Date: 10/Aug/2024

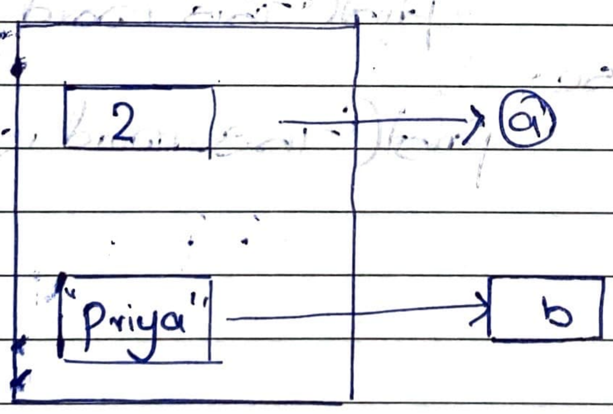
Day 11

Internal Working of Python

Heap	} Free Memory (Dynamic)
Stack	} Function calls and Local Variable
Static/Global	} Global Variable
Code	} Instruction

Application Memory

Memory Heap



a = 2
b = "priya"

papergrid

Date: / /

① Python → Dynamic Memory Allocation

Dynamic list

lst =

24	47	76	64	42	65	67
0	1	2	3	4	5	6

n=6

lst.append(67)

Linear Data Structure

lst

0	24	-10
1	47	-9
2	76	-8
3	64	-7
4	42	-6
5	65	-5
6	67	-4
7	74	-3
8		-2
9		-1

(mapping {Key : Value})

Index num   user

Lst(2)=76
or
lst(-5)=76
} Random Acess → O(1)

list.append(100)

Garbage Collection

new list

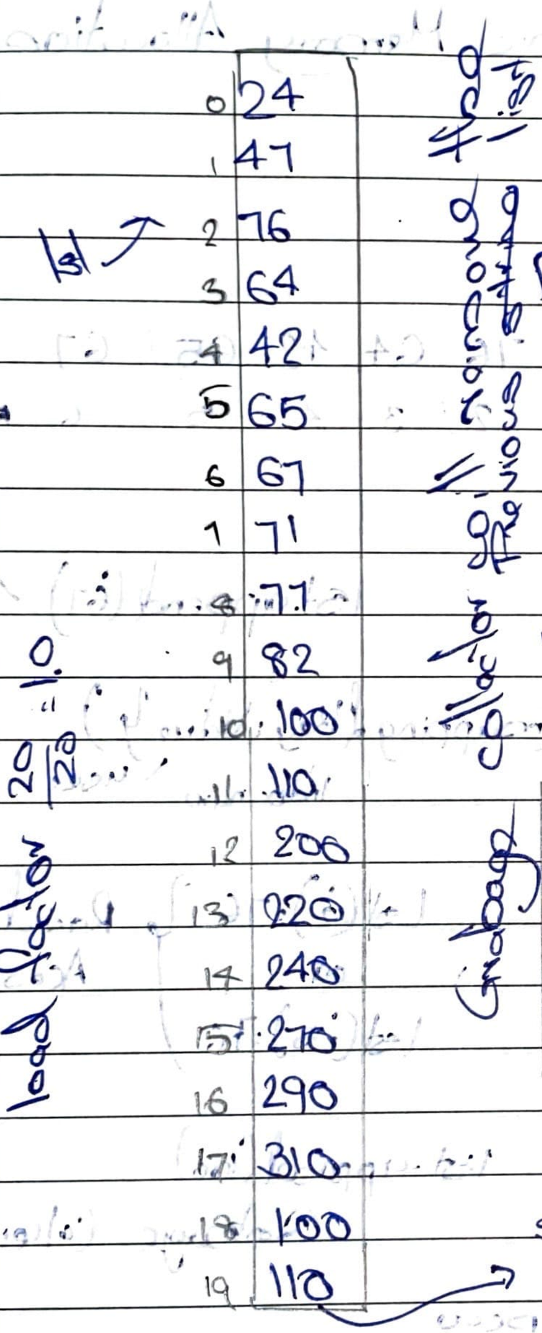
n*2 ⟹ 10 * 2 = 20

copy the existing List of element

Load factor → $\frac{8}{10}$ = 0.8

papergrid

Date: / /



0	24
1	47
2	76
3	64
4	42
5	65
6	67
7	71
8	77
9	82
10	100
11	110
12	200
13	220
14	240
15	270
16	290
17	310
18	400
19	110

Dynamic list
↓
Expensive task &
too much time

lst.append (120)

which automatically
delet the old string
and create new
with bigger string

same process repeat

③ Data Structures

→ way to store the data in an
efficient manner

Linear	Non-Linear
-Sequential manner	-Hierarchial manner
-Array / List, Stack, Queue, Linked List ↓ Website	-Tree, Graphs, Tries ↓ Networking

-Data structure in Python

1. List ⟶ Reverse
2. Tuple
3. Set
4. Dictionary

Inplace → not using any extra data structure to return the result

1. List : Mutable (can be changed)

2. Tuple : Immutable

3. Set : Remove duplicate value.