# Government Degree College, (Autonomous), Baramulla

Semester 8TH Major Course

**INFORMATION TECHNOLOGY**

Title: Python Programming Course code: BITC3822M

Credits (4+2): Theory 04, Lab -02) Contact hours: 60 (T) + 60 (P)

**Course Objectives:**

The course aims to introduce the comprehensive concepts of the Python programming language and develop a strong understanding of its syntax, data types, operators, and control structures. It focuses on building problem-solving and logical thinking skills through structured programming while enabling students to write modular and reusable code using functions and scope. The course also familiarizes learners with essential data structures such as lists, tuples, sets, and dictionaries, along with string manipulation and file handling techniques for data processing. Overall, it equips students with the ability to design, implement, and debug efficient Python programs applicable to real-world problems.

**Course Learning Outcomes**

Upon successful completion of this course, students will be able to:

- *Understand the structure of a Python program and apply fundamental concepts including data types, operators, type conversion, and variable handling.*
- *Write, debug, and execute Python programs effectively using an interpreter and standard development environment.*
- *Use control structures such as conditional statements and loops to control program flow and solve logical problems.*
- *Define and use functions with various argument types, understand scope, apply recursion, and work with lambda functions and built-in modules.*
- *Apply core data structures: lists, tuples, sets, and dictionaries to organize, store, and manipulate data efficiently.*
- *Perform a wide range of string operations and use string formatting techniques for clean and readable output.*
- *Read from and write to text, CSV, and JSON files, and manage file operations using context managers.*
- *Handle runtime errors and exceptions gracefully using try, except, else, finally, and custom exception classes.*
- *Develop simple, structured, real-world Python applications by combining multiple concepts learned throughout the course.*

# Government Degree College, (Autonomous), Baramulla

**UNIT I:**

Origins and History of Python, Structure of a Python Program, Interpreter Shell, Indentation, Comments, Identifiers and Keywords, Literals, Basic Operators (Arithmetic, Relational, Logical/Boolean, Assignment, Bitwise, Membership operators in, not in, Identity operators is, is not), Operator Precedence and Associativity, The Integer, Floating-Point, String, and Boolean Data Types, Type Conversion and Type Casting, String Concatenation and Replication, Storing Values in Variables, Multiple Assignment, Constants. Input and Output Statements, Formatted Output using print(), Control Statements: Branching (if, if-else, if-elif-else, Nested if), Looping (while, for), Range Function, break, continue, pass, exit(), Nested Loops, Loop with else clause.

**UNIT II:**

Defining and Calling a Function, def Statements with Parameters, Formal and Actual Arguments, Positional Arguments, Keyword Arguments, Default Arguments, Variable Length Arguments (*args, **kwargs), Return Values and return Statements, The None Value, Local and Global Scope, The global Statement, The nonlocal Statement, Nested Functions, Recursion and Recursive Functions, Lambda Functions, Built-in Functions (len, range, type, print, input, abs, round, etc.), Importing Modules, Standard Library Modules (math, random, os, sys, datetime), Creating User-Defined Modules.

**UNIT III:**

*Lists*: Creating Lists, Basic List Operations, Indexing and Slicing, Negative Indexing, Built-in Functions on Lists, List Methods (append, insert, remove, pop, sort, reverse, copy, extend, etc.), Nested Lists, List Comprehension, Unpacking Lists. *Tuples*: Creating Tuples, Basic Tuple Operations, Indexing and Slicing, Built-in Functions on Tuples, Tuple Methods, Nested Tuples, Tuple Unpacking, Differences between Lists and Tuples. *Sets:* Creating Sets, Basic Set Operations (Union, Intersection, Difference, Symmetric Difference), Set Methods (add, remove, discard, pop, clear, etc.), Frozen Sets, Traversing Sets, Set Comprehension. *Dictionarie*s: Creating Dictionaries, Accessing and Modifying Key-Value Pairs, Nested Dictionaries, Built-in Functions on Dictionaries, Dictionary Methods (get, keys, values, items, update, pop, setdefault, etc.), Dictionary Comprehension, Iterating over Dictionaries.

**UNIT IV:**

*Strings:* Creating and Storing Strings, Basic String Operations, Accessing Characters by Index, Negative Indexing, String Slicing and Joining, String Methods (upper, lower, strip, split, replace, find, count, startswith, endswith, etc.), Formatting Strings (format(), f-strings), Raw Strings, Multiline Strings. *Exception Handling:* Introduction to Errors and Exceptions, Types of Errors (Syntax Errors vs Runtime Errors), Common Built-in Exceptions (ZeroDivisionError, ValueError, TypeError, IndexError, KeyError, FileNotFoundError, NameError, etc.), The try and except Block, Handling Multiple Exceptions, The else Clause in Exception Handling, The finally

# Government Degree College, (Autonomous), Baramulla

Block, Raising Exceptions using raise, Creating User-Defined (Custom) Exceptions, Nested try Blocks.

*Files:* Types of Files (Text and Binary), Opening and Closing Files, File Modes (r, w, a, rb, wb), Reading Files (read(), readline(), readlines()), Writing and Appending to Files, The with Statement (Context Manager), Working with CSV Files, Working with JSON Files, Exception Handling in File Operations (try, except, finally).

**LIST OF PRACTICALS:**

1. Write programs to demonstrate the use of all types of operators: arithmetic, relational, logical, bitwise, membership, and identity. Include examples of type conversion and multiple variable assignment.
2. Write a program that accepts user input for name, age, and marks, performs type casting where necessary, and displays a formatted output using both format() and f-strings.
3. Write programs using if, if-else, if-elif-else, and nested if to solve problems such as finding the greatest of three numbers, checking whether a number is positive/negative/zero, and a simple grade calculator based on marks.
4. Write programs using for and while loops to print multiplication tables, compute factorial of a number, print Fibonacci series, and demonstrate the use of break, continue, and pass.
5. Write programs to define and call functions using positional, keyword, default, and variable-length arguments (*args, **kwargs). Implement a recursive function to compute factorial and to solve the Tower of Hanoi problem.
6. Write programs to demonstrate lambda functions for sorting and filtering data. Import and use standard library modules including math, random, os, and datetime to perform relevant operations. Create and import a simple user-defined module.
7. Write programs to create and manipulate lists - perform indexing, slicing, and use list methods such as append, insert, remove, sort, and reverse. Implement list comprehension to generate squares, filter even numbers, and flatten a nested list.
8. Write programs to demonstrate tuple creation, unpacking, indexing, and immutability. Perform set operations including union, intersection, difference, and symmetric difference. Demonstrate the use of frozen sets and set comprehension.
9. Write programs to create dictionaries, access and modify key-value pairs, and use dictionary methods such as get, keys, values, items, update, and pop. Implement dictionary comprehension and write a program to count word frequency in a sentence using a dictionary.
10. Write programs to perform string slicing, reversing, and searching. Use string methods such as split, join, strip, replace, upper, lower, find, and count. Write a program to check if a string is a palindrome and to count vowels and consonants.
11. Write programs to create, read, write, and append data to text files using the *with* statement for safe file handling. Write a program that reads a text file and counts the number of lines, words, and characters.

12. Write programs to read from and write to CSV files using the csv module and read and write JSON data using the json module. Implement a simple student record system that stores and retrieves data from a JSON file.
13. Write programs to demonstrate the use of try, except, else, and finally blocks. Handle common exceptions such as ZeroDivisionError, ValueError, FileNotFoundError, and IndexError. Write a program that defines and raises a custom exception for invalid input.
14. Write a program that robustly handles file operations including file not found, permission errors, and end-of-file conditions using appropriate exception handling with nested try blocks where necessary.

**TEXT BOOKS:**

1. Gowrishankar S., Veena A., *Introduction to Python Programming, 1st Edition*, CRC Press/Taylor & Francis, 2018.
2. Downey A.B., *Think Python: How to Think Like a Computer Scientist*, 3rd edition, 2015.
3. Taneja S. & Kumar N., *Python Programming: A Modular Approach*, Pearson Education, 2017.

**REFERENCES:**

1. Jake VanderPlas, *Python Data Science Handbook*, O'Reilly Media, 2016.
2. Aurelien Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media, 2019.