INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

INFORMATION TECHNOLOGY

TUTORIAL QUESTION BANK

Course Title	**OBJECT ORIENTED PROGRAMMING THROUGH PYTHON**				
Course Code	AITB01				
Program	B.Tech				
Semester	THREE				
Course Type	Core				
Regulation	**IARE - R18**				
Course Structure	**Theory**			**Practical**	
	Lectures	**Tutorials**	**Credits**	**Laboratory**	**Credits**
	3	-	3	-	-
Chief Coordinator	Ms. A Lakshmi, Assistant Professor				

COURSE OBJECTIVES:

The students will try to learn:	
I	The fundamental concepts of object-oriented approach for solving real-time problems.
II	The basic and advanced constructs of Python programming for developing object-oriented concepts.
III	The design concepts for developing user interface of real time applications.

COURSE OUTCOMES:

At the end of the course the students should be able to:

Course Outcomes		**Knowledge Level (Bloom's Taxonomy)**
CO 1	**Recall** the basic programming constructs in implementing in Python.	Remember
CO 2	**Identify** classes, objects, members of a class and relationship among them for real world entities.	Apply
CO 3	**Summarize** the object-oriented concepts such as Abstraction, Encapsulation, Inheritance and Polymorphism in real time context.	Understand
CO 4	**Demonstrate** abstraction feature with the help of python class properties	Understand

CO 5	**Make use** of polymorphism and inheritance concepts for achieving code reusability.	Apply
CO 6	**Apply** inbuilt strings for creating, performing basic operations and testing on text data.	Apply
CO 7	**Develop** user-defined functions for better modularity and a high degree of code reusability.	Apply
CO 8	**Explain** parameter-passing techniques while invoking recursive and non-recursive functions for solving problems.	Understand
CO 9	**Analyze** the Python exception mechanisms for handling errors and abnormal termination of program.	Analyze
CO 10	**Develop** user-defined exceptions for handling un-interrupted execution of specific programs.	Apply
CO 11	**Demonstrate** Python GUI tool kit for designing static user interfaces.	Understand
CO 12	**Make use** of widgets, containers and frames for creating user interface of web application.	Apply

MAPPING OF EACH CO WITH PO(s), PSO(s):

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
CO 1	2														
CO 2	2	3													
CO 3	2														
CO 4	2														
CO 5	2	4													2
CO 6	2	3													2
CO 7	2	4	5												2
CO 8	2	4													
CO 9	2	3													2
CO 10	2	6	5												2
CO 11	2	6	5		1										2
CO 12	2	6	5		1										2

TUTORIAL QUESTION BANK:

MODULE – I				
INTRODUCTION TO PYTHON AND OBJECT-ORIENTED CONCEPTS				
PART - A (SHORT ANSWER QUESTIONS)				
S No	**QUESTIONS**	**Blooms Taxonomy Level**	**How does this Subsume the level below**	**Course Outcomes**
1	List out the applications of Python programming language.	Remember	---	CO 1
2	Explain the role of Python interactive shell.	Understand	This would require the learner to **recall** the concept of interactive mode and script mode. Then **explain** how the python interactive shell will work.	CO 1
3	What are the different modes of working in Python?	Remember	---	CO 1
4	Explain the rules for identifier.	Understand	This would require the learner to **recall** identifier and then **understand** how an identifier will be written.	CO 1
5	How to check the number of keywords in Python?	Remember	---	CO 1
6	Demonstrate the standard data types in Python.	Understand	This would require the learner to **recall** what is data type and to **lis**t standard data types of python. Then **demonstrate** how data types are assigned to identifier	CO 1
7	Define a tuple.	Remember	---	CO 1
8	Define a List.	Remember	---	CO 1
9	Explain a set and its types.	Understand	This would require the learner to **recall** what is set data type and **demonstrate** variations in representing and using a set in python.	CO 1
10	Define a dictionary.	Remember	---	CO 1
11	List out the operators in Python.	Remember	---	CO 1
12	Define a control structure.	Remember	---	CO 1
13	What are the various types of loops in Python?	Remember	---	CO 1
14	Define a class.	Remember	---	CO 2
15	Define an object.	Remember	---	CO 2
16	Define a method.	Remember	---	CO 2
17	List out the features of object-oriented programming.	Remember	---	CO 2
18	Define Encapsulation.	Remember	---	CO 3
19	List various types of Inheritance.	Remember	---	CO 3
20	Define Abstraction.	Remember	---	CO 3

PART - B (LONG ANSWER QUESTIONS)				
1	Explain the features of Python programming language in detail.	Understand	This would require the learner to **recall** what is Procedural orientation, object orientation paradigms and then **explain** the reasons why python language is most powerful programming language.	CO 1
2	What is an operator and explain about the arithmetic operators and assignment operators in Python with an example?	Remember	---	CO 1
3	Explain about input statements in Python and formatting strings with examples.	Understand	This would require the learner to **recall** the purpose of providing values by user to identifiers and **demonstrate** the rules to accept values into python data type variables.	CO 1
4	Explain about features of Object Oriented Programming compared with the procedure-oriented programming.	Understand	This would require the learner to **recall** what is Procedural orientation, object orientation paradigms and then **demonstrate** the object oriented concepts Class, Object, abstraction, inheritance, and polymorphism.	CO 2
5	Explain in detail about the if statement and if-else statement with examples.	Understand	This would require the learner to **recall** the sequence flow of control in a program and need to deviate the sequence flow. Then **explain** the importance of branching flow control in problem solving.	CO 1
6	Demonstrate the concept of classes and objects in detail with any real time example.	Understand	This would require the learner to **recall** object orientation paradigms. Then **demonstrate** relationship between class and object.	CO 2
7	Illustrate the if-elif-else statement and while loop with examples.	Understand	This would require the learner to **recall** the sequence flow of control in a program and need to deviate the sequence flow. Then **explain** the importance of branching and iterative flow controls in problem solving.	CO 1
8	Explain about built-in data types and sequences in Python with examples.	Understand	This would require the learner to **recall** what are the standard data types in python. Then **understand** how sequences are stored and managed in python.	CO 1
9	Explain the set data type in Python and operations on set data types.	Understand	This would require the learner to **recall** what the standard data types in python are. Then **demonstrate** representation and manipulation of unordered collection of data through set data type.	CO 1
10	Explain about literals in Python and types of literals in Python with example.	Understand	This would require the learner to **recall** a literal and **explain** rules to represent and manage literals in python.	CO 1
11	Explain about encapsulation in Object Oriented Programming with example.	Understand	This would require the learner to **recall** object orientation paradigms concepts. Then **demonstrate** how data and methods bind together.	CO 3
12	Explain about output statements in Python and formatting strings with examples.	Understand	This would require the learner to **recall** the purpose of displaying results to user on output object and **demonstrate** rules to display results in preferred formats.	CO 1

13	Explain about abstraction in Object Oriented Programming with example.	Understand	This would require the learner to **recall** object orientation paradigms concepts. Then **explain** how unnecessary information will be hidden.	CO 3
14	Explain about user defined data types and constants in Python in detail.	Understand	This would require the learner to **recall** the data type and to list standard data types of python. Then **demonstrate** importance and management of custom data types in python.	CO 1
15	Explain about inheritance in Object Oriented Programming with example.	Understand	This would require the learner to **recall** object orientation paradigms concepts. Then **understand** how a class acquires properties of another class or classes.	CO 3
16	Demonstrate the logical operators and Boolean operators with example.	Understand	This would require the learner to **recall** the operators supported by python language and **explain** use of and, or & not operators to join two or more basic conditions and/or Boolean variables	CO 1
17	Explain about the unary operators and relational operators in Python with example.	Understand	This would require the learner to **recall** the operators supported by python language and **explain** the use of comparison operators in writing conditions.	CO 1
18	Explain about Bitwise operators and membership operators in Python with example.	Understand	This would require the learner to **recall** the operators supported by python language and **demonstrate** logical operators on binary numbers and membership operator.	CO 1
19	Demonstrate the for loop and the break statement and the continue statement in Python with examples.	Understand	This would require the learner to **recall** the sequence flow of control in a program and need to deviate the sequence flow. Then **explain** iterative control structure mechanism in problem solving.	CO 1
20	Explain about identity operators and operator precedence and associativity with example.	Understand	This would require the learner to **recall** theoperators supported by python language and **demonstrate** rules to be followed in evaluating expressions having combination of operators.	CO 1
PART - C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)				
1	Construct a Python program to create all possible strings by using 'a', 'e', 'i', 'o', 'u'. Use the characters exactly once.	Apply	This would require the learner to **remember**relevant control structure, string functions and **understand** them to generate strings satisfying the given constraints. By **applying** logic to output all possible combinations of strings with ovals.	CO 1
2	Construct code snippets in Python to perform the following a. Accessing elements of a tuple b. Modifying elements of a tuple c. Deleting elements of a tuple	Apply	This would require the learner to **recall** operations on tuple data type and **demonstrate** functions to manage a tuple. Then **solve** tuple management functions by **applying** code in python.	CO 1
3	Construct a Python program to count the number of words in a text.	Apply	This would require the learner to **recall**relevant control structure, string functions and **demonstrate** them to find words. By **applying** them to develop code for counting number of words.	CO 1

4	Construct a Python program using while loop first N numbers divisible by 5.	Apply	This would require the learner to **recall** while loop and **understand** the same to develop program to **find** numbers divisible by 5.	CO 1
5	Construct a simple program in Python to convert decimal number into binary, octal andhexadecimal number system in Python.	Apply	This would require the learner to **recall** number systems and control structures. Then **explain** conversion of one number system to other system and **apply** logic for number systems conversion.	CO 1

6	What is output of following code?
	class Count:
	def __init__(self, count=0):
	self__count=count
	a=Count(2)
	b=Count(2)
	print(id(a)==id(b), end = " ")
	c= "hello"
	d= "hello"
	print(id(c)==id(d))
		Remember	---	CO 1

7	Construct a Python program to get a string made of the first 2 and the last 2 chars from a given string. If the string length is less than 2, return instead of the empty string.	Apply	This would require the learner to **recall** therelevant string functions and **demonstrate** them to generated maximum 4 character strings and **apply** these functions to display desired strings.	CO 1
8	Construct a Python program to construct the following patternusing a nested for loop. *	Apply	This would require the learner to **recall** for loop and **understand** the working of nested for loops by **applying** the relevant logic to get the given pattern.	CO 1
9	Construct a Python program to add two positive integers without using the '+' operator.	Apply	This would require the learner to **recall** bitwise operators and **understand** these operators by **applying** logic for adding two integers without using arithmetic operators.	CO 1
10	Construct a Python program that prints all the numbers from 0 to 6 except 3 and 6.	Apply	This would require the learner to **recall** relevant loop structure and **understand** the break and continue statements by **applying** the acquired knowledge for the given program.	CO 1
MODULE – II				
PYTHON CLASSES AND OBJECTS				
PART – A (SHORT ANSWER QUESTIONS)				
1	Define Class with examples.	Remember	---	CO 2
2	Explain how object is createdandmethods are invoked in Python.	Understand	This would require the learner to **recall** the concept of creation of class and object. Then **explain** how the objects are created and methods are called using instance.	CO 2

3	Illustrate the use of init method in Python.	Understand	This would require the learner to **recall** the concept of method creation. Then **explain** how init() process variables	CO 4
4	Why Objects are mutable?	Remember	---	CO 2
5	List the features of the object-oriented programming through Python.	Remember	---	CO 2
6	What is inheritance?	Remember	---	CO 5
7	List different types of inheritance.	Remember	---	CO 5
8	Define namespace in Python.	Remember	---	CO 4
9	Explain self () in Python?	Understand	This would require the learner to **recall** the concept of class and objects. Then **understand** how self refers to variables and methods	CO 4
10	What is the difference between functions and methods in Python?	Remember	---	CO 2
11	Demonstrate Polymorphism in Python.	Understand	This would require t*he learner to **recall** the concept of polymorphism. Then **explain** different types of polymorphism	CO 5
12	Explain the concept of multiple inheritance.	Understand	This would require the learner to **recall** the concept of inheritance. Then **explain** multiple inheritance with an example	CO 5
13	What is operator overloading?	Remember	---	CO 5
14	What is mean single inheritance?	Remember	---	CO 5
15	Explain the purpose of inheritance in object oriented program in Python.	Understand	This would require the learner to **recall** the concept of inheritance. Then **demonstrate** how inheritance helps in code reusability	CO 5
16	What does the super() do in Python?	Remember	---	CO 5
17	Explain about __init__ in Python.	Understand	This would require the learner to **recall** the concept of method creation and **explain** the difference between method and init.	CO 4
18	Compare abstract class and interface.	Understand	This would require the learner to **recall** the concept of abstract class and interfaces. Then **explain** differences between abstract class and interfaces.	CO 5
19	What is abstract method in Python?	Remember	---	CO 5
20	Define multilevel inheritance.	Remember	---	CO 5
PART - B (LONG ANSWER QUESTIONS)				
1	Demonstrate the polymorphism working procedure with suitable example program in python.	Understand	This would require the learner to **recall** the concept of Polymorphism. Then **explain** the program with one of the polymorphism form.	CO 5
2	Demonstrate the use of inheritance. Explain with example and write a program for representing inheritance.	Understand	This would require the learner to **recall** the concept of Inheritance. Then **explain** the advantage of inheritance with an example program	CO 5
3	Explain all types of inheritance with suitable example programs in python.	Understand	This would require the learner to **recall** the concept of Inheritance .Then **explain** three types of inheritances with examples	CO 5

4	Demonstrate the following with example programs in python. i. Creating a class ii. Constructor iii. The self-variable	Understand	This would require the learner to **recall** the concept of creating class. Then **understand** how to create class, initialize variables using constructor and self-variable with an example. Program	CO 4
5	Explain a python program to represent the relation between class, objects and methods.	Understand	This would require the learner to **recall** the concept of classes and objects. Then **explain** how to create a class and instance of class and access variables of a class using methods.	CO 2
6	How do you resolve the name conflicts using namespaces? Explain with an example program in python.	Remember	---	CO 4
7	Explain a python program to find the volume and area of cube using super method.	Understand	This would require the learner to **recall** the concept of inheritance. Then **understand** base class to find the volume and area of cube and access those methods using super method of subclass.	CO 5
8	Construct a class as car and to print name of car company, model of car and manufacturing year using self-variable.	Apply	This would require the learner to **recall** the concept of class and objects. Then **demonstrate**car class and initialize the variables by **applying** the self –variable.	CO 4
9	Construct a python program to find the number of sides of square, rectangle and triangle using abstract class.	Apply	This would require the learner to **recall** the concept of abstract class and **understand** the formula for finding sides of square by **applying** abstract method in base abstract class then implement those methods in abstract sub classes.	CO 5
10	Why does the object-oriented philosophy need functions to be defined inside the classes? What could be the advantage?	Remember	---	CO 3
11	List different methods of realizing polymorphism and explain them with example program.	Remember	---	CO 5
12	Explain multiple views of an object with suitable example.	Understand	This would require the learner to **recall** the concept of classes and objects. Then **explain** how to create a class with multiple objects and invoking those objects	CO 2
13	Construct a python to print the different flying birds and non-flying birds using polymorphism.	Apply	This would require the learner to **recall** the concept of Polymorphism. Then **understand** duck type philosophy technique by **applying** different flying birds and non-flying birds using polymorphism.	CO 5
14	Explain differences between various types of inheritance?	Understand	This would require the learner to **recall** the concept of inheritances. Then **explain** three types of inheritance with examples	CO 5
15	Construct a program, which shows how to define a class, how to access member functions and how to create and access objects in Python.	Apply	This would require the learner to **recall** the concept of class and objects. Then **demonstrate** how to create a class with methods by **applying** the access member functions to access objects in Python.	CO 2
16	Construct a class as a person, which contains name and age. To print name and age of student using the person	Apply	This would require the learner to **recall** the concept of multiple inheritance. Then **understand** base class with name and sub	CO 2

	class in multiple inheritance.		class with age variable by **applying** thesuper method to implement multiple inheritance	
17	What is a nested class? What are its advantages? How it is defined and declared in Python?	Remember	---	CO 2
18	How multilevel inheritance is different from multiple inheritance? Explain with suitable examples?	Remember	---	CO 5
19	List out differences between abstract classes and interfaces with examples.	Remember	---	CO 5
20	Construct a python program to find the area of square using abstract method in python.	Apply	This would require the learner to **recall** the concept of abstract classes. Then **understand**the formulae to find area of square by **applying** base abstract class with square method and implement in subclass	CO 5
PART – C (PROBLEM SOLVING AND CRITICAL THINKING)				
1	Build a class whose object represents a complex number (A complex number contains a real part and an imaginary part). Write a program so that it is possible to add two objects of this class and store the result in third object.	Apply	This would require the learner to **recall** the concept of class and object .Then **understand** how to create a class with complex number by **applying** sum and display methods and use three instances to add and store complex numbers	CO 2
2	Explain public, private and protected access specifiers and show the ambiguity in multiple and multilevel inheritance.	Understand	This would require the learner to **recall** the concept of types of inheritance. Then **explain** how to identify the differences in multiple and multilevel inheritance and apply to access specifiers	CO 5
3	Construct a class called Time that has separate init member data for hours, minutes and seconds. One constructor should initialize this data to 0. And another should initialize it to fixed values. A member function should display it, in 11:59:59 format. Write a program to add time of two objects by overloading '+' operator.	Apply	This would require the learner to **recall** the init method in a class. Then **understand** two constructors by **applying** a member function to display in the given format by using overloading operator.	CO 2
5	Demonstrate various types of methods in object-oriented programming through python with examples.	Understand	This would require the learner to **recall** the concept of methods. Then **explain** three types of methods like instance, class, static with example programs	CO 3
6	Illustrate “Class is a template while Object is data”.	Understand	This would require the learner to **recall** the concept of class and object .Then **demonstrate** the class is a template while Object is data.	CO 2
7	Explain polymorphism as applied to OOP. Explain polymorphism with examples.	Understand	This would require the learner to **recall** the concept of object oriented concepts. Then **explain** that a variable, object or method exhibit different behavior in different context which can be applied to OOP with an example	CO 5
8	Construct a python program to assign 20, 30, 10 to ‘a’ and print these values	Apply	This would require the learner to **recall** the concept of namespace. Then **demonstrate** how to create class variable in class	CO 2

	in decreasing order using namespace in python.		namespace by **applying** those methods to print the values in decreasing order.	
9	How to override class methods in python? Describe with suitable example program.	Remember	---	CO 5
10	Explain how base class member functions can be invoked in a derived class if the derived class also has a member function with the same name.	Understand	This would require the learner to **recall** the concept of base and derived classes. Then **understand** how multiple inheritance can access super class methods	CO 2
MODULE – III				
STRINGS AND FUNCTIONS				
PART - A (SHORT ANSWER QUESTIONS)				
1	Define string. Write the syntax of creating a string with example	Remember	---	CO 6
2	"There is no difference between single quotes and double quotes while creating the string". Explain the statement.	Understand	This would require the learner to **recall**the concept of creation of strings. Then **explain** what is the difference between single-quotes and double quotesin strings.	CO 6
3	List different string operations. Write example programs for any three string operations.	Remember	---	CO 6
4	Illustrate the statement "strings are mutable or immutable".	Understand	This would require the learner to **recall**the concept of indexing in strings. Then **understand** strings are mutable or immutable**.**	CO 6
5	Define length of string and what is the predefined function used to find length of string? Illustrate with an example.	Remember	---	CO 6
6	Demonstrate the indexing concept in strings.	Understand	This wou1ld require the learner to **recall** the concept of indexing in procedural language. Then **explain** how indexing concept will useful in strings.	CO 6
7	Explain the methods that are used to find substrings in main string.	Understand	This would require the learner to **recall** the concept of substrings. Then **explain** what are the string methods to find substrings in main string.	CO 6
8	Explain about the following operations on strings. i) Concatenation of strings ii)Repeating	Understand	This would require the learner to **recall** the concept of strings. Then **understand** what are the methods for concatenation of strings and how string will repeat multiple times.	CO 6
9	Explain how to remove spaces from a string. Write related examples.	Understand	This would require the learner to **recall** the concept of strings. Then **explain** what are the methods to remove spaces from a string.	CO 6
10	Explain different string testing methods with examples.	Understand	This would require the learner to **recall** the concept of strings. Then **understand** what are the string testing methods in strings.	CO 6
CIE-II				
11	Define a function. Write the syntax of defining a function with example.	Remember	---	CO 7

12	Explain the process of calling a function.	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **explain** how to call a function.	CO 8
13	Explain the difference between functions returning single value and functions returning multiple values?	Understand	This would require the learner to **recall** the concept of functions. Then **explain** what is the difference between functions returning single value and returning multiple values.	CO 7
14	Compare actual and formal arguments with example.	Understand	This would require the learner to **recall** the concept of function arguments. Then **demonstrate** what are the actual and formal arguments in function calling.	CO 8
15	List different types of arguments. Define positional arguments.	Remember	---	CO 8
16	List the advantages of functions.	Remember	---	CO 7
17	Explain the difference between a function and method.	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **explain** what are the difference between a function and method in a class.	CO 7
18	Why functions in Python are called as first class objects?	Remember	---	CO 7
19	Construct a Python function that accepts two values and finds their sum.	Apply	This would require the learner to **recall** the concept of functions. Then **understand** how to pass two arguments in a function call by **applying** the relevant function that accepts two values and finds their sum.	CO 7
20	Define recursive function and illustrate with example program.	Remember	---	CO 8
PART – B (LONG ANSWER QUESTIONS)				
1	Summarize the escape characters that can be used in strings with an example.	Understand	This would require the learner to **recall** the concept of strings. Then **understand** what are the escape characters in strings with an example**.**	CO 6
2	Illustrate the following operations on strings. i)Length of string ii)Comparing strings iii)counting substrings in a string	Understand	This would require the learner to **recall** the concept of strings. Then **explain** how the different string operations performed on the strings.	CO 6
3	Explain the following methods. i)upper() ii)lower() iii)swapcase() iv)title()	Understand	This would require the learner to **recall** the concept of string functions. Then **explain** how the different string operations performed on the strings.	CO 6
4	Explain different strings and string testing methods with examples.	Understand	This would require the learner to **recall** the concept of strings. Then **demonstrate** how the different string testing methods performed on the strings with an example.	CO 6
5	Explain how can we split and join strings in Python with an example.	Understand	This would require the learner to **recall** the concept of strings operations. Then **explain** how can split and join the strings in python with an example.	CO 6
6	Construct a Python program to display all positions of a substring in a given main string.	Apply	This would require the learner to **recall** the concept of string indexing in strings. Then **understand**the substrings and main strings	CO 6

			by **applying** the relevant methods to display all positions of a substring in a given main string.	
7	Illustrate the concept of slicing the strings with an example program.	Understand	This would require the learner to **recall** the concept of string indexing in strings. Then **explain** how we can slicing the sting using indexing with an example.	CO 6
8	Identify the following methods that are used to remove spaces from a string. i) rstrip() ii)lstrip() iii)strip()	Apply	This would require the learner to **recall** the concept of string operations in strings. Then **understand** the spaces at different places in strings and **applying** the relevant methods to remove spaces from a string.	CO 6
9	Explain the methods that are useful to locate sub strings in a string with example programs.	Understand	This would require the learner to **recall** the concept of string indexing in strings. Then **explain,** what are the methods to locate a substring into a main string with an example.	CO 6
10	Explain various ways of assigning a group of characters to a variable?	Understand	This would require the learner to **recall** the concept of strings. Then **demonstrate**various ways of assigning a group of characters to a variable.	CO 6
CIE-II				
11	Explain the following. i)Assign a function to a variable ii)Define one function inside another function iii)Pass a function as parameter to another function	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **explain** what are the different ways of calling a function.	CO 7
12	Explain about i)Keyword arguments ii)Default arguments	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **understand** what are the keyword arguments and default arguments in function calling.	CO 8
13	Explain the role of Python interpreter in functions. Explain possible ways of assigning a function.	Understand	This would require the learner to **recall** the concept of functions in python. Then **explain** what are the possible ways of assigning a function.	CO 7
14	Explain the steps involved in Towers of Hanoi problem through recursion.	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **explain** how the recursion concept will be useful in real time applications.	CO 8
15	Explain how a function can return multiple values with an example.	Understand	This would require the learner to **recall** the concept of functions in procedural oriented programming. Then **explaining** how a function can return multiple values with example.	CO 8
16	Demonstrate about i)Keyword variable length arguments ii)Variable length arguments	Understand	This would require the learner to **recall** the concept of function calling arguments. Then **demonstrate** how keyword length arguments and variable length arguments will work in functions.	CO 8
17	Construct a python program to calculate factorial values of numbers from 1 to 10.	Apply	This would require the learner to **recall** the concept of functions. Then **understand** correct values for the attributes by	CO 8

			applying the method for factorial program without using recursion.	
18	List and explain different ways of passing values to function with examples.	Remember	---	CO 8
19	Construct a Python function to check the given number is prime or not.	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes and **applying** the relevant method to checking prime number or not for the given number.	CO 7
20	Construct a Python function to check the given number is palindrome or not.	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes and **applying** the relevant method to checking palindrome number or not for the given number.	CO 7
PART – C (PROBLEM SOLVING AND CRITICAL THINKING)				
1	Construct a Python program to access characters of a string using for loop.	Apply	This would require the learner to **recall** the concept of string indexing. Then **assigning** correct values for the attributes by **applying** the for loop to access the characters in a string.	CO 6
2	Construct a Python program that implements i)string concatenation ii)string comparison iii)string length	Apply	This would require the learner to **recall** the concept of string functions. Then **demonstrate** the two main strings and **applying** the string operations on different strings.	CO 6
3	Build a Python program to find the first occurrence of sub string in given main string.	Apply	This would require the learner to **recall** the concept of string functions. Then **demonstrate** the sub string and main string **by applying** these methods to find the first occurrence of sub string in given main string.	CO 6
4	Construct Python program that implements different string testing methods.	Apply	This would require the learner to **recall** the concept of string functions. Then **demonstrate** the different testing methods by **applying** those methods on different strings with an example.	CO 6
5	Develop a Python program to update or delete a string.	Apply	This would require the learner to **recall** the concept of strings. Then **understand** strings are immutable by **applying** the string indexing to update or delete a string.	CO 6
CIE-II				
6	Construct a Python function. i)To test whether a number is even or odd. ii)To calculate factorial value of numbers from 1 to 10	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes and **applying** the relevant method to test the programs based on the given criteria.	CO 7
7	Construct a Python program to understand the positional arguments of a function.	Apply	This would require the learner to **recall** the concepts of function calling. Then **understand** the positional argument and **applying** the relevant method for positional arguments.	CO 8
8	Predict the output of following code. def swap(x, y):	Create	This would require the learner to **recall** the syntax of function definition, **Explain** how	CO 7

	temp = x;
	x = y;
	y = temp;
	# Driver code
	x = 2
	y = 3
	swap(x, y)
	print(x)
	print(y)
			this program will work in these examples **Identify** the problem in the program and **analyze** the function definition and calling and **predict** the correct output for the program.	

9	Construct a Python function to sum all the numbers in a list. Sample List : (8, 2, 3, 0, 7) Expected Output : 20	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes by **applying** the relevant method to sum all the numbers in a given list.	CO 7
10	Construct a Python program to calculate factorial of a given number using recursion concept.	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes and **applying** the method for factorial of a given number using recursion.	CO 8
MODULE –IV				
EXCEPTION HANDLING				
PART – A (SHORT ANSWER QUESTIONS)				
1	How many except statements can a try-except block have?	Remember	---	CO 9
2	What is an exception?	Remember	---	CO 9
3	When will the else part of try-except-else be executed?	Remember	---	CO 9
4	Explain can we write only try block without catch and finally blocks.	Understand	This would require the learner to**recall** the concept of exception handling mechanisms .Then **explaining** which clauses are used along with try block to catch exceptions**.**	CO 9
5	Explain can we keep other statements in between try, catch and finally blocks.	Understand	This would require the learner to**recall** the concept of flow of control of try, catch and finallyblocks. Then **explaining** whether any statements are used with try, catch and finally blocks.	CO 9
6	How can you handle an exception?	Remember	---	CO 9
7	How can you catch multiple exceptions?	Remember	---	CO 9
8	What is try-except?	Remember	---	CO 9
9	What is try-finally statement?	Remember	---	CO 9
10	Explain raise syntax.	Understand	This would require the learner to **recall** the concept of throwing an exception manually. Then **explaining** how to write the syntax of raise statement tofind the text to print to the user.	CO 9
11	How to handle all types of exception with except?	Remember	---	CO 9
12	How to handle multiple exceptions with except?	Remember	---	CO 9

13	How to handle exceptions with try-finally?	Remember	---	CO 9
14	How to raise exception with arguments?	Remember	---	CO 9
15	Identify the type of error in the codes shown below. Print("Good Morning") print("Good night)	Apply	This would require the learner to **recall** the syntax of print() function. Then **assigning** correct syntax and **solving and correcting** the python code.	CO 9
16	Explain the following code valid. # Do something except: # Do something else: # Do something	Understand	This would require the learner to **recall** the concept of exception handling. Then **explaining** what happen whenwriting of code in the sequence try, except and else blocks.	CO 9
17	What is the output of the following code? Def foo(): try: return 1 finally: return 2 k = foo() print(k)	Remember	---	CO 9
18	What is the output of the following code? Def foo(): try:- print(1) finally: print(2) foo()	Remember	---	CO 9

19	What is the output of the following?
	Try:
	if'1' != 1:
	raise"someError"
	else:
	print("someError has not occurred")
	except"someError":
	print ("someError has occurred")
		Remember	---	CO 9

20	What is the output of the code shown below? #generator def f(x): yield x+1 g=f(8) print(next(g))	Remember	---	CO 9
PART – B (LONG ANSWER QUESTIONS)				
1	How the exceptions are handled in Python? Explain exception-handling mechanism in Python?	Remember	---	CO 9
2	Explain the different types of errors in Python? Give examples?	Understand	This would require the learner to **recall** the concept of errors. Then **explaining** how many types of errors in python and exception-handling mechanisms in Python.	CO 9
3	Explain in detail syntax of exception handling.	Understand	This would require the learner to**recall** the concept of try, except and finally blocks.	CO 9

			Then **explaining** how to write the syntax of exception handling.	
4	What is unreachable catch block error?	Remember	---	CO 9
5	Explain the hierarchy of exceptions in Python.	Understand	This would require the learner to **recall** hierarchy of exception. Then **explaining** how to code explicitly catch or rescue the raised exception and programmatically react to it in an appropriate manner.	CO 9
6	What are different types of exceptions in Python? Give example?	Remember	---	CO 9
7	Construct a python program to handle syntax error given by eval() function.	Apply	This would require the learner to **recall** eval() function then **understand** how to use eval() function for syntax error by **applying** those knowledge to solve syntax error. .	CO 9
8	Explain whether finally block get executed If either try or catch blocks are returning the control.	Understand	This would require the learner to **recall** the concept of finally block usage catch block. Then **explaining** this problem typically arises when there is no need to take any action in response to an exception.	CO 9
9	Explain whether we can throw an exception manually? If yes, how?	Understand	This would require the learner to **recal**l the methods. Then **explaining** how to handle exceptions in catch block and explaining what are the methods used to throw exception manually.	CO 9
10	What are the legal combinations of try,catch and finally blocks? Explain?	Remember	---	CO 9
11	Construct a program to create our own exception and raise it when needed?	Apply	This would require the learner to **recall** the concept of throwing exception manually. Then **understand** how to define custom exceptions by **applying** those methods depending on requirements and raising errors.	CO 10
12	What is runtime error in python? Explain with an example?	Remember	---	CO 9
13	How do you create customized exceptions in Python?	Remember	---	CO 10
14	Justify can one block of except statements handle multiple exceptions?	Evaluate	This would require the learner to **recall** the concepts of exception handling, **understand** the try-except blocks, **Apply** the procedure how to use multiple except blocks. **Analyze** the one block of except statements handle multiple exceptions and **justify** this statement	CO 9
15	Construct a python program to handle multiple exceptions.	Apply	This would require the learner to **recall** the methods to handle multiple exceptions then **understand** the code for **solving** multipleExceptions.	CO 9
16	What are assertions? Explain about the assertions.	Remember	---	CO 9
17	What is the difference between an exception and error? Explain with program?	Remember	---	CO 9
18	What are the rules in Python we need to follow when overriding a method	Remember	---	CO 9

	that throws an exception?			
19	How to handle exceptions with try-finally? Explain with a suitable example.	Remember	---	CO 9
PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)				
1	Explain what happens if the file is not found in the code shown below. A=False whilenot a: try: f_n = input(“Enter file name”) i_f = open(f_n, ‘r’) except: print(“Input file not found”)	Understand	This would require the learner to**recall** types of errors. Then **explaining** where exception is raised andunderstandthe output if the file is not found.	CO 9

2	Explain what is the output of the code shown below if the input entered is 6.
	valid = False
	whilenot valid:
	try:
	n=int(input("Enter a number"))
	while n%2==0:
	print("Bye")
	valid = True
	exceptValueError:
	print("Invalid")
		Understand	This would require the learner to**recall** types of errors. Then **understand** where exception is raisedandthe output by passing input as even number.	CO 9

3	Let’s take example in which trying to open a file in the READ mode. Then perform a WRITE operation on it. Upon execution, it’ll throw an exception.
	try:
	fob = open("test", "r")
	fob.write("It's my test file to verify exception handling in Python!!")
	except IOError:
	print "Error: can\'t find the file or read data"
	else:
	print "Write operation is performed successfully on the file"
	What is the output the above code produces?
		Remember	---	CO 9

4	Justify that we can either define an “**except**” or a “**finally**” clause with every try block. You can’t club these together. Also, you shouldn’t use the “**else**” clause along with a “**finally**” clause.	Evaluate	This would require the learner to **recall** the concepts of exception handling, **understand** the try-except blocks, **Apply** the procedure how to define else clause along with a finally clause. **Analyze** the except and finally clauses used with try block and **justify** this statement.	CO 9
5	Compare the two codes shown below and state the output if the input entered in each case is -6? CODE 1 import math	Analyze	This would require the learner to **recall** the types of errors.Then **understand** which type of error raised in the code 1 and code2 **Apply** those knowledge to find the exception raised **compare** the two codes	CO 9

	num=int(input("Enter a number of whose factorial you want to find")) print(math.factorial(num)) CODE 2 num=int(input("Enter a number of whose factorial you want to find")) print(math.factorial(num))			

6	What is the output of the following code?
	def a():
	try:
	f(x, 4)
	finally:
	print('after f')
	print('after f?')
	a()
		Remember	---	CO 9

7	What is the output of the code shown below? defgetMonth(m): if m<1 or m>12: raise ValueError("Invalid") print(m) getMonth(6)	Remember	---	CO 9
8	A try statement can have more than one except clause, to specify handlers for different exceptions. Explain with example program.	Understand	This would require the learner to **recall** try, except and final blocks. Then **explaining** a try statement can have more than one except clause, to specify handlers for different exceptions.	CO 9
9	In Python, you can use **else clause** on try-except block which must be present after all the except clauses. The code enters the else block only if the try clause does not raise an exception. Justify the above statement?	Evaluate	This would require the learner to **recall** the concepts of exception handling, **understand** the try-except blocks, **Apply** the procedure how to define else clause along with a try clause. **Analyze** the else block only if the try clause does not raise an exception and **justify** this statement.	CO 9
10	In Python Re-raising the exception, that has been caught in the **except** block. Explain in detail with a program?	Understand	This would require learner to **recall** the exception mechanisms. Then **explaining** how to code the re-raising exceptions that has been caught in the except block.	CO 9
MODULE –V				
GRAPHICAL USER INTERFACE				
PART – A (SHORT ANSWER QUESTIONS)				
1	Define root window.	Remember	---	CO 11
2	What are fonts and colors? Explain.	Remember	---	CO 11
3	Define containers.	Remember	---	CO 12
4	Define Canvas.	Remember	---	CO 12
5	Explain the types of Widgets.	Understand	This would require the learner to **recall** the concept of creating canvas. Then **explain** howto create widgets in root window and the types of widgets.	CO 12

6	Define frames.	Remember	----	CO 12
7	Define button widget.	Remember	---	CO 12
8	Explain label widget.	Understand	This would require the learner to **recall** the concept of creating frames. Then **explain** how to create label widgets in frames.	CO 12
9	Explain message widget.	Understand	This would require the learner to **recall** the concept of creating frames. Then **understand** how to create message widgets in frames.	CO 12
10	Define radio button Widget.	Remember	---	CO 12
11	Define entry widget.	Remember	---	CO 12
PART - B (LONG ANSWER QUESTIONS)				
1	Explain various types of containers and working procedure of containers with suitable examples.	Understand	This would require the learner to **recall** the concept of creating root window. Then **explain**how to create containers in root window and working procedure of containers.	CO 12
2	Construct the python program for canvas and frames.	Apply	This would require the learner to **recall** the attributes and its methods. Then **assigning** correct values for the attributes by**applying** the method for creating canvas and frames.	CO 12
3	How to create a button widget in Python? Write suitable python code for button widgets.	Remember	---	CO 12
4	Discuss the working procedure of label widgets with suitable example programs in python.	Create	This would require the learner to **recall** the concepts of widgets and GUI, **understand** the applications of widgets, **Identify** the label widgets, **Analyze** their attributes and **develop** the required python GUI program.	CO 12
5	What are the differences between message widget and text widget in object-oriented programming?	Remember	---	CO 12
6	Explain the working procedure of message widgets with suitable example program.	Understand	This would require the learner to **recall** the concept of creating frames. Then **explaining** how to create message widgets in root window and working procedure of message widget with example.	CO 12
7	Discuss working procedure of text widget with suitable python code.	Create	This would require the learner to **recall** the concepts of widgets and GUI, **understand** the applications of widgets, **Identify** the text widget, **Analyze** their attributes and **develop** the required python GUI program.	CO 12
8	How to create radio button widget in python? Write suitable python program for button widgets.	Remember	---	CO 12
9	Discuss the working procedure entry widget with suitable example python program.	Create	This would require the learner to **recall** the concepts of widgets and GUI, **understand** the applications of widgets, **Identify** the entry widget, **Analyze** their attributes and **develop** the required python GUI program.	CO 12
10	Demonstrate form application from the experimental machine learning to	Understand	This would require the learner to **recall** the concept of statistics Then **explaining** how	CO 11

	interactive with data mining exploration using Python		to explore data mining applications in machine learning using python programming.	
11	What is Python widget? Explain interactive linear and nonlinear regression model	Understand	This would require the learner to **recall** the concept of statistics Then **explaining** how to create the linear and nonlinear regression models.	CO 11
12	What exactly are “containers” in Python? What are all the Python container types?	Remember	---	CO 12
13	How do you create a GUI in Python? Is Python good for desktop application?	Remember	---	CO 11
14	Create a Python GUI program that produces a window with the following widgets 1. A text box to display the value of one element of a given list 2. A button to retrieve the previous value in that list (if there is one). This button is displayed if there is no previous value in the list	Create	This would require the learner to **recall** the concepts of widgets and GUI, **understand** the applications of widgets, **Identify** the textbox and button widgets, **Analyze** their attributes and **develop** the required python GUI program.	CO 12
15	Create a Python GUI program that produces a window with the following widgets 1. A button to retrieve the next value in that list (if there is one). This button is displayed if there is no next value in the list 2. A label to display the number of the item being displayed and the total number of items.	Create	This would require the learner to **recall** the concepts of widgets and GUI, **understand** the applications of widgets, **Identify** the labels and button widgets, **Analyze** their attributes and **develop** the required python GUI program.	CO 12

Prepared by:
Ms. A Lakshmi, Assistant Professor **HOD, IT**