0:6])
```

```
# By default the index value is 0 so its starts from the 0th element and go for index -1.
```

```
print(list[:])
```

```
print(list[2:5])
```

```
print(list[1:6:2])
```

List Slicing

```
list = [1,2,3,4,5]
```

```
print(list[-1])
```

```
print(list[-3:])
```

```
print(list[:-1])
```

```
print(list[-3:-1])
```

List slicing – Using slice()

- A slice object is used to specify how to slice a sequence. You can specify where to start the slicing, and where to end. You can also specify the step, which allows you to e.g. slice only every other item.

Syntax

```
slice(start, end, step)
```

List slicing – Using slice()

```
nums = [1,2,3,4,5,6,7,8,9,10]
```

```
portion1 = slice(9)
```

```
portion2 = slice(2, 8, 2)
```

```
print('List value: ', nums[portion1])
```

```
print('List value: ', nums[portion2])
```

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

```
List value: [3, 5, 7]
```

Concatenation & Repetition

- We can also use + operator to combine two lists. This is also called concatenation.
- The * operator repeats a list for the given number of times.

```
# Concatenating and repeating lists
odd = [1, 3, 5]

print(odd + [9, 7, 5])

print(["re"] * 3)
```

Delete / Remove List

```
# Deleting list items
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

# delete one item
del my_list[2]

print(my_list)

# delete multiple items
del my_list[1:5]

print(my_list)

# delete entire list
del my_list

# Error: List not defined
print(my_list)
```

Contd..

```
thislist = ["apple", "banana", "cherry"]  
thislist.remove("banana")  
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]  
thislist.pop(1)  
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]  
thislist.pop()  
print(thislist)
```






















Contd..

```
thislist = ["apple", "banana", "cherry"]  
del thislist[0]  
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]  
del thislist
```


List Methods

Python List Methods

Input	Method	Output
	<code>.append()</code>	
	<code>.insert(1, )</code>	
	<code>.pop(1)</code>	
	<code>.remove()</code>	
	<code>.reverse()</code>	
	<code>.sort()</code>	
	<code>.index()</code>	2
	<code>.count()</code>	2
	<code>.copy()</code>	

List Methods

Method	Description
<code><u>append()</u></code>	Adds an element at the end of the list
<code><u>clear()</u></code>	Removes all the elements from the list
<code><u>copy()</u></code>	Returns a copy of the list
<code><u>count()</u></code>	Returns the number of elements with the specified value
<hr/>	
<code><u>extend()</u></code>	Add the elements of a list (or any iterable), to the end of the current list
<code><u>index()</u></code>	Returns the index of the first element with the specified value
<code><u>insert()</u></code>	Adds an element at the specified position
<code><u>pop()</u></code>	Removes the element at the specified position

List Methods

remove() Removes the item with the specified value

reverse() Reverses the order of the list

sort() Sorts the list

List Methods

Intent	Method / Operation	Description
Initialize methods	<i>none</i>	initialize an empty list, using a tuple, using another list
<i>operations</i>	<code>[]</code> , <code>list()</code> or <code>list(sequence)</code> , the <code>=</code> operator	<code>b1 = a1</code> initializes <code>b1</code> as an alias for the <code>a1</code> list object
Access methods	<code>[idx]</code> , <code>.find(elem)</code> , <code>.index(elem)</code> <code>.count()</code>	<code>find</code> returns <code>-1</code> , <code>index</code> throws an exception if not present
<i>operations</i>	<code>in</code> and <code>not in</code> , <code>any</code> and <code>all</code> , <code>max</code> , <code>min</code> , <code>len</code> , <code>sum</code>	membership <code>in</code> list, <code>all</code> or <code>any</code> if element(s) is <code>True</code>
Modify methods	<i>addition</i> <code>.append(val)</code> , <code>.insert(loc, val)</code> , <code>.extend(lst)</code>	add elements to the list at specific locations; <code>extend</code> adds multiple elements from sequence
	<i>extraction</i> <code>.pop(loc)</code> , <code>.remove(elem)</code> , <code>.clear()</code>	take out from specified index, or element or all elements
	<i>ordering</i> <code>.reverse()</code> , <code>.sort()</code>	rearrange elements in the list

List Methods

<i>operations</i>	<code>del</code> , <code>sorted</code>	same effect as the methods with better performance?
Allocate methods	<code>.copy()</code> , <code>[:]</code> , <code>[s:s:s]</code> , repetition (using <code>*</code>) and concatenation (using <code>+</code>)	Create distinct list objects by duplicating existing ones
<i>operations</i>	<code>slice(start, stop, step)</code> , <code>zip</code> , <code>enumerate</code> , cloning (<code>copy.copy</code> and <code>deepcopy</code>) and list comprehension	same as slice operator <code>[]</code> , <code>zip</code> creates a new list with tuples from two lists, <code>enumerate</code> provides list with (<i>index, element</i>) tuple

Memcodes – Flashcards

<https://www.memcode.com/courses/5627>

Summary

- List is a sequence data type which can hold heterogenous data
- List have following operations
 - Indexing
 - Slicing
 - Concatenation
 - Repetition
 - Updation
 - Membership
 - Comparison

Summary

- List have in-built methods to perform various operations such as insert, delete, finding the length, etc.,
- List can be splitted in 8 ways. This process is called slicing.



THANK YOU