# ITM (SLS) BARODA UNIVERSITY, VADODARA

**Credit: (4-0-2) - 5**

Discrete Mathematics with Python

# SYLLABUS

**Prerequisite:** The students are required to have a reasonable mastery over Algebra, Logic.

# What is Discrete Mathematics?

Discrete Mathematics is a branch of mathematics involving discrete elements that uses algebra and arithmetic. It is increasingly being applied in the practical fields of mathematics and computer science. It is a very good tool for improving reasoning and problem-solving capabilities. This course explains the fundamental concepts of Sets, Relations and Functions, Mathematical Logic, Counting Theory, Probability, Mathematical Induction and Recurrence Relations, Graph Theory, Trees and Boolean Algebra.

# What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python has several built in libraries which can be used to understand the concept of Sets, Algebraic structures and Matrix representation of Graphs.

# Course Overview

The purpose of this course is to understand and use (abstract) discrete structures that are backbones of computer science. In particular, this class is meant to introduce logic, proofs, sets, relations, functions, counting, and probability, with an emphasis on applications in computer science.

Students can also determine whether a mathematical argument is logically correct, study the relationship between finite sets, count the number of ways to arrange objects in a certain pattern and analyze processes that involve a finite number of steps.

Throughout the teaching of various topics in this subject, teachers discuss how a statement, a function or a complete Python program can help visualize and concretize that topic. It is a fond hope and desire of the designers of this course that many students will directly be helped by this and they will be better computer scientists/engineers having learned Discrete Mathematics with Python. Actually the Python language is separately taught in its full glory under a separate subject title.

Learning how to think mathematically is far more important than knowing how to do all the computations. Consequently, the principal objective of this course is to help you develop the analytic skills you need to learn mathematics. To achieve this goal, we will show you the motivation behind the ideas, explain the results, and dissect why some solution methods work while others do not.

This course will introduce you to the wonderful world of Python programming. You will learn about the essential elements of programming and how to construct basic Python programs for solving the examples in Discrete Mathematics. The topic becomes more understandable if the student is able to write a computer program in python and relate it with the manual solution. This will also develop the logical ability of the student. You will learn expressions, variables, functions, logic, and conditionals, which are used to solve the problems for most of the topic. You will also learn how to use Python modules, which enable you to benefit from the vast array of functionality that is already a part of the Python language. These concepts and skills will help you to begin to think like a computer programmer and to understand how to go about writing Python programs. By the end of the course, you will be able to write short Python programs that are able to accomplish real, practical tasks. This course is the foundation for understanding the python modules which are used in discrete mathematics for understanding the concepts of sets, objects, graphs, matrices which are fundamental for writing a program for several applications in real world life such as optimisation problem, finding shortest path image processing etc.

# Learning outcome

By the end of the course students shall be confident and equipped with elementary knowledge of Discrete Mathematics and how python is used for solving the problems.

* **Understand** the basic principles of sets and operations in sets and apply counting principles to determine probabilities, domain and range of a function, identify one-to- one functions, perform the composition of functions and apply the properties of functions to application problems.
* **Write** an argument using logical notation and determine if the argument is or is not valid. To simplify and evaluate basic logic statements including compound statements, implications, inverses, converses, and contrapositives using truth tables and the properties of logic. To express a logic sentence in terms of predicates, quantifiers, and logical connectives.
* **Apply** relations and to determine their properties. Be familiar with recurrence relations.
* **Interpret** different traversal methods for trees and graphs. Model problems in Computer Science using graphs.
* **Analyze** how to write functions and pass arguments in Python.
* **Create** problems of Discrete mathematics and solve them using Python programs.

# Tutorials, video demonstration, and exercise

T1: Set theory, Function & Counting.

T2: Propositional Logic & Predicate Logic

T3: Relations, Partial ordering & Recurrence T4: Graph theory

T5: Algebraic Structure

T6: Finite State Automata: Deterministic and Non Deterministic Finite State Automata.

* <https://youtu.be/XOH1wxrBMpE>

Students will learn about sets, functions and relations.

* <https://youtu.be/mrCrjeqJv6U>

Students will learn about the basic principles of counting.

* <https://youtu.be/E40r8DWgG40>

Students will get good knowledge of graph theory.

# Textbook

* J. P. Tremblay and R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, Tata McGraw-Hill,1997
* K. H. Rosen, Discrete Mathematics and its applications, Tata McGraw-Hill, 6th Ed., 2007.

# Reference book

* S. Lipschutz and M. L. Lipson, Schaum’s Outline of Theory and Problems of Discrete Mathematics, 2nd Ed., Tata McGraw-Hill,1999.
* David Liben-Nowell, Discrete Mathematics for Computer Science, Wiley publication, July 2017.
* Eric Gossett, Discrete Mathematics with Proof, 2nd Edition,Wiley publication, July 2009.
* Al Doerr and Ken Levasseur, "Applied Discrete Structures" 2020, 3rd Edition - version 7 https://faculty.uml.edu/klevasseur/ADS2

# Required software

* <https://www.python.org/downloads/>: Python Software.

# Learning resources

* <https://swayam.gov.in/nd1_noc20_cs82/preview>

Discrete Mathematics - By Prof. SudarshanIyengar, Prof, Neeldhara – IIT Ropar, IIT Gandhinagar.

* <https://www.edx.org/course/probability-the-science-of-uncertainty-and-data> Probability – The Science of Uncertainty and data by Massachusetts Institute of Technology.
* <https://youtu.be/h_9WjWENWV8> Course on Discrete Mathematics byMIT.
* <https://www.py4e.com/book.php>
* [https://Py4e.com](https://py4e.com)
* <https://colab.research.google.com/>

**Pedagogy:**

The real value of Discrete Mathematics is in the abstract, mathematical models which help a Computer Scientist or Engineer to think clearly about complex systems, keeping away mundane details. However, study of such models requires the student to have background in more basic topics like Set theory, combinatorics, propositional logic, and basic matrix algebra. Thus the time spent in studying these basic topics will pay out well in the more abstract topics which are studied later on.

Another factor which helps the student is properly selected examples, both in terms of quality and quantity, that are relevant to the material being introduced.

With each major topic being covered is accompanied by at least one application in the computer field. One of the group of most immediate applications would be in present day complex computer software and that is where dovetailing a very convenient programming language like Python is justified. The teacher demonstrates in the class links to Python program constructs for a particular topic and the students later on develop and test the complete programs, which helps manifestation of abstract concepts in form of tangible results of a running program.

| **Name of Course – B.Tech (Discrete Mathematics with Python)** | | **Pedagogy** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Topic** | **Lecture** | **Workshop** | **ABL/ PBL** | **Tutorial** | **Practical** | **Industrial Visit** | **Expert Lecture** | **MOOC/ NPTEL** | **Any other Tool** |
| 1 | Set theory, Function & Counting | 6 | - | - | 1 | P-1, P-2, P3 | - | - |  |  |
| 2 | Propositional Logics, Predicate Logics | 7 | - | - | 2 | P-4, P-5, P-6 | - | - |  |  |
| 3 | Relations, Partial  Ordering, Recurrence Relation | 9 | - | - | 3 | P-7, P-8, P-9 | - | - |  |  |
| 4 | Graphs, Types of Graph, Subgraph, Connectedness | 9 | - | - | 4 | P-10 | - | - |  |  |
| 5 | Algebraic Structures | 4 | - | - | 5 |  | - | - |  |  |
| 6 | Deterministic Finite  Automata, Nondeterministic Finite Automata | 4 |  |  | 6 | P-11 |  |  |  |  |

# Course Outline

| Class | Topic | Readings | Lab | Tutorial |
| --- | --- | --- | --- | --- |

| 1 | Set Theory | * Definitions- Inclusion, Equality of Sets, Cartesian product, The Power Set, Some operations on Sets, Venn Diagrams * Basic Concepts of Set Theory * Some Basic Set Identities * Understanding set operations using Python | Lesson 1 | P-1 | T1 |
| --- | --- | --- | --- | --- | --- |

| 2 | Function | * Introduction & definition * Co-domain, range, image, value of a function, Examples * Surjective, injective, bijective; examples * Composition of functions, examples * Inverse function, Identity map * Condition of a function to be invertible, examples * Inverse of composite functions * Properties of Composition of functions * Defining the mathematical expressions in python | Lesson 2-3 | P-1 | T1 |
| --- | --- | --- | --- | --- | --- |
| 3 | Counting | * The Basics of Counting * The Pigeonhole Principle * Permutations and Combinations * Binomial Coefficients * Generalized Permutations and Combinations * Generating Permutations and Combinations. * Solving different problem related to Counting using python | Lesson 4-6 | P-2 | T1 |
| 4 | Proposition al Logics | * Definition, Statements & Notation * Truth Values, Connectives * Statement Formulas & Truth Tables * Well-formed Formulas * Tautologies * Equivalence of Formulas * Duality Law, * Tautological Implications, Examples * Formation of truth table using python. | Lesson 6-9 | P-3, P-4 | T2 |
| 5 | Predicate Logics | * Definition of Predicates * Statement functions, Variables, Quantifiers * Predicate Formulas, Free & Bound Variables * The Universe of Discourse, Examples * Valid Formulas & Equivalences, Examples. | Lesson 10-12 |  | T2 |
| 6 | Relations | * Definition, Binary Relation * Representation, Domain, Range, Universal Relation, Void Relation, Union, Intersection, and Complement Operations on Relations * Properties of Binary Relations in a Set: Reflexive, Symmetric, Transitive, Anti-symmetric Relations * Understanding the properties of Relations using python. | Lesson 13 | P-8 | T2 |

| 7 | Matrices & Equivalence Classes | * Definition of Matrices * Addition, subtraction, transpose * Multiplication by a scalar, multiplication of two matrices, special matrices like Identity, determinant and inverse of a 2x2 matrix. * Relation Matrix and Graph of a Relation * Partition and Covering of a Set, Equivalence Relation, Equivalence Classes * Compatibility Relation, Maximum Compatibility Block, Composite Relation * Converse of a Relation, Transitive Closure of a Relation R in Set X . * Performing the matrix operation using Python. | Lesson 14-16 | P-5 |  |
| --- | --- | --- | --- | --- | --- |
| 8 | Partial  Ordering | * Definition, Examples * Simple or Linear Ordering * Totally Ordered Set (Chain), Frequently Used Partially Ordered Relations * Representation of Partially Ordered Sets * Hasse Diagrams, Least & Greatest Members, Minimal & Maximal Members, Least Upper Bound (Supremum), Greatest Lower Bound (infimum) * Well-ordered Partially Ordered Sets (Posets) * Lattice as Posets, complete * Distributive modular and complemented lattices Boolean and pseudo Boolean lattices. * Finding the Least, Maxima element from the POSET using python. | Lesson 17-19 | P-6 | T3 |
| 9 | Recurrence Relation | * Introduction, Recursion * Recurrence Relation * Solving, Recurrence Relation * Solving the recurrence relations using python. | Lesson 20-21 | P-7,  P-9 | T3 |
| 10 | Graph | * Introduction, definition, examples * Nodes, edges, adjacent nodes, directed and undirected edge * Directed graph, undirected graph, examples * Initiating and terminating nodes, Loop (sling) * Distinct edges, Parallel edges | Lesson 22 |  | T4 |
| 11 | Types of Graph | * Multi-graph, simple graph, weighted graphs, examples * Isolated nodes * Null graph; Isomorphic graphs, examples * Degree, In-degree, out-degree, total degree of a node, examples | Lesson 23-24 |  | T4 |
| 12 | Subgraph | * Definition, examples * Converse (reversal or directional dual) of a digraph, examples * Path: Definition, Paths of a given graph, length of path, examples * Simple path (edge simple), elementary path (node simple), examples * Cycle (circuit), elementary cycle, examples | Lesson 25-26 |  | T4 |
| 13 | Connectedness | * Definition, weakly connected, strongly connected, unilaterally connected, examples * Strong, weak, and unilateral components of a graph, examples * Matrix representation of graph: Definition, Adjacency matrix, Boolean (or bit) matrix, examples * Determine number of paths of length n through Adjacency matrix, examples; Path (Reachabi blity) matrix of a graph, examples * Warshall’s algorithm to produce Path matrix, Flowchart. * Obtaining Boolean matrices, Adjacency matrix using python. | Lesson 27-30 | P-10 | T4 |
| 14 | Algebraic Structures | * Algebraic structures with one binary operation – semigroups * Monoids and Groups * congruence relation and quotient structures * Free and cyclic monoids and groups * permutation groups * substructures * Normal subgroups * Understanding Algebraic structures using Python | Lesson 31-34 |  | T5 |
| 15 | Deterministic Finite  Automata | * Alphabets, Strings, Languages * Finite Automata (FA), acceptance of strings, and languages * Deterministic Finite Automata (DFA) | Lesson 35-36 |  | T6 |
| 16 | Nondeterministic Finite Automata | * An informal view of Nondeterministic Finite Automata * Definition of Nondeterministic Finite Automata, The extended transition function * The labnguage of an NFA, Equivalence of Deterministic and Nondeterministic Finite Automata. | Lesson 37-38 | P-11 | T6 |

**List of Program using Python for Discrete Mathematics:**

1. Write a python program to define a Set, List the elements and perform the basic operations such as : Union, Intersection, Complement, Cartesian product of two sets.
2. Write a program to find the factorial of a number.
3. Write a program to find scalar multiply, matrix multiply of matrix.
4. Write a Python function that takes a list of pairs as an argument and determines if the argument constitutes a function or a general relation.
5. Write a program to denote Relation: as subsets of a cartesian product of a set, matrix representation.
6. Write a program for Propositional Logic, Logic operators, Truth Tables.
7. Write a program for Equivalence and implication.
8. Write a program for Functions: functional Python programming.
9. Write a program for Recursion: functions, generators.
10. Write a program for Graphs: matrix representation, association.
11. Write a program for Regular Expressions, Finite State Machine.