Diploma Engineering

Laboratory Manual

Data Structure with Python
Course Code: 4331601

Department of Information Technology

Semester: 3

Enrolment No	
Name	
Branch	
Academic Term	
Institute	

Directorate Of Technical Education

Gandhinagar - Gujarat

DTE's Vision:

- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls' education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

DTE's Mission:

Institute's Vision:

Student should write

Institute's Mission:

Student should write

Department's Vision:

Student should write

Department's Mission:

Student should write

Certificate

This is to certify that Mr./Ms …… Enrollment No. …… of ……. Semester of *Diploma in*……..of ……. (GTU Code) has satisfactorily completed the term work in course ……for the academic year: …… Term: Odd.

Place:……..

Date: ……..

Signature of Course Faculty **Head of the Department**

Preface

The primary aim of any laboratory/Practical/field work is enhancement of required skills as well as creative ability amongst students to solve real time problems by developing relevant competencies in psychomotor domain. Keeping in view, GTU has designed competency focused outcome-based curriculum -2021 (COGC-2021) for Diploma engineering programmes. In this more time is allotted to practical work than theory. It shows importance of enhancement of skills amongst students and it pays attention to utilize every second of time allotted for practical amongst Students, Instructors and Lecturers to achieve relevant outcomes by performing rather than writing practice in study type. It is essential for effective implementation of competency focused outcome- based Green curriculum-2021. Every practical has been keenly designed to serve as a tool to develop & enhance relevant industry needed competency in each and every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual has been designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual, students can read procedure one day in advance to actual performance day of practical experiment which generates interest and also, they can have idea of judgement of magnitude prior to performance. This in turn enhances predetermined outcomes amongst students. Each and every Experiment /Practical in this manual begins by competency, industry relevant skills, course outcomes as well as practical outcomes which serve as a key role for doing the practical. The students will also have a clear idea of safety and necessary precautions to be taken while performing experiment.

This manual also provides guidelines to lecturers to facilitate student-centric lab activities for each practical/experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve outcomes. It also gives an idea that how students will be assessed by providing Rubrics.

Development of application systems and software that use underlying architecture of machines efficiently and effectively requires the ability to use and manipulate various types of Data Structures and other constructs. This being a fundamental ability which is language neutral yet requires use of a language for its implementation. As far as data structures are concerned, the course covers Python dictionaries as well as classes and objects for defining user defined data types such as linked lists and binary search trees. It is designed to develop an integrated ability to efficient software development and apply the knowledge to various application systems; hence this course is very important for IT diploma engineers.

Although we try our level best to design this lab manual, but always there are chances of improvement. We welcome any suggestions for improvement.

Programme Outcomes (POs):

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the *engineering* problems.
2. **Problem analysis**: Identify and analyse well-defined *engineering* problems using codified standard methods.
3. **Design/ development of solutions:** Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
4. **Engineering Tools, Experimentation and Testing:** Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.
5. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.
6. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
7. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes *in field of engineering.*

Practical Outcome - Course Outcome matrix

Course Outcomes (COs):

CO1: Understand linear and non-linear data structures.
CO2: Implement Object Oriented Programming concepts in Python.
CO3: Implement basic data structures such as stacks, queues and linked lists.
CO4: Apply Algorithms for solving problems like searching and sorting of data.
CO5: *Implement non linear data structures like trees.*

S. No.	Practical Outcome/Title of experiment	CO1	CO2	CO3	CO4	CO5
1	Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.	✔				
2	Build a program to count the frequency of words appearing in a string using a dictionary.	✔				
3	Implement a Program for two matrix multiplication using simple nested loop and numpy module.	✔				
4	Implement basic operations on arrays.	✔				
5	Design an employee class for reading and displaying the employee information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be a private method.		✔			
6	Design a class Complex for adding the two complex numbers and also show the use of constructor.		✔			
7	Design a class for single level inheritance using public and private type derivation.		✔			
8	Implement multiple and hierarchical inheritance.		✔			
9	Write a Python program to demonstrate method overriding using inheritance.		✔			
10	Implement push and pop algorithms of stack using list.			✔		

11	Implement a program to convert infix notation to postfix notation using stack.			✔		
12	Implement recursive functions.			✔		
13	Implement a program to implement QUEUE using list that performs following operations: ENQUEUE, DEQUEUE, DISPLAY			✔		
14	Implement program to perform following operation on singly linked list: a. Insert a node at the beginning of a singly linked list. b. Insert a node at the end of a singly linked list. c. Insert a node after the given node of a singly linked list. d. Insert a node before the given node of singly linked list. e. Delete a node from the beginning of a singly linked list. f. Delete a node from the end of a singly linked list. g. Count the number of nodes of a singly linked list. h. Display content of singly linked list			✔		
15	Implement a python program to search a particular element from list using Linear and Binary Search.				✔	
16	Implement Bubble sort algorithm.				✔	
17	Implement Selection sort and Insertion sort algorithm.				✔	
18	Implement Merge sort algorithm.				✔	
19	Implement construction of binary search trees.					✔

20	Write a menu driven program to perform following operation on Binary Search Tree: a. Create a BST. b. Insert an element in BST. c. Pre-order traversal of BST. d. In-order traversal of BST. e. Post-order traversal of BST. f. Delete an element from BST					✔

Industry Relevant Skills

The following industry relevant skills are expected to be developed in the students by performance of experiments of this course.

- *Student will attain expertise in object oriented concept using python.*
- *Students will be able to write code quickly and efficiently while achieving complex tasks.*

Guidelines to Course Faculty

1. Couse faculty should demonstrate experiment with all necessary implementation strategies described in curriculum.
2. Couse faculty should explain industrial relevance before starting of each experiment.
3. Course faculty should Involve & give opportunity to all students for hands on experience.
4. Course faculty should ensure mentioned skills are developed in the students by asking.
5. Utilise 2 hrs of lab hours effectively and ensure completion of write up with quiz also.
6. Encourage peer to peer learning by doing same experiment through fast learners.

Instructions for Students

1. Organize the work in the group and make record of all observations.
2. Students shall develop maintenance skill as expected by industries.
3. Student shall attempt to develop related hand-on skills and build confidence.
4. Student shall develop the habits of evolving more ideas, innovations, skills etc.
5. Student shall refer technical magazines and data books.
6. Student should develop habit to submit the practical on date and time.
7. Student should well prepare while submitting write-up of exercise.

Continuous Assessment Sheet

Enrolment No: **Name:**

Name: **Term:**

Sr no	Practical Outcome/Title of experiment	Page	Date	Marks (25)	Sign
1	Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.				
2	Build a program to count the frequency of words appearing in a string using a dictionary.				
3	Implement a Program for two matrix multiplication using simple nested loop and numpy module.				
4	Implement basic operations on arrays.				
5	Design an employee class for reading and displaying the employee information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be a private method.				
6	Design a class Complex for adding the two complex numbers and also show the use of constructor.				
7	Design a class for single level inheritance using public and private type derivation.				
8	Implement multiple and hierarchical inheritance.				
9	Write a Python program to demonstrate method overriding using inheritance.				
10	Implement push and pop algorithms of stack using list.				
11	Implement a program to convert infix notation to postfix notation using stack.				
12	Implement recursive functions.				
13	Implement a program to implement QUEUE using list that performs following operations: ENQUEUE, DEQUEUE, DISPLAY				

14	Implement program to perform following operation on singly linked list: a. Insert a node at the beginning of a singly linked list. b. Insert a node at the end of a singly linked list. c. Insert a node after the given node of a singly linked list. d. Insert a node before the given node of singly linked list. e. Delete a node from the beginning of a singly linked list. f. Delete a node from the end of a singly linked list. g. Count the number of nodes of a singly linked list. h. Display content of singly linked list				
15	Implement a python program to search a particular element from list using Linear and Binary Search.				
16	Implement Bubble sort algorithm.				
17	Implement Selection sort and Insertion sort algorithm.				
18	Implement Merge sort algorithm.				
19	Implement construction of binary search trees.				
20	Write a menu driven program to perform following operation on Binary Search Tree: a. Create a BST. b. Insert an element in BST. c. Pre-order traversal of BST. d. In-order traversal of BST. e. Post-order traversal of BST. f. Delete an element from BST				

Date: ……

Practical No.1: Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.

A. Objective:

Apply various list functions to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using list functions for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO1: Understand linear and non-linear data structures.

E. Practical Outcome(PRO)

Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Python List functions:

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

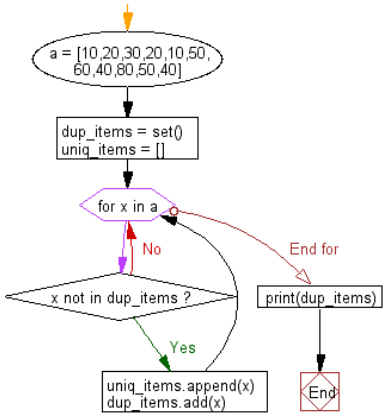
```
# Empty list
my_list = []

# List with elements
my_list = [1, 2, 3, 4, 5]

# List with different data types
my_list = [1, "hello", 3.14, True]

# Nested list
my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

the **remove_duplicates function** takes a list as input and iterates over each element. It uses a set, **unique_elements**, to keep track of unique elements encountered so far. If an element is not already present in the set, it is appended to the unique_list and added to the new list. Finally, the modified list without duplicate elements is printed.

H. Program Logic-Flow chart:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

```
Original List:
fruits = ['apple', 'banana', 'orange', 'apple', 'banana']
Modified List:
['apple', 'banana', 'orange']
```

M. Practical related Quiz.

Find the output of the following program:

```
d1 = [10, 20, 30, 40, 50]
```

```
d2 = [1, 2, 3, 4, 5]
subtracted = list()
for d1, d2 in zip(d1, d2):
   item = d1 -d2
   subtracted.append(item)
print(subtracted)
```

A. *[10, 20, 30, 40, 50]*
B. *[1, 2, 3, 4, 5]*
C. *[10, 20, 30, 40, 50,-1, -2, -3, -4, -5]*
D. *9, 18, 27, 36, 45*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified

	modules, Selection of design framework					
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.2: Build a program to count the frequency of words appearing in a string using a dictionary.

A. Objective:

Apply dictionaries functions and operations to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

1. *This practical is expected to develop the following skills for the industry-identified competency:*
2. *Problem solving skill using dictionary for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO1: Understand linear and non-linear data structures.

E. Practical Outcome(PRO)

Write a program to calculate frequency of a given word in input string. To achieve this PRO use various concepts of dictionary.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Python Dictionary functions:

The data type *dictionary* fall under mapping. It is a mapping between a *set of keys* and a *set of values*. The key-value pair is called an *item*. A key is separated from its value by a colon(:) and consecutive items are separated by commas. Items in

dictionaries are unordered, so we may not get back the data in the same order in which we had entered the data initially in the dictionary.

Example

```
#dict1 is an empty Dictionary created
#curly braces are used for dictionary
>>> dict1 = {}
>>> dict1
{}
#dict2 is an empty dictionary created using
#built-in function

>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> dict3
{'Mohan': 95, 'Ram': 89, 'Suhel': 92,
'Sangeeta': 85}
```

- **Accessing Items in a Dictionary**
  The following example shows how a dictionary returns the value corresponding to the given key:

```
>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> dict3['Ram']
89
>>> dict3['Sangeeta']
85
#the key does not exist
>>> dict3['Shyam']
KeyError: 'Shyam'
```

- **Adding a new item**
  We can add a new item to the dictionary as shown in the following example:

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}

>>> dict1['Meena'] = 78
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92,
```

'Sangeeta': 85, 'Meena': 78}

- **Modifying an Existing Item**

  The existing dictionary can be modified by just overwriting the key-value pair. Example to modify a given item in the dictionary:

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
#Marks of Suhel changed to 93.5
>>> dict1['Suhel'] = 93.5
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 93.5,
'Sangeeta': 85}
```

- ## Dictionary Operations:

- **Membership**

  The membership operator in checks if the key is present in the dictionary and returns True, else it returns False.

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> 'Suhel' in dict1
True
```

  The not in operator returns True if the key is not present in the dictionary, else it returns False.

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> 'Suhel' not in dict1
False
```

- ## Traversing a Dictionary

  We can access each item of the dictionary or traverse a dictionary using for loop.

```
>>> dict1 ={'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> for key in dict1:
print(key,':',dict1[key])
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

- ## Dictionary methods and Built-in functions:

Method	Description	Example

len()	Returns the length or number of key: value pairs of the dictionary passed as the argument	>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> len(dict1)
		4

dict()	Creates a dictionary from a sequence of key-value pairs	pair1 = [('Mohan',95),('Ram',89), ('Suhel',92),('Sangeeta',85)]
		>>> pair1
		[('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)]
		>>> dict1 = dict(pair1)
		>>> dict1
		{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}

keys()	Returns a list of keys in the dictionary	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict1.keys()
		dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])

values()	Returns a list of values in the dictionary	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict1.values()
		dict_values([95, 89, 92, 85])

items()	Returns a list of tuples(key – value) pair	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} >>> dict1.items()

		dict_items([('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)])

get()	Returns the value corresponding to the key passed as the argument If the key is not present in the dictionary it will return None	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict1.get('Sangeeta')
		85
		>>> dict1.get('Sohan')
		>>>

update()	appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict2 = {'Sohan':79,'Geeta':89}
		>>> dict1.update(dict2)
		>>> dict1
		{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan': 79, 'Geeta': 89}
		>>> dict2
		{'Sohan': 79, 'Geeta': 89}

del()	Deletes the item with the given key
	To delete the dictionary from the memory we write:
	del Dict_name
		>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> del dict1['Ram']
		>>> dict1
		{'Mohan':95,'Suhel':92, 'Sangeeta': 85}
		>>> del dict1 ['Mohan']
		>>> dict1
		{'Suhel': 92, 'Sangeeta': 85}
		>>> del dict1

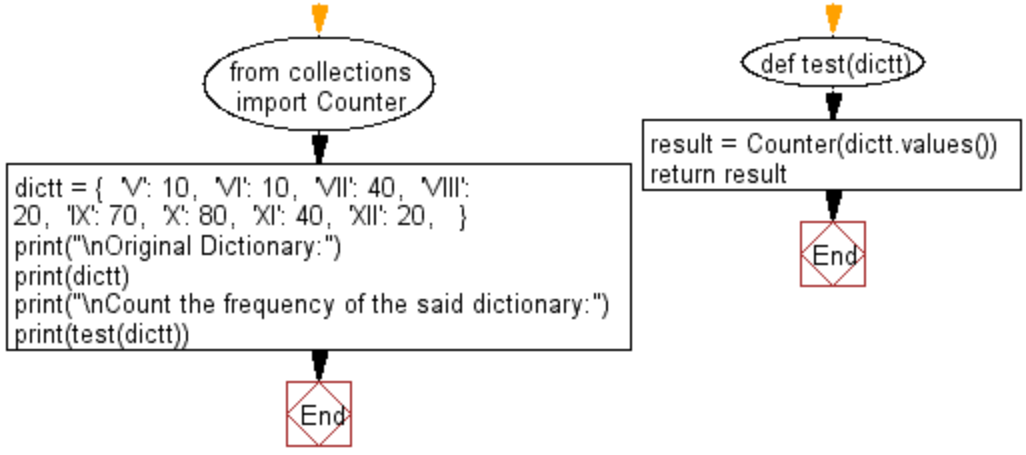
		>>> dict1 NameError: name 'dict1' is not defined

clear()	Deletes or clear all the items of the dictionary	>>> dict1 = {'Mohan':95,'Ram':89,
		'Suhel':92, 'Sangeeta':85}
		>>> dict1.clear()
		>>> dict1
		{ }

In this program, the count_word_frequency() function takes a string as input and returns a dictionary with the frequencies of each word in the string. It splits the string into a list of words using the split() method. Then, it iterates over each word and checks if it is already in the dictionary. If the word is present, it increments its count by 1. If the word is not present, it adds it to the dictionary with a count of 1. Finally, it returns the word frequency dictionary.

You can test the program with different strings by changing the example_string variable.

H. Program Logic-Flow chart:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

Case 1:
Enter string: hello world program world test
{'test': 1, 'world': 2, 'program': 1, 'hello': 1}

Case 2:
Enter string: orange banana apple apple orange pineapple
{'orange': 2, 'pineapple': 1, 'banana': 1, 'apple': 2}

M. Practical related Quiz.

Find the output of the following program:

```
d1 = [10, 20, 30, 40, 50]
d2 = [1, 2, 3, 4, 5]
subtracted = list()
for d1, d2 in zip(d1, d2):
    item = d1 -d2
    subtracted.append(item)
print(subtracted)
```

A. [10, 20, 30, 40, 50]
B. [1, 2, 3, 4, 5]
C. [10, 20, 30, 40, 50,-1, -2, -3, -4, -5]
D. 9, 18, 27, 36, 45

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implement	Properly Followed & implemented partly	Properly followed & Not implement	Partially Followed and Partially implement	Partially followed and Not implement

		ed		ed	ed	ed
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.3: Implement a Program for two matrix multiplication using simple nested loop and numpy module.

A. Objective:

Use nested loop with numpy module to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using list functions for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO1: Understand linear and non-linear data structures.

E. Practical Outcome(PRO)

Write a program to read two matrices. Multiply them with the use of simple nested loop in numpy module.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Matrix multiplication in NumPy is a fundamental operation that allows for efficient computation of linear algebraic operations. However, if we were to perform matrix multiplication using loops instead of relying on NumPy's built-in

functions, it would result in a less efficient and more cumbersome implementation.

How matrix multiplication can be implemented using loops in NumPy:

i. Suppose we have two matrices A and B, where A is of shape (m, n) and B is of shape (n, p). The resulting matrix C will have shape (m, p).

ii. We initialize an empty matrix C of shape (m, p) to store the result of the matrix multiplication.

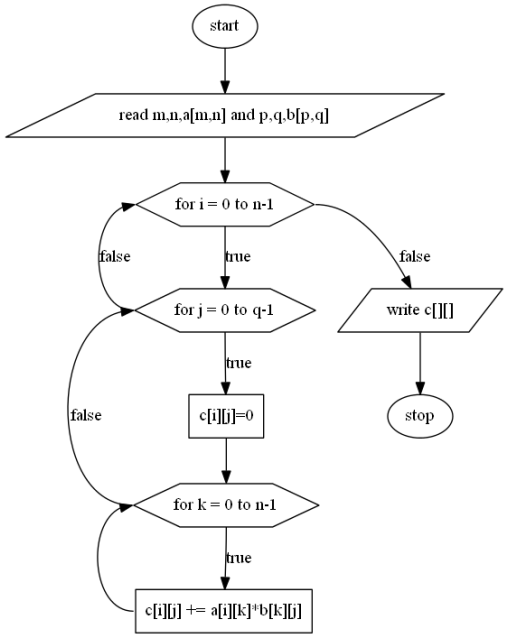
iii. We can iterate over the rows of A using a for loop and the columns of B using another for loop.

iv. Within the nested loops, we iterate over the range of n, which represents the common dimension of the matrices.

v. For each iteration, we multiply the corresponding elements of the current row of A and the current column of B and accumulate the sum.

vi. Finally, we assign the accumulated sum to the corresponding element in the result matrix C.

H. Program Logic-Flow chart:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

Input:
A = [[1, 2, 3], [4, 5, 6]]
B = [[7, 8], [9, 10], [11, 12]])
Output:
C= [[58 64], [139 154]]

M. Practical related Quiz.

What does the following code print?

```
output = ""
x = -5
while x < 0:
   x = x + 1
   output = output + str(x) + " "
print(output)
```

A. 5 4 3 2 1

B. -4 -3 -2 -1 0

C. -5 -4 -3 -2 -1

D. This is an infinite loop, so nothing will be printed

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.4: Implement basic operations on arrays.

A. Objective:

Demonstrate various operations of array in single program i.e. create, search, modify, slicing, concatenation and length of array.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using list functions for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO1: Understand linear and non-linear data structures.

E. Practical Outcome(PRO)

Write a program to implement create, access, modify, concatenation, slicing and check length operations on array.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

In Python, arrays can be manipulated using a variety of built-in operations and methods. Here are some basic array operations you can perform in Python:

Creating an array:

You can create an array using the array module or the numpy library.

Example:

```
# Using the array module
import array

my_array = array.array('i', [1, 2, 3, 4, 5])

# Using the numpy library
import numpy as np

my_array = np.array([1, 2, 3, 4, 5])
```

Accessing elements:
You can access individual elements in an array using indexing.

Example:

```
my_array = np.array([1, 2, 3, 4, 5])

# Accessing elements
print(my_array[0])   # Output: 1
print(my_array[2])   # Output: 3
```

Modifying elements:
You can modify elements in an array by assigning new values to specific indices.

Example:

```
my_array = np.array([1, 2, 3, 4, 5])

# Modifying elements
my_array[0] = 10
print(my_array)      # Output: [10, 2, 3, 4, 5]
```

Array slicing:
Slicing allows you to extract a subset of elements from an array.

Example:

```
my_array = np.array([1, 2, 3, 4, 5])

# Array slicing
print(my_array[1:4])  # Output: [2, 3, 4]
```

```
print(my_array[2:])    # Output: [3, 4, 5]
print(my_array[:3])    # Output: [1, 2, 3]
```

Array concatenation:

You can concatenate two or more arrays together using the concatenate function from the numpy library.

Example:

```
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])

# Array concatenation
result = np.concatenate((array1, array2))
print(result)        # Output: [1, 2, 3, 4, 5, 6]
```

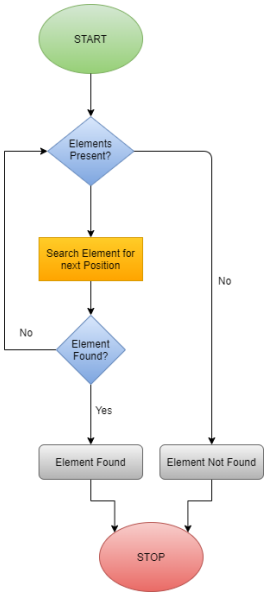
Array length:

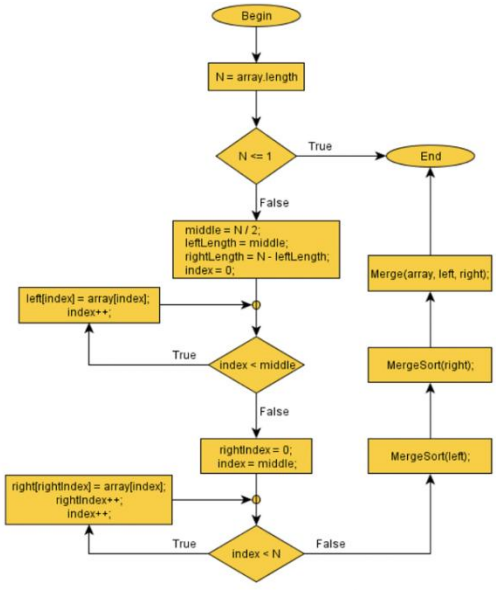
You can find the length of an array using the len() function.

Example:

```
my_array = np.array([1, 2, 3, 4, 5])

# Array length
length = len(my_array)
print(length)        # Output: 5
```

H. Program Logic-Flow chart: (Search & Check length operation)





I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

```
# Using the array module
import array

my_array = array.array('i', [1, 2, 3, 4, 5])

# Using the numpy library
import numpy as np

my_array = np.array([1, 2, 3, 4, 5])
```

```
my_array = np.array([1, 2, 3, 4, 5])

# Accessing elements
print(my_array[0])   # Output: 1
print(my_array[2])   # Output: 3
```

```
my_array = np.array([1, 2, 3, 4, 5])

# Modifying elements
my_array[0] = 10
print(my_array)      # Output: [10, 2, 3, 4, 5]
```

```
my_array = np.array([1, 2, 3, 4, 5])

# Array slicing
print(my_array[1:4])  # Output: [2, 3, 4]
print(my_array[2:])   # Output: [3, 4, 5]
print(my_array[:3])   # Output: [1, 2, 3]
```

```
my_array = np.array([1, 2, 3, 4, 5])

# Array length
length = len(my_array)
print(length)         # Output: 5
```

M. Practical related Quiz.

1. Which of the following is used to find the maximum element in a NumPy array?
a) max()
b) maximum()
c) amax()
d) All of the above

2. Which of the following is used to find the indices of the maximum and minimum elements in a NumPy array?
a) argmax() and argmin()
b) max() and min()
c) amax() and amin()
d) None of the above

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.5: Design an employee class for reading and displaying the employee information, the _init_() and print_employee_details() methods will be used respectively. Where _init_() will be a private method.

A. Objective:

Apply concept of class to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using list functions for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO2: Implement Object Oriented Programming concepts in Python.

E. Practical Outcome(PRO)

Write a program to design a class employee. This class should contain 2 methods: _init_() - to read data of employee and print_employee_details() - to display data of employee. _init_() method must be private.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

In Python, a class is a blueprint or template for creating objects (instances) that share similar properties and behaviours. It provides a way to define the structure

and behaviour of objects based on their characteristics and actions. The class serves as a blueprint that encapsulates data (attributes) and functions (methods) that operate on that data.

To define a class in Python, you use the class keyword followed by the class name. Here's a basic example of a class definition:

```
class MyClass:
    def __init__(self, name):
        self.name = name

    def greet(self):
        print(f"Hello, {self.name}!")

# Creating an instance of the class
obj = MyClass("Alice")

# Accessing attributes and calling methods
print(obj.name)     # Output: "Alice"
obj.greet()         # Output: "Hello, Alice!"
```

In this example, we define a class called MyClass. It has two components: the constructor method __init__ and the greet method. The __init__ method is called when creating an object of the class and is used to initialize the object's attributes. The self-parameter refers to the instance of the class itself.

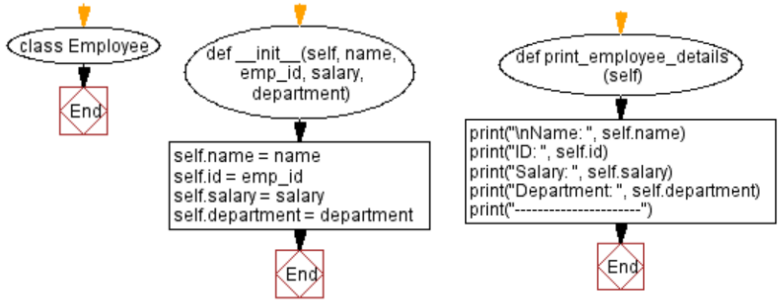
The greet method takes the self-parameter (which refers to the instance) and prints a greeting message using the name attribute.

To create an object (instance) of the class, you simply call the class name followed by parentheses, optionally passing any required arguments. In the example, obj is an instance of the MyClass class.

You can access the attributes and methods of an object using the dot notation. For example, obj.name accesses the name attribute, and obj.greet() calls the greet method.

Classes in Python allow you to create multiple objects with the same structure and behavior, making code more modular, reusable, and organized. They enable you to create complex data structures, encapsulate related data and functions together, and implement object-oriented programming principles like inheritance and polymorphism.

H. Program Logic-Flow chart:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

Employee Details:
Name: ADAMS
ID: E7876
Salary: 50000
Department: ACCOUNTING

M. Practical related Quiz.

Design an Student class for reading and displaying the student information, the getinfo() and printinfo() methods will be used respectively. Where getinfo() will be a private method.

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.6: Design a class Complex for adding the two complex numbers and also show the use of constructor.

A. Objective:

Apply various types of constructor to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using constructor for data operations in advanced python.*

D. Expected Course Outcomes(COs)

CO2: Implement Object Oriented Programming concepts in Python.

E. Practical Outcome(PRO)

Write a program to design a class Complex for adding the two complex numbers and also show the use of constructor.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Python constructor:

A constructor is a special type of method (function) which is used to initialize the instance members of the class.

In C++ or Java, the constructor has the same name as its class, but it treats constructor differently in Python. It is used to create an object.

Constructors can be of two types.

1. Parameterized Constructor
2. Non-parameterized Constructor

Constructor definition is executed when we create the object of this class. Constructors also verify that there are enough resources for the object to perform any start-up task.

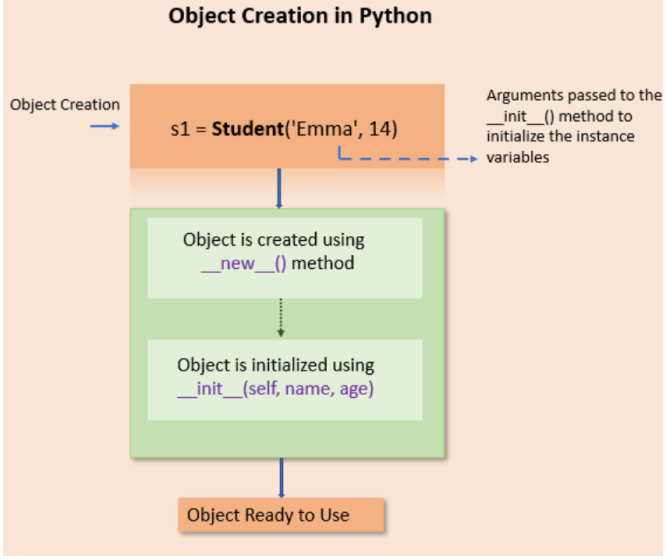
In Python, the method the **__init__()** simulates the constructor of the class. This method is called when the class is instantiated. It accepts the **self**-keyword as a first argument which allows accessing the attributes or method of the class.

We can pass any number of arguments at the time of creating the class object, depending upon the **__init__()** definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

Example:

```
class Employee:
    def __init__(self, name, id):
        self.id = id
        self.name = name
```

H. Program Logic-Flow chart:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

J. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

K. Source code:

L. Input-Output:

```
Enter first complex number
Enter the Real Part: 7
Enter the Imaginary Part: 5
First Complex Number:  7+5i
Enter second complex number
Enter the Real Part: 8
Enter the Imaginary Part: 7
Second Complex Number:  8+7i
Sum of two complex numbers is 15+12i
```

M. Practical related Quiz.

1. Which of the following is the default constructor in python?

A. __init__()
B. new()
C. constructor()
D. init()

2. How many times a constructor runs in each instantiation?

A. Depends on the number of method calls.
B. Depends on system
C. Only once.
D. None of these.

3. Find the output of the code given below?

```
class org:
        def __init__(self,name='QuizOrbit',loc='Pune'):
            self.name = name
            self.loc = loc
        def getDetails(self):
            return self
    orgDetail = org()
    print(getattr(orgDetail,'name'))
```

A. Syntax Error
B. QuizOrbit
C. Garbage Value
D. name

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.7: Write a program to Design a class for single level inheritance using public and private type derivation.

A. Objective:

Implement Object Oriented Programming concepts in Python.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
3. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
4. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
5. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using inheritance of public and private data in advanced python.*

D. Expected Course Outcomes(COs)

CO2: Implement Object Oriented Programming concepts in Python.

E. Practical Outcome(PRO)

Write a program to Design a class for single level inheritance using public and private type derivation.

Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

F. Prerequisite Theory:

One of the core concepts in object-oriented programming (OOP) languages is inheritance. It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class.

Inheritance is the capability of one class to derive or inherit the properties from another class.

Python Inheritance Syntax

The syntax of simple inheritance in Python is as follows:

```
Class BaseClass:
    {Body}
Class DerivedClass(BaseClass):
    {Body}
```

Creating a Parent Class

A parent class is a class whose properties are inherited by the child class. Let's create a parent class called **Person** which has a **Display** method to display the person's information.

```
# A Python program to demonstrate inheritance
class Person(object):

  # Constructor
  def __init__(self, name, id):
    self.name = name
    self.id = id

  # To check if this person is an employee
  def Display(self):
    print(self.name, self.id)


# Driver code
emp = Person("Satyam", 102) # An Object of Person
emp.Display()
```

Creating a Child Class

A child class is a class that drives the properties from its parent class. Here **Emp** is another class that is going to inherit the properties of the **Person** class(base class).

What is an object class in Python?

Like the Java Object class, in Python (from version 3. x), the object is the root of all classes.

- In Python 3.x, "class Test(object)" and "class Test" are same.

- In Python 2. x, "class Test(object)" creates a class with the object as a parent (called a new-style class), and "class Test" creates an old-style class (without an objecting parent).

Subclassing (Calling constructor of parent class)

A child class needs to identify which class is its parent class. This can be done by mentioning the parent class name in the definition of the child class.
Example: class subclass_name (superclass_name)

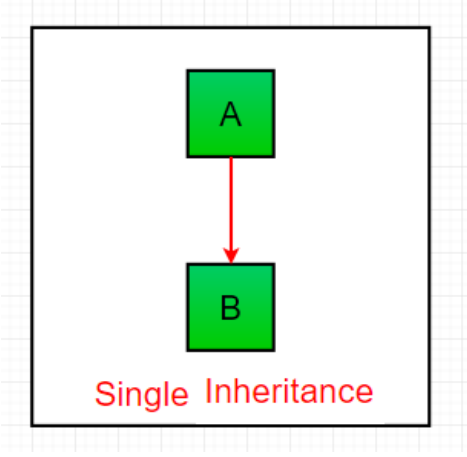
The super() Function

The super() function is a built-in function that returns the objects that represent the parent class. It allows to access the parent class's methods and attributes in the child class.

Different types of Python Inheritance

There are 5 different types of inheritance in Python. They are as follows:
Single inheritance: When a child class inherits from only one parent class, it is called single inheritance. We saw an example above.
Multiple inheritances: When a child class inherits from multiple parent classes, it is called multiple inheritances.
Multilevel inheritance: When we have a child and grandchild relationship. This means that a child class will inherit from its parent class, which in turn is inheriting from its parent class.
Hierarchical inheritance: More than one derived class can be created from a single base.
Hybrid inheritance: This form combines more than one form of inheritance. Basically, it is a blend of more than one type of inheritance.

G. Program Logic-Flow chart:



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDEs and Code Editors Open Source : IDLE, Jupyter*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code:

K. Input-Output:

Input- 'Person' is the parent class, and 'Employee' is its child class.
(person1 is an object of class Person & person2 is an object of class Employee)

Output- `person1 False`

`person2 True`

L. Practical related Quiz.

A. What is inheritance?
B. List types of inheritance.
C. Write a python program to demonstrate single level inheritance.

M. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 8: Implement multiple and hierarchical inheritance.

A. Objective:

Demonstrate the concept of Multiple and Hierarchical inheritance using python.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry identified competency:

1. *Develop program on multiple inheritance and Hierarchical inheritance using python.*

D. Expected Course Outcomes(COs)

CO2: Implement Object Oriented Programming concepts in Python.

E. Practical Outcome(PRO)

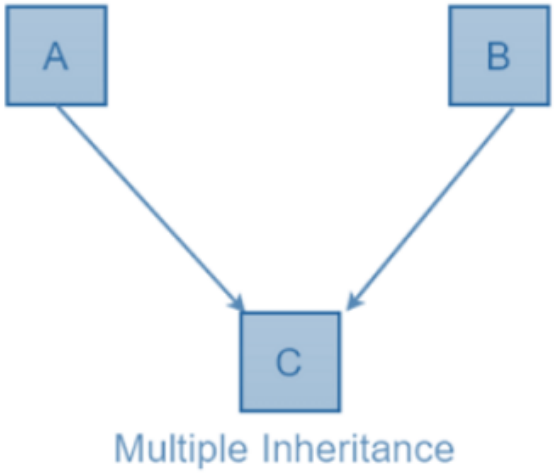
Write a program in Python to demonstrate following operations: Multiple inheritance, Hierarchical inheritance.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Multiple Inheritance: In multiple inheritance, a class can inherit from multiple base classes. This allows the derived class to inherit attributes and methods from multiple sources.



Syntax:

```
class DerivedClass(BaseClass1, BaseClass2, ...):
    # Class body
    Pass
```

Example: Multiple Inheritance

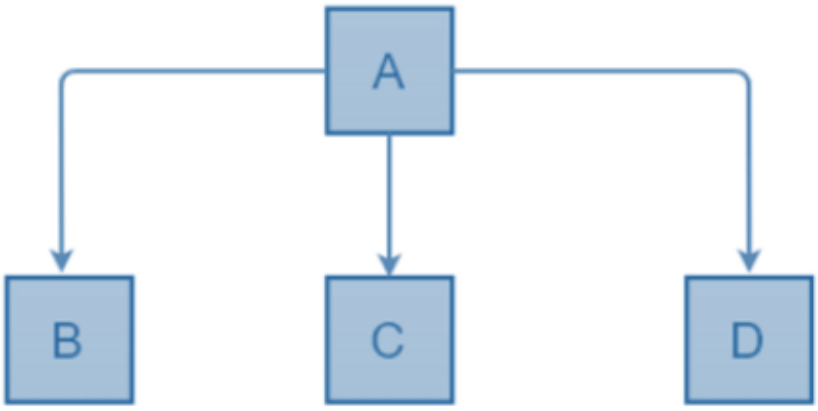
```
# Base class 1
class BaseClass1:
    def method1(self):
        print("BaseClass1 method1")

# Base class 2
class BaseClass2:
    def method2(self):
        print("BaseClass2 method2")

# Derived class inheriting from both BaseClass1 and BaseClass2
class DerivedClass(BaseClass1, BaseClass2):
    def method3(self):
        print("DerivedClass method3")
```

```
# Create an instance of DerivedClass and demonstrate multiple inheritance
instance = DerivedClass()
instance.method1()  # Inherited from BaseClass1
instance.method2()  # Inherited from BaseClass2
instance.method3()  # Defined in DerivedClass
```

Output:

```
BaseClass1 method1
BaseClass2 method2
DerivedClass method3
```

Hierarchical Inheritance: In hierarchical inheritance, a class serves as a base class for multiple derived classes. This means that multiple classes inherit from a single class, forming a hierarchy.



Hierarchical Inheritance

Syntax:

```
class BaseClass:
    # Base class members
class DerivedClass1(BaseClass):
    # Derived class 1 members
class DerivedClass2(BaseClass):
    # Derived class 2 members
```

Example:

```
# Base class
class Animal:
    def __init__(self, name):
        self.name = name
    def eat(self):
        print(f"{self.name} is eating.")

# First derived class inheriting from Animal
class Dog(Animal):
    def bark(self):
        print(f"{self.name} is barking.")

# Second derived class inheriting from Animal
class Cat(Animal):
    def meow(self):
        print(f"{self.name} is meowing.")

# Create instances of Dog and Cat and demonstrate hierarchical inheritance
dog = Dog("Buddy")
dog.eat()     # Inherited from Animal
dog.bark()    # Defined in Dog

cat = Cat("Whiskers")
cat.eat()     # Inherited from Animal
cat.meow()    # Defined in Cat
```

Output:

```
Buddy is eating.
Buddy is barking.
Whiskers is eating.
```

Whiskers is meowing.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*PYTHON IDE(any)*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables.*

J. Source code:

K. Input-Output:

L. Practical related Quiz.

1. Which type of inheritance allows a class to inherit from multiple base classes?

a) Single inheritance b) Multiple inheritance c) Hierarchical inheritance

2. In multiple inheritance, the order of base classes in the class definition determines the _____.

a) Method resolution order b) Initialization order c) Attribute lookup order

3. In hierarchical inheritance, multiple derived classes inherit from _____ base class(es).

a) One b) Two c) Multiple

4. In hierarchical inheritance, can a derived class have its own unique attributes and methods in addition to inheriting from the base class?

a) Yes b) No

M. References / Suggestions (lab manual designer should give)

1. *Introduction to Problem Solving with Python E. Balagurusamy Mc Graw Hill India, New Delhi.*
2. https://www.w3resource.com/python-exercises/oop/index.php
3. *https://www.geeksforgeeks.org/inheritance-in-Python/*

4. https://www.javatpoint.com/types-of-inheritance-python
5. https://youtu.be/-hmx1GDqbQE

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 9: Write a Python program to demonstrate method overriding using inheritance.

A. Objective:

Demonstrate the concept of method overriding using inheritance in python.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program of method overriding using inheritance in python.*

D. Expected Course Outcomes(COs)

CO2: Implement Object Oriented Programming concepts in Python.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate following operations: Method Overriding using Inheritance.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Method overriding: Method overriding is a concept in object-oriented programming where a subclass provides a different implementation of a method

that is already defined in its superclass. In other words, the subclass overrides the method inherited from the superclass with its own implementation.

When a method is called on an object of the subclass, the overridden method in the subclass is executed instead of the method in the superclass. This allows the subclass to modify or extend the behavior of the inherited method according to its specific requirements.

Example: Method Overriding using Inheritance

```
class Animal:
    def sound(self):
        print("Animal makes a sound")
class Cat(Animal):
    def sound(self):
        print("Meow")
class Dog(Animal):
    def sound(self):
        print("Woof")
class Cow(Animal):
    pass

# Create instances of each class
animal = Animal()
cat = Cat()
dog = Dog()
cow = Cow()

# Call the sound method of each instance
animal.sound()
cat.sound()
```

```
dog.sound()
cow.sound()
```

Output:

```
Animal makes a sound
Meow
Woof
Animal makes a sound
```

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

I. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables.

J. Source code:

K. Input-Output:

L. Practical related Quiz.

1. What is method overriding in Python?
a) Creating a new method with a different name in a subclass
b) Modifying the behavior of a method inherited from a superclass
c) Creating a new method in a subclass that has the same name but different parameters

2. When a method is called on an object of the subclass, which implementation is executed?
a) Implementation of the method in the subclass
b) Implementation of the method in the superclass
c) Both implementations are executed simultaneously

3. Can a subclass override multiple methods from the superclass?
a) Yes b) No c) It depends on the programming language

M. References / Suggestions (lab manual designer should give)

1. *Introduction to Problem Solving with Python E. Balagurusamy Mc Graw Hill India, New Delhi.*
2. https://www.w3resource.com/python-exercises/oop/index.php
3. https://www.geeksforgeeks.org/method-overriding-in-python/
4. https://www.javatpoint.com/inheritance-in-python
5. https://youtu.be/8xtVEDgoLWE

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified

Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 10: Implement push and pop algorithms of stack using list.

A. Objective:

Implement the stack operation – push and pop using list in python.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program of stack operations using python list.*

D. Expected Course Outcomes(COs)

CO3: Implement basic data structures such as stacks, queues and linked lists.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate following Stack operations: Push and Pop using Python List.

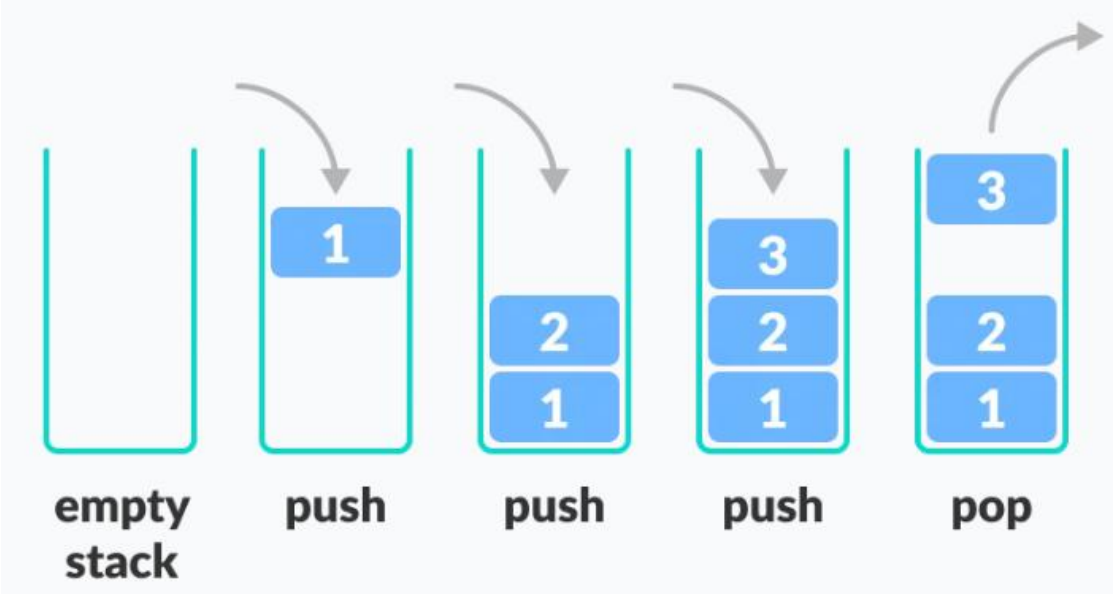
F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

The stack data structure is a fundamental data structure in computer science that follows the Last-In-First-Out (LIFO) principle. It is an ordered collection of elements in which the addition and removal of items can only be done from the same end, traditionally referred to as the "top" of the stack.

Stacks can be implemented using various data structures, such as arrays, linked lists, or dynamic arrays.



Here, I'll provide an example implementation of a stack using a Python list:

In this implementation, we define the Stack class with several methods:

- push(item): Adds an item to the top of the stack.
- pop(): Removes and returns the item from the top of the stack.
- is_empty(): Checks if the stack is empty and returns a boolean value.
- traverse(): Print the elements of a stack.

Overflow Condition: Trying to push an element in a full stack is called overflow.

Underflow Condition: Trying to pop out an empty stack is called underflow

Applications of Stack:

There are many applications of a stack. Some of them are:

• Stacks are used in backtracking algorithms.

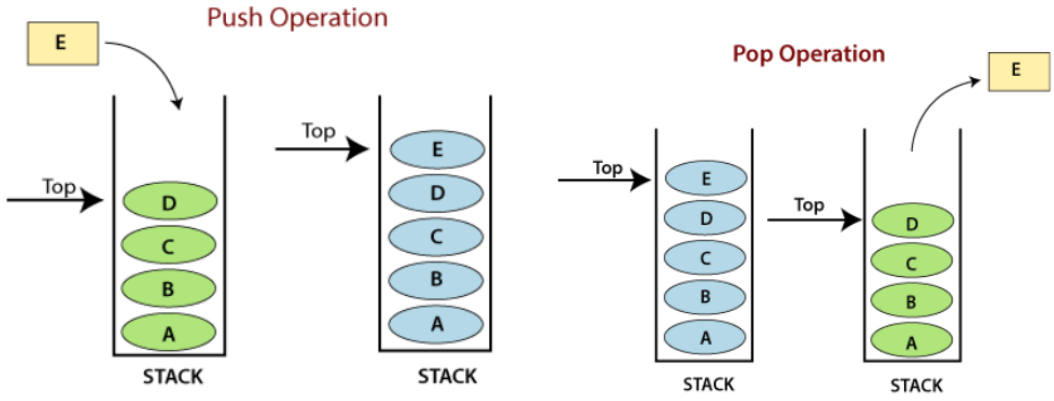
• They are also used to implement undo/redo functionality in a software.

• Stacks are also used in syntax parsing for many compilers.

• Stacks are also used to check proper opening and closing of parenthesis.

H. Program Logic-Flow chart

Python built-in list operation append() and pop() used to push and pop elements in the stack respectively.



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables.

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. Which operation adds an element to the top of the stack?
a) push() b) pop() c) peek() d) remove()

2. Which operation removes an element from the top of the stack?
a) push() b) pop() c) peek() d) remove()

3. What is the principle that a stack follows?
a) First-In-First-Out (FIFO) b) Last-In-First-Out (LIFO) c) First-Come-First-Served (FCFS) d) Last-Come-First-Served (LCFS)

4. Which of the following data structures can be used to implement a stack?
a) Array b) Linked List c) Both Array and Linked List d) Neither Array nor Linked List

5. Consider these operations on an empty stack: push(3), push(5), pop(), push(10), push(11), pop(), push(100). What will be the stack configuration (first number is top of the stack, last is the bottom)
a) 100,10,3 b) 3,5,10,11,100 c) 3,10,100 d) None of the above

N. References / Suggestions (lab manual designer should give)

1. *Data structures and Algorithms in Python M.Goodrich Wiley, 2013*
2. *Data Structures and Algorithmic Thinking with Python N.Karumanchi, Career Monk Publications, 2016*
3. *https://www.javatpoint.com/stack-in-python*
4. *https://www.sanfoundry.com/python-problems-solutions/#python-programs-stack*
5. *https://ds1-iiith.vlabs.ac.in/exp/stacks-queues/stacks/index.html*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 11: Implement a program to convert infix notation to postfix notation using stack.

A. Objective:

Demonstrate the application of stack - infix notation to postfix notation.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program of conversion of infix to postfix notation using stack.*

D. Expected Course Outcomes(COs)

CO3: Implement basic data structures such as stacks, queues and linked lists.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate conversion of infix to postfix notation using stack.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Infix expression: An expression is said to be in infix notation if the operators in the expression are placed in between the operands on which the operator works. For example, a + b * c

Postfix expression(Reverse Polish Notation): An expression is said to be in postfix notation if the operators in the expression are placed after the operands on which the operator works. For example, abc*+

Examples:

Input: A + B * C + D

Output: ABC*+D+

When converting an infix expression to a postfix expression, it is important to consider the precedence of operators to ensure that the resulting postfix expression is equivalent to the original infix expression. Here is a common precedence order for operators, from highest to lowest:

1. Exponentiation (^)
2. Multiplication (*) and division (/)
3. Addition (+) and subtraction (-)

In this precedence order, operators with higher precedence are evaluated before operators with lower precedence. Parentheses can be used to override the default precedence order.

In addition to considering operator precedence, the associativity of operators is also important when converting infix notation to postfix notation. Associativity determines how operators of the same precedence are grouped in the absence of parentheses. There are two types of associativity:

Left-associative: Operators with left-associativity are evaluated from left to right. For example, in the expression "5 - 3 - 1", the subtraction operators are left-associative, so the evaluation proceeds as "(5 - 3) - 1".

Right-associative: Operators with right-associativity are evaluated from right to left. For example, in the expression "2 ^ 3 ^ 2", the exponentiation operator is right-associative, so the evaluation proceeds as "2 ^ (3 ^ 2)".

H. Program Logic-Flow chart

To convert an infix notation to postfix notation using a stack, you can follow these steps:

Step 1: Initialize an empty stack and an empty string to store the postfix expression.

Step 2: Read the infix expression from left to right, character by character.

Step 3: If the current character is an operand (an alphanumeric character), append it directly to the postfix expression.

Step 4: If the current character is an opening parenthesis '(', push it onto the stack.

Step 5: If the current character is a closing parenthesis ')', pop operators from the stack and append them to the postfix expression until an opening parenthesis '(' is encountered. Remove the opening parenthesis from the stack but do not append it to the postfix expression.

Step 6: If the current character is an operator (+, -, *, /, etc.), compare its precedence with the operator on the top of the stack (if any). If the stack is empty or the top operator has lower precedence, push the current operator onto the stack. If the top operator has higher or equal precedence, pop operators from the stack and append them to the postfix expression until a lower precedence operator is encountered or the stack becomes empty. Then push the current operator onto the stack.

Step 7: Repeat steps 3-6 until all characters in the infix expression have been processed.

Step 8: After processing all characters, pop any remaining operators from the stack and append them to the postfix expression.

Step 9: The resulting postfix expression is the desired output.

Example: Infix Expression: A+ (B*C-(D/E^F)*G)*H, where ^ is an exponential operator.

Symbol	Scanned	STACK	Postfix Expression	Description
1.		(		Start
2.	A	(	A	
3.	+	(+	A	
4.	(	(+(	A	
5.	B	(+(	AB	
6.	*	(+(*	AB	
7.	C	(+(*	ABC	
8.	-	(+(-	ABC*	'*' is at higher precedence than '-'
9.	(	(+(-(	ABC*	
10.	D	(+(-(	ABC*D	
11.	/	(+(-(/	ABC*D	
12.	E	(+(-(/	ABC*DE	
13.	^	(+(-(/^	ABC*DE	
14.	F	(+(-(/^	ABC*DEF	
15.	)	(+(-	ABC*DEF^/	Pop from top on Stack , that's why '^' Come first
16.	*	(+(-*	ABC*DEF^/	
17.	G	(+(-*	ABC*DEF^/G	
18.	)	(+	ABC*DEF^/G*-	Pop from top on Stack , that's why '^' Come first
19.	*	(+*	ABC*DEF^/G*-	
20.	H	(+*	ABC*DEF^/G*-H	
21.	)	Empty	ABC*DEF^/G*-H*+	END

I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables.

K. Source code:

L. Input-Output:

Input − Infix expression.

Input: ((A + B) – C * (D / E)) + F

Output − Convert infix expression to postfix form.

Output: AB+CDE/*-F+

M. Practical related Quiz.

1. When converting infix to postfix, which data structure is commonly used?
a) Queue b) Stack c) Linked List d) Array)

2. What is the result of converting the infix expression "2 + 3 * 4" to postfix notation?
a) "2 3 4 + *" b) "2 3 + 4 *" c) "2 3 4 * +" d) "2 3 4 * +"

3. Convert (A + B) * (C – D) infix expression into POSTFIX expression.

N. References / Suggestions (lab manual designer should give)

1. *Data structures and Algorithms in Python M.Goodrich Wiley, 2013*
2. *Data Structures and Algorithmic Thinking with Python N.Karumanchi, Career Monk Publications, 2016*
3. *https://www.geeksforgeeks.org/convert-infix-expression-to-postfix-expression/*
4. *https://ds1-iiith.vlabs.ac.in/exp/infix-postfix/index.html*
5. *https://youtu.be/qtbKh9jJxok*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 12: Implement recursive functions.

A. Objective:

Demonstrate the application of stack - recursion.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program to show the concepts of recursive function using python.*

D. Expected Course Outcomes(COs)

CO3: Implement basic data structures such as stacks, queues and linked lists.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate the concepts of recursive function.

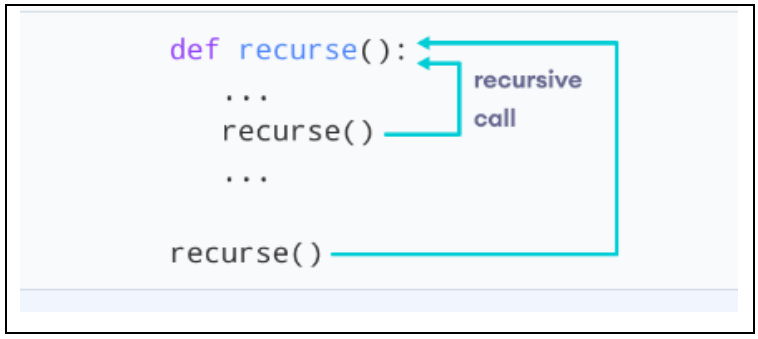
F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Recursion means a function call itself. A function is said to be recursive if it calls itself again and again within its body. Recursion uses stack.

Syntax:



Recursive functions typically have two parts:

Base Case: This is the termination condition that defines when the recursion should stop. It is the simplest case that can be solved directly without further recursion. When the base case is reached, the function stops calling itself and starts returning values back up the call stack.

Recursive Case: This is the part of the function where it calls itself to solve a smaller or simpler version of the problem. The function breaks down the original problem into smaller subproblems, making progress towards the base case with each recursive call.

Here's an example of a recursive function that calculates the factorial of a non-negative integer:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

In this function, the base case is when ‘n’ reaches 1, where the factorial is defined as 1. For any other value of ‘n’, the function calls itself with ‘n – 1’ to calculate the factorial of the smaller subproblem. The function keeps calling itself recursively until it reaches the base case and then starts returning the computed values back up the call stack.

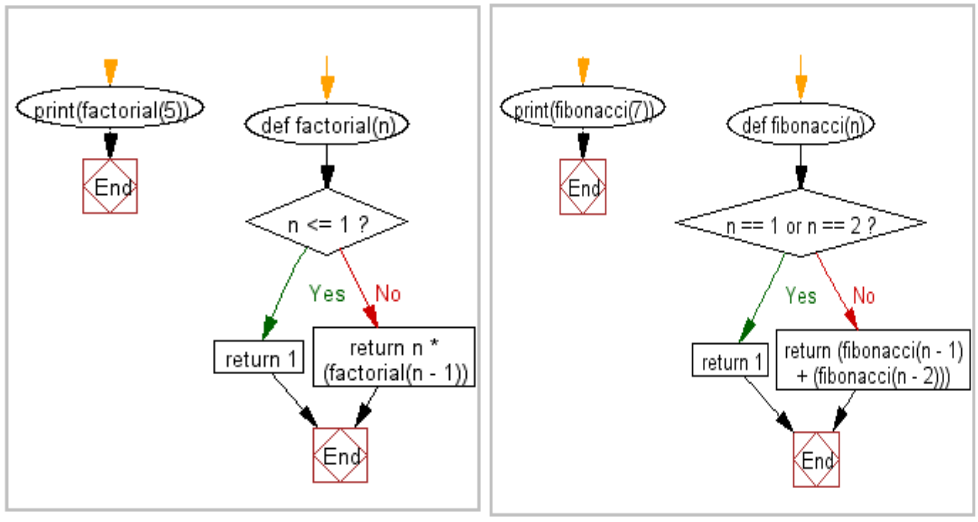
H. Program Logic-Flow chart

In the above example, factorial() is a recursive function as it calls itself.

When we call this function with a positive integer, it will recursively call itself by decreasing the number. Each function multiplies the number with the factorial of the number below it until it is equal to one. This recursive call can be explained in the following steps.

```
factorial(3)        # 1st call with 3
3 * factorial(2)    # 2nd call with 2
3 * 2 * factorial(1)  # 3rd call with 1
3 * 2 * 1           # return from 3rd call as number=1
3 * 2               # return from 2nd call
6                   # return from 1st call
```

Let's look at the process of factorial(left) and Fibonacci (right) using recursion:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.

2. *Do not plug out any computer cables.*

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. When using recursion, what is the call stack?

a) A stack data structure used in recursion.

b) A memory area where recursive function calls are stored.
c) A way to keep track of the number of recursive calls.
d) A mechanism to prevent infinite recursion.

2. What can happen if a recursive function does not have a base case?
a) The program executes with no issues.
b) The program gives an error message.
c) The function runs infinitely and may result in a stack overflow.
d) The function becomes faster and more efficient.

3. Which of the following is an example of a recursive data structure?
a) List b) Dictionary c) Stack d) Tuple.

N. References / Suggestions (lab manual designer should give)

1. *Data structures and Algorithms in Python M.Goodrich Wiley, 2013*
2. *Data Structures and Algorithmic Thinking with Python N.Karumanchi, Career Monk Publications, 2016*
3. *https://www.javatpoint.com/essential-recursion-programs-in-python*
4. *https://www.programiz.com/python-programming/recursion*
5. *https://www.w3resource.com/python-exercises/data-structures-and-algorithms/python-recursion.php*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

	concept.					
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justificatio	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

		n				

Sign with Date

Date: ……

Practical No. 13: Implement a program to implement QUEUE using list that performs following operations: ENQUEUE, DEQUEUE, DISPLAY.

A. Objective:

Demonstrate the concept of Queue using python list.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program to show the concepts of queue data structure using python.*

D. Expected Course Outcomes(COs)

CO3: Implement basic data structures such as stacks, queues and linked lists.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate the concepts of queue data structure.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

A queue is a data structure that follows the First-In-First-Out (FIFO) principle. It is similar to a real-life queue, such as people waiting in line, where the first person who joins the queue is the first one to be served.

In a queue, elements are inserted at the rear (enqueue) and removed from the front (dequeue). This order ensures that the oldest elements are processed first, preserving the FIFO behavior.

Key operations performed on a queue are:

Enqueue: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.

Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition

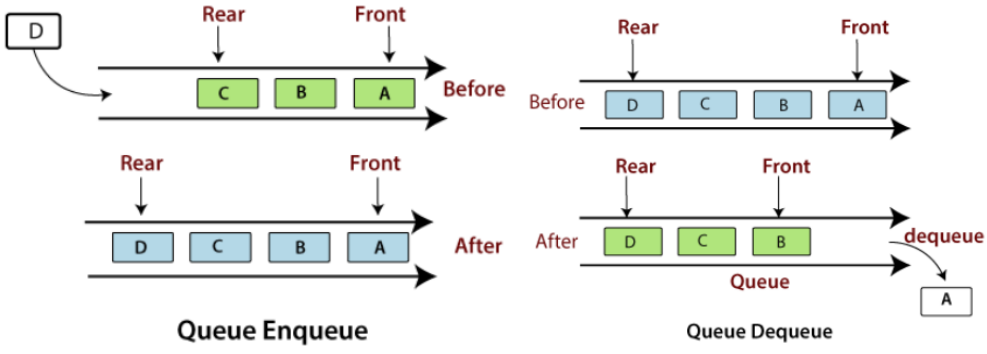
Traversing/Displaying - It involves to visit each element of the queue.

Applications of Queue:

- Queue is used to implement many algorithms like Breadth First Search (BFS), etc.
- It can be also used by an operating system when it has to schedule jobs with equal priority.
- Customers calling a call center are kept in queues when they wait for someone to pick up the calls

H. Program Logic-Flow chart

Python built-in list operation append() and pop() used to push and pop elements in the queue respectively.



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables.

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. _____ is a linear list of elements in which insertion and deletion takes place from different ends.

2. Insertion operation in a queue is called _____ and deletion operation in a queue is called _____.

3. Deletion of elements is performed from _____ end of the queue.

4. Elements 'A','S','D' and 'F' are present in the queue, and they are deleted one at a time, _____ is the sequence of element received.

5. What is the fundamental principle followed by a queue?
a) Last-In-First-Out (LIFO) b) First-In-Last-Out (FILO) c) First-In-First-Out (FIFO) d) Last-In-Last-Out (LILO)

6. Which data structure can be used to implement a queue?
a) List b) Set c) Dictionary d) Tuple

N. References / Suggestions (lab manual designer should give)

1. *Data structures and Algorithms in Python M.Goodrich Wiley, 2013*
2. *Data Structures and Algorithmic Thinking with Python N.Karumanchi, Career Monk Publications, 2016*
3. https://www.javatpoint.com/queue-in-python
4. *https://www.programiz.com/dsa/queue*
5. https://www.sanfoundry.com/python-problems-solutions/#python-programs-queue

6. https://ds1-iiith.vlabs.ac.in/exp/stacks-queues/queues/index.html

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No. 14: Implement program to perform following operation on singly linked list: a. Insert a node at the beginning of a singly linked list. b. Insert a node at the end of a singly linked list. c. Insert a node after the given node of a singly linked list. d. Insert a node before the given node of singly linked list. e. Delete a node from the beginning of a singly linked list. f. Delete a node from the end of a singly linked list. g. Count the number of nodes of a singly linked list. h. Display content of singly linked list.

A. Objective:

Demonstrate the different operations of singly linked list.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects*
6. ***Life-long learning solution:*** *Ability to analyse individual needs and engage in updating in the context of technological changes.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Develop program to show the operations of singly linked list using python.*

D. Expected Course Outcomes(COs)

CO3: Implement basic data structures such as stacks, queues and linked lists.

E. Practical Outcome(PRO)

Write a program in Python to demonstrate the operations on singly linked list.

F. Expected Affective domain Outcome(ADOs)

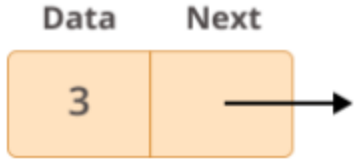
1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

A singly linked list is a type of data structure that represents a collection of elements. It consists of a sequence of nodes, where each node contains data and a reference (or pointer) to the next node in the sequence. The first node in the list is called the head, and the last node typically has a reference to None, indicating the end of the list.

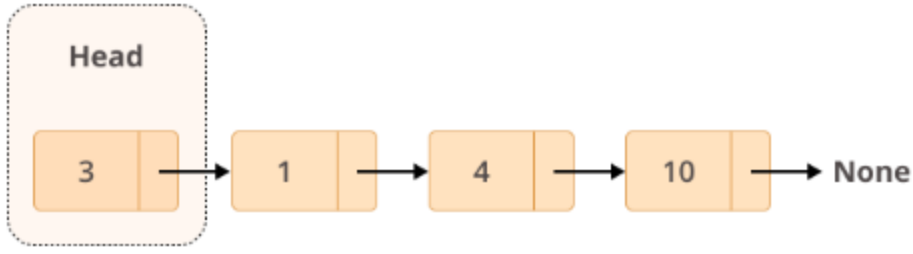
Here are the key components of a singly linked list:

1. Node: Each node in the list contains two parts: data and next. The data part holds the value or information associated with the node, while the next part is a reference (or pointer) to the next node in the list.



Node

2. Head: The head is a special node that represents the first element of the linked list. It serves as the starting point to access the rest of the list.
3. Next Pointer: Each node's next pointer points to the next node in the sequence. It establishes the logical link between the nodes, allowing traversal from one node to the next.



Linked List

Python code to represent the Node structure:

```
class Node:
  def __init__(self, data):
    self.data = data
    self.next = None
```

Singly linked lists support various operations to manipulate the list and perform common tasks. Here are some of the most common operations performed on a singly linked list:

Insertion at the Beginning:
Insert a new node at the beginning of the linked list by creating a new node and updating the next pointer of the new node to point to the current head of the list. Then, update the head of the list to point to the new node.

Insertion at the End:
Insert a new node at the end of the linked list by creating a new node and traversing to the last node. Set the next pointer of the last node to the new node, and update the new node's next pointer to None.

Insertion after a Given Node:
Insert a new node after a specified node in the linked list by creating a new node and updating the next pointer of the new node to point to the next node of the specified node. Then, update the next pointer of the specified node to point to the new node.

Deletion from the Beginning:
Remove the first node from the linked list by updating the head of the list to point to the second node. This effectively removes the first node from the list.

Deletion from the End:
Remove the last node from the linked list by traversing to the second-to-last node and setting its next pointer to None. This effectively removes the last node from the list.

Counting the Number of Nodes:
Traverse the linked list while incrementing a count variable for each node encountered. Return the final count, which represents the number of nodes in the list.

Displaying the Contents of the List:
Traverse the linked list and print or store the data value of each node in the desired format.

These operations provide the basic functionality needed to manipulate and work with a singly linked list. By combining these operations, you can perform various tasks and solve different problems using singly linked lists.

H. Program Logic-Flow chart

To insert a node at the beginning of a singly linked list, you can follow these steps:

1. Create a new node with the given data.
2. Set the next pointer of the new node to the current head of the linked list.
3. Update the head of the linked list to point to the new node.

To insert a node at the end of a singly linked list, you can follow these steps:

1. Create a new node with the given data.
2. If the linked list is empty, make the new node the head of the list.
3. Otherwise, traverse the linked list until you reach the last node.
4. Set the next pointer of the last node to the new node.

To insert a node after a given index in a singly linked list, you can follow these steps:

1. Create a new node with the given data.
2. Traverse the linked list to find the node at the given index.
3. If the given index is out of range (greater than the number of nodes), return an error or do nothing.
4. Set the next pointer of the new node to the next node of the node at the given index.
5. Set the next pointer of the node at the given index to the new node.

To insert a node before a given index in a singly linked list, you can follow these steps:

1. Create a new node with the given data.
2. If the given index is 0, insert the new node at the beginning of the list.
3. Traverse the linked list to find the node at the given index and its previous node.
4. If the given index is out of range (greater than the number of nodes), return an error or do nothing.
5. Set the next pointer of the new node to the node at the given index.
6. If the previous node is None, update the head of the linked list to point to the new node.
7. Otherwise, set the next pointer of the previous node to the new node.

To delete the first node from a singly linked list, you can follow these steps:

1. Check if the linked list is empty:
    i. If the head node is None, it means the linked list is empty, so there is nothing to delete.

   ii. In this case, you can return an error or take appropriate action based on your implementation requirements.
2. Update the head of the linked list:
   i. Set the head attribute to the next node of the current head, effectively removing the first node from the list.

To delete the last node from a singly linked list, you can follow these steps:

1. Check if the linked list is empty:
   a. If the head node is None, it means the linked list is empty, so there is nothing to delete.
   b. In this case, you can return an error or take appropriate action based on your implementation requirements.
2. Check if the linked list has only one node:
   a. If the next node of the head is None, it means there is only one node in the list.
   b. In this case, set the head attribute to None, effectively deleting the only node in the list.
3. Traverse the linked list to find the last node and the second-to-last node:
   a. Start at the head node and traverse the list until you reach the last node.
   b. Keep track of the current node and the second-to-last node during traversal.
4. Update the next pointer of the second-to-last node:
   a. Set the next pointer of the second-to-last node to None, effectively removing the last node from the list.

To count the number of nodes in a singly linked list, you can follow these steps:

1. Initialize a count variable to 0.
2. Traverse the linked list:
   a. Start at the head node of the linked list.
   b. Increment the count variable by 1 for each node visited.
   c. Move to the next node until reaching the end of the list.
3. Return the count.

To display the content of a singly linked list, you can follow these steps:

1. Start at the head node of the linked list.
2. Traverse the linked list:
   a. While the current node is not None:
      i. Print the data of the current node.
      ii. Move to the next node.
3. Repeat step 2 until reaching the end of the list.

I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables.

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. What is a singly linked list?
a) A list that allows elements to be accessed in any order.
b) A list where each node contains a reference to the previous node.
c) A list where each node contains a reference to the next node.

2. Insertion of an element at the middle of a linked list requires the modification of how many pointers?
a) 2 b) 1 c) 3 d) 4.

3. Which data structure represents a node in a singly linked list?
a) Dictionary b) Tuple c) Class

N. References / Suggestions (lab manual designer should give)

1. *Data structures and Algorithms in Python M.Goodrich Wiley, 2013*
2. *Data Structures and Algorithmic Thinking with Python N.Karumanchi, Career Monk Publications, 2016*
3. https://www.tutorialspoint.com/python_data_structure/python_linked_lists.htm
4. *https://www.w3resource.com/python-exercises/data-structures-and-algorithms/python-linked-list.php*
5. *https://www.sanfoundry.com/python-programming-examples-linked-lists/*

6. https://ds1-iiith.vlabs.ac.in/exp/linked-list/index.html

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.15: Implement a python program to search a particular element from list using Linear and Binary Search.

A. Objective:

Apply linear search and binary search to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using linear and binary search in data structure with python.*

D. Expected Course Outcomes(COs)

CO4: Apply Algorithms for solving problems like searching and sorting of data.

E. Practical Outcome(PRO)

Write a program to demonstrate the linear search and binary search.

F. Expected Affective domain Outcome (ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Searching algorithm:

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. They search for a target (key) in the search space.

Search algorithms can be classified based on their mechanism of searching into three types of algorithms: linear, binary, and hashing.

Linear or Sequential Search:

Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set. The time complexity of the Linear search is O(n).

The basic steps to perform linear Search:

- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, return the index.
- If x doesn't match with any of the elements, return -1.

Binary Search or Interval Search:

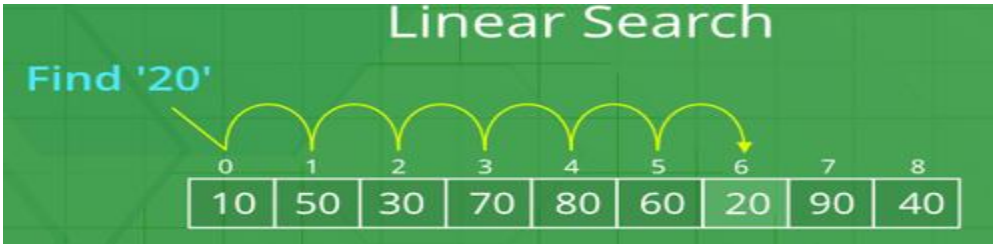
Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

The basic steps to perform binary Search:

- Begin with the mid element of the whole array as a search key. Middle=(low+high)/2
- If the value of the search key is equal to the item, then return an index of the search key.
- Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. High=Middle-1
- Otherwise, narrow it to the upper half. Low=Middle+1
- Repeatedly check from the second point until the value is found or the interval is empty.

H. Program Logic-Flow chart:

Example: To search a particular element from list using Linear and Binary Search.



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *What is a search algorithm and what types are there?*

2. *What is the difference between binary search and linear search?*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified

Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.16: Implement Bubble sort algorithm.

A. Objective:

Apply bubble sort algorithm to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using bubble sort algorithm.*

D. Expected Course Outcomes(COs)

CO4: Apply Algorithms for solving problems like searching and sorting of data.

E. Practical Outcome(PRO)

Write a program to implement Bubble sort algorithm.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Sorting:

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

Bubble sort:

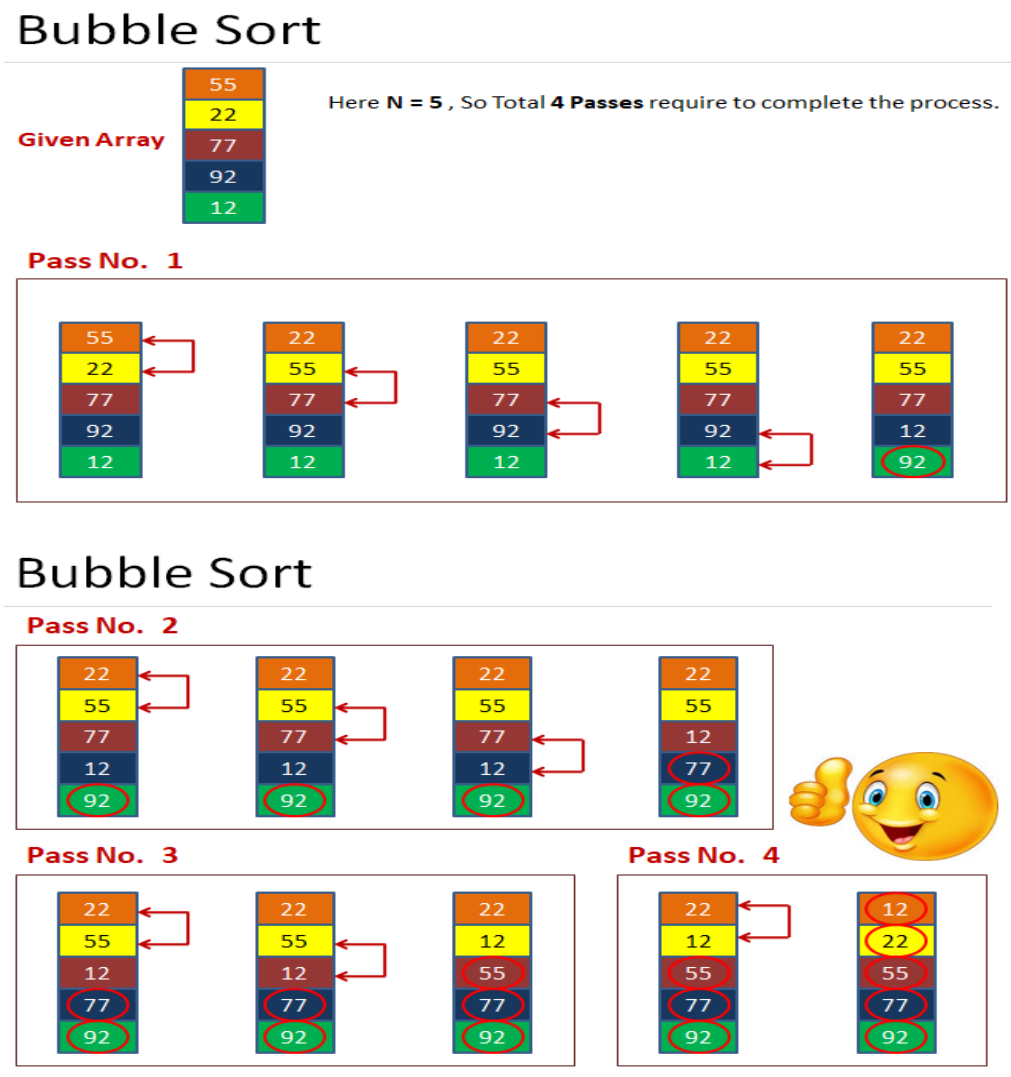
Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.

The basic steps to perform bubble sort algorithm:

- traverse from left and compare adjacent elements and the higher one is placed at right side.
- In this way, the largest element is moved to the rightmost end at first.
- This process is then continued to find the second largest and place it and so on until the data is sorted.

H. Program Logic-Flow chart:

Example: To implement Bubble sort algorithm.



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *How Bubble Sort Works?*

2. *What are the number of swapping's needed to sort the numbers 8, 22, 7, 9, 31, 5, 13 in ascending order, using bubble sort?*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.17: Implement Selection sort and Insertion sort algorithm.

A. Objective:

Apply Selection sort and Insertion sort algorithm to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using Selection sort and Insertion sort algorithm.*

D. Expected Course Outcomes(COs)

CO4: Apply Algorithms for solving problems like searching and sorting of data.

E. Practical Outcome(PRO)

Write a program to implement Selection sort and Insertion sort algorithm.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Selection Sort:

It is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list. This process is repeated for the remaining unsorted portion until the entire list is sorted.

The basic steps to perform selection sort algorithm:

- Set MIN to location 0
- Search the minimum element in the list
- Swap with value at location MIN
- Increment MIN to point to next element
- Repeat until list is sorted

Insertion Sort:

It is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

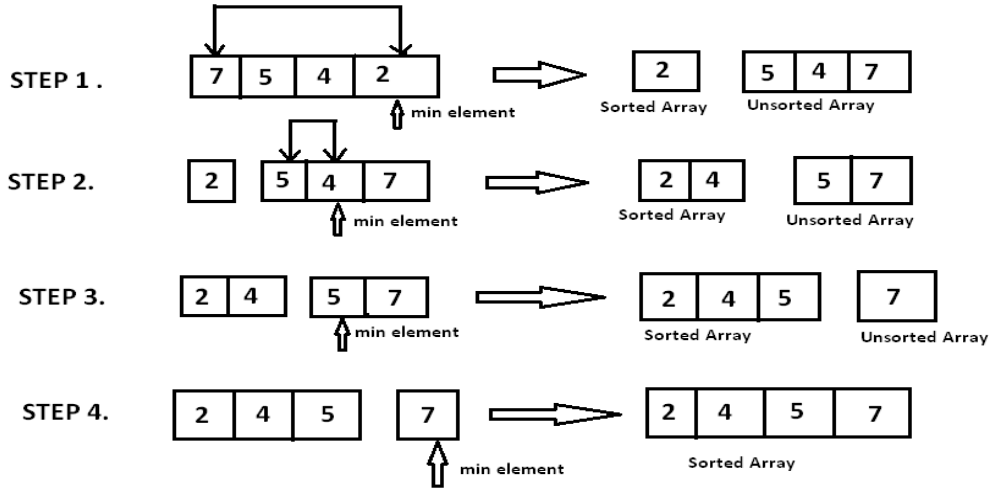
The basic steps to perform insertion sort algorithm:

- If it is the first element, it is already sorted. return 1
- Pick next element
- Compare with all elements in the sorted sub-list
- Shift all the elements in the sorted sub-list that is greater than the value to be sorted
- Insert the value
- Repeat until list is sorted

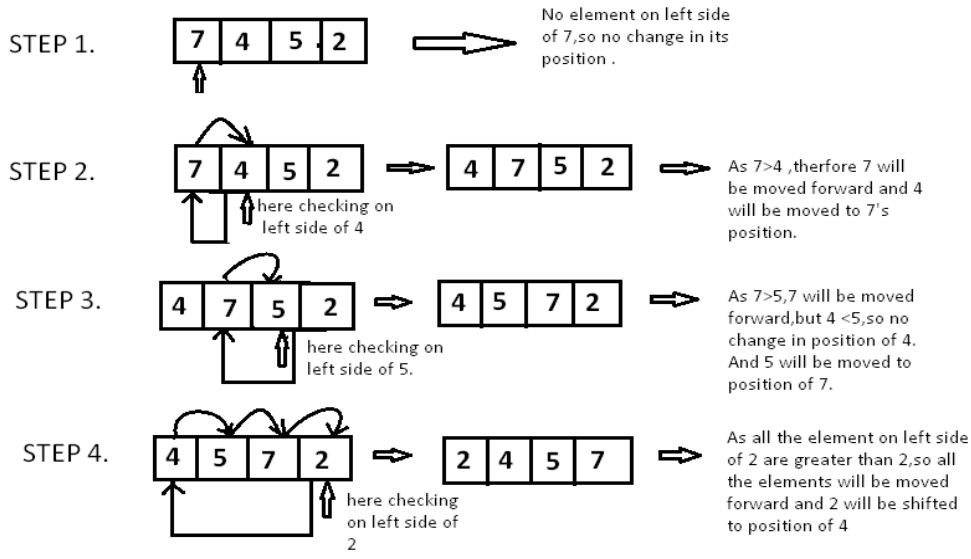
H. Program Logic-Flow chart:

Example: To implement Selection sort and Insertion sort algorithm.

Selection Sort:



Insertion Sort:



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *How Does the Selection Sort Algorithm Work?*

2. *What will be the number of passes to sort the elements using insertion sort? 14, 12,16, 6, 3, 10*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.18: Implement Merge sort algorithm.

A. Objective:

Apply Merge sort algorithm to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using Merge sort algorithm.*

D. Expected Course Outcomes(COs)

CO4: Apply Algorithms for solving problems like searching and sorting of data.

E. Practical Outcome(PRO)

Write a program to implement Merge sort algorithm.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Merge Sort:

Merge sort is yet another sorting algorithm that falls under the category of Divide and Conquer technique. It works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array. It is a recursive algorithm that continuously splits the array in

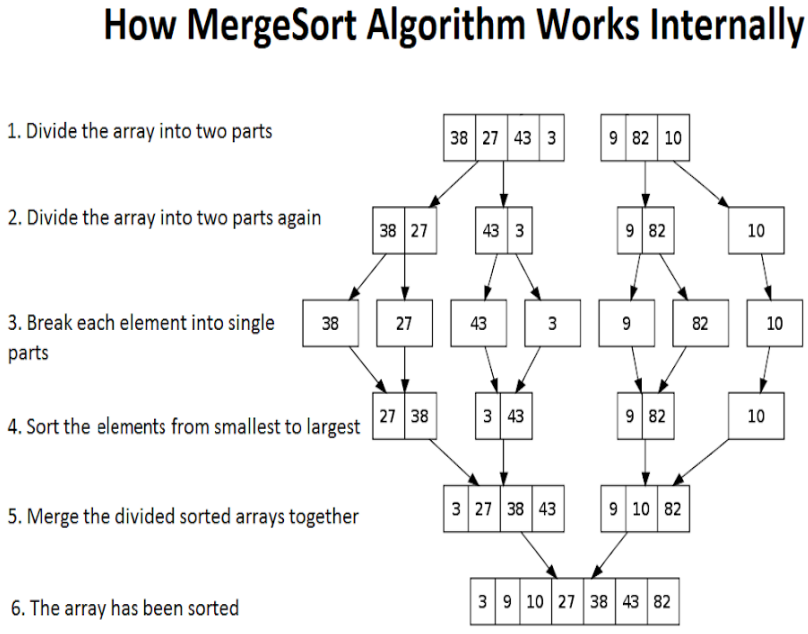
half until it cannot be further divided i.e., the array has only one element left (an array with one element is always sorted). Then the sorted subarrays are merged into one sorted array.

The basic steps to perform merge sort algorithm:

- If it is only one element in the list it is already sorted, return.
- Divide the list recursively into two halves until it can no more be divided.
- Merge the smaller lists into new list in sorted order.

H. Program Logic-Flow chart:

Example: To implement Merge sort algorithm.



I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *Merge sort uses which of the following technique to implement sorting?*

2. *How Merge Sort Works?*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.19: Implement construction of binary search trees.

A. Objective:

Apply binary search trees algorithm to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using binary search trees algorithm.*

D. Expected Course Outcomes(COs)

CO5: Implement nonlinear data structures like trees.

E. Practical Outcome(PRO)

Write a program to implement binary search trees algorithm.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Binary search trees:

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties:

- The left sub-tree of a node has a key less than or equal to its parent node's key.

- The right sub-tree of a node has a key greater than or equal to its parent node's key.

BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree. BST is a collection of nodes arranged in a way where they maintain BST properties.

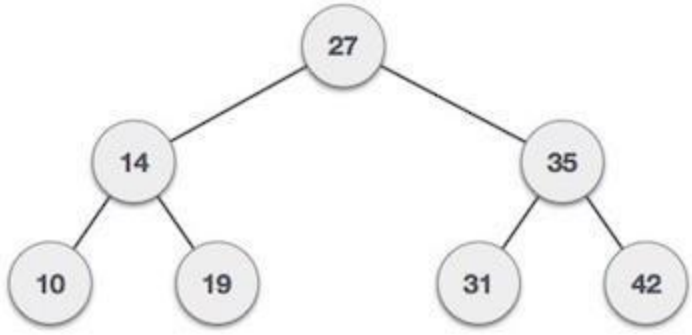
The basic steps to construct binary search tree:

- First, pick the first element of the array and make it root.
- Pick the second element, if its value is smaller than the root node value makes it left child,
- Else make it right child
- Now recursively call step (2) and step (3) to make a BST from its level Order Traversal.

H. Program Logic-Flow chart:

Example: To implement binary search trees algorithm.

Following is a pictorial representation of BST −



We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *What is a binary search tree?*

2. *How are binary trees used for sorting?*

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.20: Write a menu driven program to perform following operation on Binary Search Tree:

a. Create a BST.
b. Insert an element in BST.
c. Pre-order traversal of BST.
d. In-order traversal of BST.
e. Post-order traversal of BST.
f. Delete an element from BST

A. Objective:

Apply operation on Binary Search Tree to solve particular problem.

B. Expected Program Outcomes (POs)

1. ***Basic and Discipline specific knowledge:*** *Be able to apply engineering knowledge of computing appropriate to the problem*
2. ***Problem Analysis:*** *Identify and analyse well-defined engineering problems using codified standard methods.*
3. ***Design/ development of solutions:*** *Be able to design, implement, and evaluate a computer-based system, process, component, or program to meet the desired needs*
4. ***Engineering Tools, Experimentation and Testing:*** *Be able to use and apply current technical concepts and best practices in information technologies*
5. ***Project Management:*** *Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.*
6. ***Life-long learning:*** *Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.*

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. *Problem solving skill using operation on Binary Search Tree.*

D. Expected Course Outcomes(COs)

CO5: Implement nonlinear data structures like trees.

E. Practical Outcome(PRO)

Write a program to implement different operations on Binary Search Tree.

F. Expected Affective domain Outcome(ADOs)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

Basic Operations on Binary Search Tree:

Following are the basic operations of a tree −

Search − Searches an element in a tree.
Insert − Inserts an element in a tree.
Pre-order Traversal − Traverses a tree in a pre-order manner.
In-order Traversal − Traverses a tree in an in-order manner.
Post-order Traversal − Traverses a tree in a post-order manner.
Delete - Delete an element from BST

Search Operation:

Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Follow the same algorithm for each node.

The basic steps to perform search operation:

- Check whether the tree is empty or not
- If the tree is empty, search is not possible
- Otherwise, first search the root of the tree.
- If the key does not match with the value in the root, search its subtrees.
- If the value of the key is less than the root value, search the left subtree
- If the value of the key is greater than the root value, search the right subtree.
- If the key is not found in the tree, return unsuccessful search.

Insert Operation:

Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the right subtree and insert the data.

The basic steps to perform insert operation:

- If the tree is empty, insert the first element as the root node of the tree. The following elements are added as the leaf nodes.
- If an element is less than the root value, it is added into the left subtree as a leaf node.
- If an element is greater than the root value, it is added into the right subtree as a leaf node.
- The final leaf nodes of the tree point to NULL values as their child nodes.

Inorder Traversal:

The inorder traversal operation in a Binary Search Tree visits all its nodes in the following order:

- Traverse the left subtree, recursively
- Then, traverse the root node
- Traverse the right subtree, recursively

Preorder Traversal:

The preorder traversal operation in a Binary Search Tree visits all its nodes. However, the root node in it is first printed, followed by its left subtree and then its right subtree.

- Traverse the root node first.
- Then traverse the left subtree, recursively
- Later, traverse the right subtree, recursively

Postorder Traversal:

Like the other traversals, postorder traversal also visits all the nodes in a Binary Search Tree and displays them. However, the left subtree is printed first, followed by the right subtree and lastly, the root node.

- Traverse the left subtree, recursively
- Traverse the right subtree, recursively.
- Then, traverse the root node

Delete Operation:

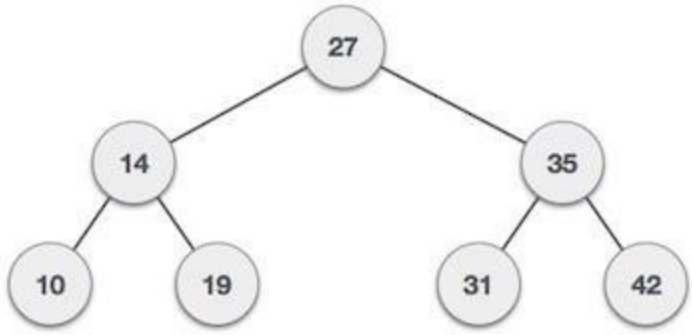
The basic steps to perform delete operation:

- Create a recursive function which returns the correct node that will be in the position.
- If the root is NULL, then return root (Base case)
- If the key is less than the root's value, then set root->left = deleteNode (root->left, key)
- If the key is greater than the root's value, then set root->right = deleteNode (root->right, key)
- Else check
    - If the root is a leaf node then return null
    - Else if it has only the left child, then return the left child
    - Else if it has only the right child, then return the right child
    - Otherwise, set the value of root as of its inorder successor and recur to delete the node with the value of the inorder successor.

H. Program Logic-Flow chart:

Example: To implement different operations on Binary Search Tree.

Following is a pictorial representation of BST −



We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

I. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	PYTHON IDE(any)		1

J. Safety and necessary Precautions followed

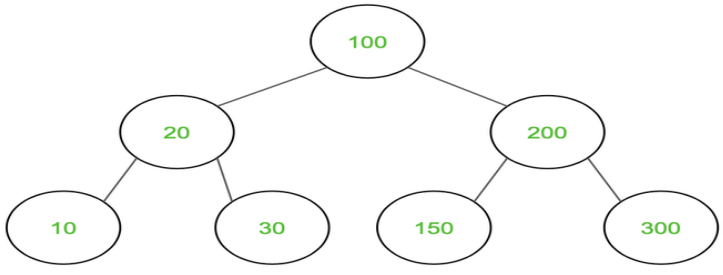
1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

K. Source code:

L. Input-Output:

M. Practical related Quiz.

1. *The task is to print the elements in inorder, preorder, and postorder traversal of the Binary Search Tree.*



Binary Search Tree

2. *Draw the binary search tree that is created if the following numbers are inserted in the tree in the given order.*
   12 15 3 35 21 42 14

N. References / Suggestions (lab manual designer should give)

1. *Data structures and algorithms in python by M.Goodrich*
2. *https://www.w3resource.com/python-exercises*

O. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Department of Information Technology

Data Structure with Python (4331601)

Lab manuals are prepared by

Ms. Ayesha S. Shaikh
Lecturer, Information Technology
RCTI, Ahmedabad

Shri Pradipsinh Chavda
Lecturer, Information Technology
LEC Polytechnic, Morbi

Ms. Jigna J Desai
Lecturer, Information Technology
GGP, Surat

Branch Coordinator

Shri Nandu Fatak

HOD-IT

GGP, Ahmedabad

Committee Chairman

Shri R. D. Raghani

(HOD-EC)

Principal (I/C)

Government Polytechnic, Gandhinagar