MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(Formerly West Bengal University of Technology)
Syllabus of BCA
(Effective from 2023-24 Academic Sessions)

SEMESTER: III

DEFINITION OF CREDIT

1 HR LECTURE PER WEEK	1 CREDIT
1 HR TUTORIAL PER WEEK	1CREDIT
2 HR PRACTICAL PER WEEK	1 CREDIT

SUBJECT NUMBERING SCHEME:

CODE FOR THE DEPT. OFFERING SUBJECT	SUBJECT TYPE	SEM	SUBJECT CODE

C	CORE MAJOR

SUBJECT NAME: Python Programming **Credit: 3L + 2P**
SUBJECT CODE: BCAC301

COURSE OBJECTIVE:

The course objectives of a Python programming course typically aim to equip students with the fundamental knowledge and skills needed to understand and utilize Python as a programming language effectively. students should have a solid foundation in Python programming, enabling them to write Python code independently, understand and contribute to Python-based projects, and pursue further specialization in specific areas of Python development.

COURSE OUTCOME	
CO1	Will gain a solid understanding of Python programming fundamentals, including syntax, data types, control structures, and functions.
CO2	Will learn techniques for acquiring, cleaning, and analyzing data

CO3	Will be able to design and implement modular programs using functions and modules to improve code reusability and maintainability.
CO4	Will be capable of reading from and writing to files, as well as handling different file formats for data input and output operations.
CO5	Will understand the principles of object-oriented programming and be able to create classes, objects, and inheritance hierarchies to model real-world entities and solve problems.

DETAILED SYLLABUS:

Module No:	**NAME OF THE TOPIC**	**HOURS**	**MARKS**
M1	**INTRODUCTION:** Features of Python, Execution of Python Program, Viewing the byte code, Python Virtual machine, Frozen binaries, Memory management of Python, Compare between C and Python, Comapre between Java and Python	**3**	**5**
M2	**Python Fundaments**: python character set, Tokens (Keywords, Identifiers, Literals, Operators, Punctuations), Comments in Python (Single line and Multi line), Variables and assignments (Creating a variables, Multiple assignments)	**3**	**4**
M3	**Data Handling and Flow Control:** Data Types, Mutable and Immutable types, Operators, Negative Number artithmetic in Python, Evaluation of Expression, Type casting Flow Control: if statement, if..else, if..elif..else statement, range function,while loop, for loop, nested loop, break statement, Continue statement, return statement	**6**	**6**

M4	**String and Character:** Introduction, Traversing the string,String concatenation and replication, Membership operator, comparision operator, determine unicode value of single character, slicing, built in functions (len(), capitalize(), count(), find(), index(),isalpha(), isalnum(), isdigit(), isspace(), islower(),lower(), upper(), strip(0, lstrip(), rstrip(), join(), title(), split(), partition(), endswith(), startswith(), replace()	**5**	**5**
M5	Functions: Differenece Function and method,define a function, Calling a function, Return results from function, Return multiple values from function, Formal and Actual arguments, Positional arguments, Default arguments, Keyword arguments, Variable length arguments Local and Global variable, Recursive function, using Lambdas with filter(), lambdas with map()	**4**	**5**
M6	**Lists**: Creation of list,empty list, nested list, use of list(), Accessing list, length of list, indexing and slicing of list, Traverse the list, Compare the list, Joining the list, Replication of list, Making the true copy of list, index(0, append() and extend(), insert(), pop(), popitem(), del and clear(), count(), reverse(), sort and sorted, two dimensional list	**6**	**10**
M7	Tuples: Creation of tuple (empty tuple, single element, create tuple from existing sequence, nested tuple), Accessing tuples, Traverse tuple, join , len(), max(), min()	**3**	**5**

M8	Dictionary: Creating dictionary empty dictionary, add key:value pair in dictionary, use of dict(), specify value pair separately in sequence, Add ewlements to dictionary, Check existence of a key in dictionary, get(), items(), keys(), values(), len(), fromkeys(), extens/ update dictionary with new value, making shallow copy of dictionary, delete elemnts from dictionary(clera(), pop(), poitem(), del) , max(), min(), sum()	**6**	**10**
M9	Text file: opening a text file, text file open modes (r, r+, w, w+, a, a+), closing a text file, opening a file using with clause, writing/appending data to a text file using write() and writelines(), reading from a text file using read(), readline() and readlines(), seek and tell methods, manipulation of data in a file	**3**	**5**
M10	Class and Object: Features of OOPs Programming, Creation of class, self cvariable, Constructor, types of variable, namespace, types of methods(Instance method, class methos, static method) Inheritance: Constructor in inheritance, super(), Types of inheritance, Method of resolution order, Ploymorphism, operator and method overloading, abstract class and interface	**6**	**15**
	INTERNAL EXAMINATION	**3**	**30**
	TOTAL	**48**	**100**

Practical:

SUBJECT NAME: Python Programming Lab **Credit: 2**
SUBJECT CODE: BCAC391

List of Practical:

1. Fizz Buzz: Write a program that prints the numbers from 1 to 100. But for multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "Fizz Buzz".
2. Palindrome Checker: Write a function to determine if a given string is a palindrome (reads the same forwards and backwards). Ignore spaces, punctuation, and capitalization.
3. Factorial Calculation: Write a function to calculate the factorial of a given number recursively.
4. Prime Number Generator: Write a function to generate a list of prime numbers up to a given number using the Sieve of Eratosthenes algorithm.
5. Word Count: Write a program that takes a string as input and counts the frequency of each word in the string. Ignore case and punctuation.
6. Reverse a Linked List: Implement a function to reverse a singly linked list in-place.
7. Binary Search: Implement the binary search algorithm to find the index of a given element in a sorted list.
8. Anagram Checker: Write a function to determine if two strings are anagrams of each other (contain the same characters in a different order).
9. Matrix Transpose: Write a function to transpose a given matrix (convert rows to columns and vice versa).
10. Tower of Hanoi: Implement the Tower of Hanoi puzzle using recursion.

List:

1. **Sum of List Elements**: Write a program that calculates the sum of all elements in a list of numbers.
2. **Maximum and Minimum Element in List**: Write a program to find the maximum and minimum elements in a list.
3. **List Reversal**: Write a program to reverse a given list.
4. **List Sorting**: Write a program to sort a list of numbers in ascending or descending order.
5. **List Filtering**: Write a program to filter out even or odd numbers from a list.
6. **List Concatenation**: Write a program to concatenate two lists into one.
7. **List Intersection**: Write a program to find the intersection of two lists (i.e., elements that appear in both lists).
8. **List Union**: Write a program to find the union of two lists (i.e., all unique elements from both lists).
9. **List Flattening**: Write a program to flatten a nested list (i.e., convert a list of lists into a single list).
10. **List Element Removal**: Write a program to remove all occurrences of a specific element from a list.

11. **List Rotation**: Write a program to rotate a list by a given number of positions.
12. **List Comprehensions**: Write a program to generate a new list based on a given list using list comprehensions (e.g., square each element of a list).

Dictionary:

Word Frequency Counter:
Write a program that takes a string as input and counts the frequency of each word using a dictionary. Ignore case and punctuation.

Merge Two Dictionaries:
Write a function to merge two dictionaries into one, where the values of duplicate keys are added together.

Dictionary Key Sort:
Write a function to sort the keys of a dictionary in alphabetical order and return a new dictionary with the sorted keys.

Nested Dictionary Access:
Write a function to access a value in a nested dictionary given a list of keys. For example, given the dictionary {'a': {'b': {'c': 1}}} and the keys ['a', 'b', 'c'], the function should return 1.

Dictionary Inversion:
Write a function to invert a dictionary, where the keys become values and the values become keys. Assume that the values are unique.

Dictionary Intersection:
Write a function to find the intersection of two dictionaries (i.e., keys that appear in both dictionaries) and return a new dictionary with the common keys and their values.

SUGGESTED READING:

1. Python **Programming: A Modular Approach" by Sheetal Taneja and Naveen Kumar** (Publisher: Oxford University Press India)
2. **Python for Beginners: A Step-by-Step Guide to Learn Python from Zero with Hands-on Exercises" by Ajit Kumar** (Publisher: BPB Publications)
3. "**Python: A Practical Introduction to Programming" by Subin Siby** (Publisher: BPB Publications)
4. "**Python Programming: Problems and Solutions" by S.S. Srivastava and M.H. Khan** (Publisher: Khanna Publishers)
5. "**Python Programming: A Beginner's Guide to Learn Python in 7 Days" by Darshan Patel** (Publisher: BPB Publications)

SUBJECT NAME: Data Structure through C **Credit: 3L + 2P**
SUBJECT CODE: BCAC302

COURSE OBJECTIVE:

The course aims to provide students with a solid foundation in fundamental data structures and algorithms, as well as proficiency in implementing them using C. This will empower students with the knowledge, skills, and problem-solving abilities necessary to tackle complex computational problems and excel in their academic and professional pursuits in the field of computer application.

COURSE OUTCOME	
CO1	Students will comprehend the fundamental concepts of data structures, including arrays, linked lists, stacks, queues, and trees, and how they are implemented in the C programming language.
CO2	Gain proficiency in implementing various data structures using C programming language, including dynamic memory allocation, pointers, and structures.
CO3	Develop the ability to analyze problems and choose appropriate data structures and algorithms to solve them efficiently.
CO4	Enhance problem-solving skills by applying data structures and algorithms to solve real-world problems and algorithmic challenges.
CO5	Collaborate effectively in team projects involving the design and implementation of complex data structures and algorithms, fostering communication and teamwork skills.

DETAILED SYLLABUS:

Module No:	**NAME OF THE TOPIC**	**HOURS**	**MARKS**
M1	Structure and Union, typedef definition, Implementation of structure and Union, Accessing members of the structure, Pointer to structure,passing structure in function, Passing structure through pointer, Self-referential pointer, Nested Structure	**4**	**5**
M2	Arrays: 1D, 2D and Multi-Dimensional Arrays, Sparse Matrices. Polynomial representation, Implementation of Stack and Queue, Example of Infix, Postfix, and prefix, Priority Queue	**7**	**10**

M3	Linked Lists : Singly, Doubly and Circular Lists, Normal and Circular representation of Self Organizing Lists, Skip Lists, Polynomial representation, Implementation of Stack and Queue, Circular List, Stack as Circular list, Queue as Circular list	**8**	**15**
M4	Recursion: Definition, Internal Stack representation, Factorial function, Fibonacci Sequence, Binary Search, The tower of Hanoi Problem	**5**	**8**
M5	Trees : Introduction to Tree as a data structure, Binary Trees (Insertion, Deletion, Recursive and Iterative Traversals of Binary Search Trees), Threaded Binary Trees (Insertion, Deletion, Traversals), Height-Balanced Trees (Various operations on AVL Trees).	**8**	**15**
M6	Searching and Sorting: Linear Search, Binary Search, Comparison of Linear and Binary Search, Selection Sort, Insertion Sort, Merge Sort, Quick sort, Shell Sort, Comparison of Sorting Techniques	**8**	**12**
M7	Hashing : Introduction to Hashing, Deleting from Hash Table, Efficiency of Rehash Methods, Hash Table Reordering, Resolving collision by Open Addressing, Coalesced Hashing, Separate Chaining, Dynamic and Extendible Hashing, Choosing a Hash Function, Perfect Hashing	**5**	**5**
	INTERNAL EXAMINATION	**3**	**30**
	TOTAL	**48**	**100**

Practical:

SUBJECT NAME: Data Structure Lab **Credit:2**
SUBJECT CODE: BCAC392

List of Practical:
1. Implementation of array operations.
2. Stacks and Queues: adding, deleting elements.
3. Circular Queue: Adding & deleting elements
4. Merging Problem: Evaluation of expressions operations on Multiple stacks & queues
5. Implementation of linked lists: inserting, deleting, and inverting a linked list.
6. Implementation of stacks & queues using linked lists:

7. Polynomial addition, Polynomial multiplication
8. Sparse Matrices: Multiplication, addition.
9. Recursive and Non Recursive traversal of Trees Threaded binary tree traversal. AVL tree implementation Application of Trees.
10. Application of sorting and searching algorithms Hash tables' implementation: searching, inserting and deleting, searching & sorting techniques.
Assignments:
Based on the curriculum as covered by the subject teacher

SUGGESTED READING:

1. **Data Structures Through C in Depth** by S. K. Srivastava and Deepali Srivastava - BPB Publications
2. **Data Structures Through C** by Yashavant Kanetkar - BPB Publications
3. **Data Structures: A Pseudocode Approach with C** by Richard F. Gilberg and Behrouz A. Forouzan (Adapted by Dinesh P. Mehta) - Cengage Learning India
4. **Data Structures and Algorithm Analysis in C** by Mark Allen Weiss (Adapted by Dinesh Mehta) - Pearson Education India
5. **Data Structures Using C and C++** by Tanenbaum - Pearson Education India
6. **Data Structures and Algorithms Made Easy** by M. S. Kutti Swamy - Pearson Education India