Annexure I

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSC 01 **Object Oriented Programming using Python**	**4**	**3**	**0**	**1**	Pass in Class XII	

Course Objective

This course is designed as the first course that introduces object oriented programming concepts using Python to Computer Science students. The course focuses on the development of Python programming to solve problems of different domains using object- oriented programming paradigm.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Understand the basics of programming language
2. Develop, document, and debug modular Python programs.
3. Apply suitable programming constructs and built-in data structures to solve a problem.
4. Use and apply various data objects in Python.
5. Use classes and objects in application programs and handle files.
6. apply OOPs concepts such as encapsulation, inheritance and polymorphism in writing programs.

Syllabus

Unit 1 **(4 hours)**

Introduction to Programming: Problem solving strategies; Structure of a Python program; Syntax and semantics; Executing simple programs in Python.

Unit 2 **(10 hours)**

Creating Python Programs: Identifiers and keywords; Literals, numbers, and strings; Operators; Expressions; Input/output statements; Defining functions; Control structures (conditional statements, loop control statements, break, continue and pass, exit function), default arguments.

Unit 3 **(15 hours)**

Built-in data structures: Mutable and immutable objects; Strings, built-in functions for string, string traversal, string operators and operations; Lists creation, traversal, slicing and splitting operations, passing list to a function; Tuples, sets, dictionaries and their operations.

Unit 4 **(10 hours)**

Object Oriented Programming: abstraction, encapsulation, objects, classes, methods, constructors, inheritance, polymorphism, static and dynamic binding, overloading, abstract classes, interfaces and packages.

Unit 5 **(6 hours)**

File and exception handling: File handling through libraries; Errors and exception handling.

References

1. Allen B. Downey,**Think Python: How to Think Like a Computer Scientist,** O'Reilly Media, 2024.
2. J.V. Guttag, **Introduction to Computation and Programming Using Python: With Application to Understanding Data**, MIT Press, 2016.
3. Robert Sedgewick , Kevin Wayne, Robert Dondero, **Introduction to Programming in Python: An Interdisciplinary Approach,** Addison-Wesley Professional, 2015
4. Tony Gaddis, **Starting Out with Python**, Pearson, 2021.

Additional References

(i) Brown, Martin C. *Python: The Complete Reference*, 2nd edition, McGraw Hill Education, 2018.

Suggested Practical List

1. WAP to find the roots of a quadratic equation
2. WAP to accept a number 'n' and
    a. Check if 'n' is prime
    b. Generate all prime numbers till 'n'
    c. Generate first 'n' prime numbers

    This program may be done using functions
3. WAP to create a pyramid of the character '*' and a reverse pyramid

```
    *
   ***
  *****
 *******
*********
```

```
*********
 *******
  *****
   ***
    *
```

4. WAP that accepts a character and performs the following:
    a. print whether the character is a letter or numeric digit or a special character
    b. if the character is a letter, print whether the letter is uppercase or lowercase
    c. if the character is a numeric digit, prints its name in text (e.g., if input is 9, output is NINE)
5. WAP to perform the following operations on a string
    a. Find the frequency of a character in a string.
    b. Replace a character by another character in a string.
    c. Remove the first occurrence of a character from a string.
    d. Remove all occurrences of a character from a string. WAP to swap the first n characters of two strings.
6. Write a function that accepts two strings and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string then it should return -1.
7. WAP to create a list of the cubes of only the even integers appearing in the input list (may have elements of other types also) using the following:
    a. 'for' loop
    b. list comprehension

8. WAP to read a file and

   a. Print the total number of characters, words and lines in the file.
   b. Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.
   c. Print the words in reverse order.
   d. Copy even lines of the file to a file named 'File1' and odd lines to another file named 'File2'.

9. Define a class *Employee* that stores information about employees in the company. The class should contain the following:

   (1) data members- count (to keep a record of all the objects being created for this class) and for every employee: an employee number, Name, Dept, Basic, DA and HRA.

   (2) function members:

      a. __init__ method to initialize and/or update the members. Add statements to ensure that the program is terminated if any of Basic, DA and HRA is set to a negative value.
      b. function salary, that returns salary as the sum of Basic, DA and HRA.
      c. __del__ function to decrease the number of objects created for the class
      d. __str __ function to display the details of an employee along with the salary of an employee in a proper format.

10. Write a program to define a class "2DPoint" with coordinates x and y as attributes. Create relevant methods and print the objects. Also define a method distance to calculate the distance between any two point objects.

11. Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are cubes of the keys.

12. Inherit the above class to create a "3Dpoint" with additional attribute z. Override the method defined in "2DPoint" class , to calculate distance between two points of the " 3DPoint" class.

13. Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). WAP to perform following operations:

   a. Print half the values of the tuple in one line and the other half in the next line.

b. Print another tuple whose values are even numbers in the given tuple.

c. Concatenate a tuple t2=(11,13,15) with t1.

d. Return maximum and minimum value from this tuple

14. WAP to accept a name from a user. Raise and handle appropriate exception(s) if the text entered by the user contains digits and/or special characters.

2. Forouzan, A. B., Gilberg, R. F. Computer Science: A Structured Approach using C++, 2nd edition, Cengage Learning, 2010

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

DSC02/DSC03/GE2c: COMPUTER SYSTEM ARCHITECTURE

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
Computer System Architecture	4	3	0	1	Class XII pass	NIL

Course Objectives

The objectives of this course are as follows:

• Introduces the students to the fundamental concepts of digital computer organization, design and architecture.
• Develop a basic understanding of the building blocks of the computer system and highlight how these blocks are organized together to architect a digital computer system.

Learning Outcomes

On successful completion of the course, students will be able to:

• Design Combinational Circuits using basic building blocks. Simplify these circuits using Boolean algebra and Karnaugh maps. Differentiate between combinational circuits and sequential circuits.
• Represent data in binary form, convert numeric data between different number systems and perform arithmetic operations in binary.
• Determine various stages of instruction cycle, pipelining and describe interrupts and their handling.
• Describe how CPU communicates with memory and I/O devices.

• Distinguish between different types of processors.
• Simulate the design of a basic computer using a software tool.

Syllabus

Theory

Unit – 1 **(6 hours)**

Digital Logic Circuits

Logic Gates, Truth Tables, Boolean Algebra, Digital Circuits, Combinational Circuits, Introduction to Sequential Circuits, Circuit Simplification using Karnaugh Map, Don't Care Conditions, Flip-Flops, Characteristic Tables, Excitation Table.

Unit – 2 **(9 hours)**

Digital Components (Fundamental building blocks)

Designing of combinational circuits- Half Adder, Full Adder, Decoders, Encoders, Multiplexers, Registers and Memory (RAM, ROM and their types), Arithmetic Microoperations, Binary Adder, Binary Adder-Subtractor.

Unit – 3 **(6 hours)**

Data Representation and Basic Computer Arithmetic

Number System, r and (r-1)'s Complements, data representation and arithmetic operations.

Unit – 4 **(9 hours)**

Basic Computer Organization and Design

Bus organization, Microprogrammed vs Hardwired Control, Instruction Codes, Instruction Format, Instruction Cycle, Instruction pipelining, Memory Reference, Register Reference and Input Output Instructions, Program Interrupt and Interrupt Cycle..

Unit – 5 **(6 hours)**

Processors

General register organization, Stack Organization, Addressing Modes, Overview of Reduced Instruction Set Computer (RISC), Complex Instruction Set Computer (CISC), Multicore processor and Graphics Processing Unit (GPU).

Unit – 6 **(9 hours)**

Memory and Input-Output Organization

Memory hierarchy (main, cache and auxiliary memory), Input-Output Interface, Modes of Transfer: Programmed I/O, Interrupt initiated I/O, Direct memory access.

Essential Readings

• David A. Patterson and John L. Hennessy. "Computer Organization and Design: The Hardware/Software interface", 5th edition, Elsevier, 2012.

• Mano, M. Computer System Architecture, 3rd edition, Pearson Education, 1993.

Suggestive Readings

• Mano, M. Digital Design, Pearson Education Asia, 1995.

• Null, L., & Lobur, J. The Essentials of Computer Organization and Architecture. 5th edition, (Reprint) Jones and Bartlett Learning, 2018.

• Stallings, W. Computer Organization and Architecture Designing for Performance 8th edition, Prentice Hall of India, 2010

Practicals (30 hours)

(Use Simulator – CPU Sim 3.6.9 or any higher version for the implementation)

1. Create a machine based on the following architecture
2. Create a Fetch routine of the instruction cycle.
3. Write an assembly program to simulate ADD operation on two user-entered numbers.
4. Write an assembly program to simulate SUBTRACT operation on two user-entered numbers.
5. Write an assembly program to simulate the following logical operations on two user-entered numbers.
   AND, OR, NOT, XOR, NOR, NAND
6. Write an assembly program for simulating following memory-reference instructions.
   ADD
   LDA
   STA

BUN
ISZ

7. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution:
   CLA
   CMA
   CME
   HLT
8. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution
   INC
   SPA
   SNA
   SZE

9. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution:
   CIR
   CIL \
10. Write an assembly program that reads in integers and adds them together; until a negative non-zero number is read in. Then it outputs the sum (not including the last number).
11. Write an assembly program that reads in integers and adds them together; until zero is read in. Then it outputs the sum.

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

DSC07/DSC02/GE4a: DATA STRUCTURES

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

	Credits	Credit distribution of the course		

DISCIPLINE SPECIFIC CORE COURSE – 3: MATHEMATICS FOR COMPUTING

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course				Eligibility criteria	Pre-requisite c the course (if any)
		Lecture	Tutorial	Practical/ Practice			
Mathematics for Computing	4	3	0	1		Class XII pass	NIL

Learning Objectives

The Learning Objectives of this course are as follows:

- Introduces the students to the fundamental concepts and topics of linear algebra and vector calculus.
- To build the foundation for some of the core courses in later semesters.

Learning Outcomes

This course will enable the students to:

- Perform operations on matrices and sparse matrices.
- Compute the determinant, rank and eigenvalues of a matrix.
- Perform diagonalization.
- Perform operations on vectors, the dot product and cross product.
- Represent vectors geometrically and calculate the gradient, divergence, curl.
- Apply linear algebra and vector calculus to solve problems in sub-disciplines of computer science.

SYLLABUS OF DSC – 3

Theory

Unit – 1 **(6 hours)**

Introduction to Matrix Algebra

Echelon form of a Matrix, Rank of a Matrix, Determinant and Inverse of a matrix, Solution of System of Homogeneous & Non-Homogeneous Equations: Gauss elimination and Solution of System of Homogeneous Equations: Gauss Jordan Method.

Unit – 2 **(21 hours)**

Vector Space and Linear Transformation

Vector Space, Sub-spaces, Linear Combinations, Linear Span, Convex Sets, Linear Independence/Dependence, Basis & Dimension, Linear transformation on finite dimensional vector spaces, Inner Product Space, Schwarz Inequality, Orthonormal Basis, Gram-Schmidt Orthogonalization Process.

Unit – 3 **(9 hours)**

EigenValue and EigenVector

Characteristic Polynomial, Cayley Hamilton Theorem, Eigen Value and Eigen Vector of a matrix, Eigenspaces, Diagonalization, Positive Definite Matrices, Applications to Markov Matrices.

Unit – 4 **(9 hours)**

Vector Calculus

Vector Algebra, Laws of Vector Algebra, Dot Product, Cross Product, Vector and Scalar Fields, Ordinary Derivative of Vectors, Space Curves, Partial Derivatives, Del Operator, Gradient of a Scalar Field, Directional Derivative, Gradient of Matrices, Divergence of a Vector Field, Laplacian Operator, Curl of a Vector Field.

Practical **(30 hours)**

List of Practicals:

1. Create and transform vectors and matrices (the transpose vector (matrix) conjugate transpose of a vector (matrix))
2. Generate the matrix into echelon form and find its rank.
3. Find cofactors, determinant, adjoint and inverse of a matrix.
4. Solve a system of Homogeneous and non-homogeneous equations using Gauss elimination method.
5. Solve a system of Homogeneous equations using the Gauss Jordan method.
6. Generate basis of column space, null space, row space and left null space of a matrix space.
7. Check the linear dependence of vectors. Generate a linear combination of given vectors of Rn/ matrices of the same size and find the transition matrix of given matrix space.
8. Find the orthonormal basis of a given vector space using the Gram-Schmidt orthogonalization process.
9. Check the diagonalizable property of matrices and find the corresponding eigenvalue and verify the Cayley- Hamilton theorem.
10. Application of Linear algebra: Coding and decoding of messages using nonsingular matrices.
    eg code “Linear Algebra is fun” and then decode it.
11. Compute Gradient of a scalar field.
12. Compute Divergence of a vector field.
13. Compute Curl of a vector field.

Essential Reading

- Strang Gilbert. Introduction to Linear Algebra, 5th Edition, Wellesley-Cambridge Press, 2021.
- Kreyszig Erwin. Advanced Engineering Mathematics, 10th Edition, Wiley, 2015.
- Strang Gilbert. Linear Algebra and Learning from Data, 1st Edition, Wellesley-Cambridge Press, 2019.
- Jain R. K., Iyengar S.R. K. Advanced Engineering Mathematics, 5th Edition, Narosa, 2016.

Suggestive Reading

- Deisenroth, Marc Peter, Faisal A. Aldo and Ong Cheng Soon. Mathematics for Machine Learning, 1st Edition, Cambridge University Press, 2020.
- (Lipschutz Seymour and Lipson Marc. Schaum's Outline of Linear Algebra, 6th Edition, McGraw Hill, 2017.

DSC04/DSC01/GE1b: PROGRAMMING USING C++

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre- requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
Programming using C++	4	3	0	1	Class XII pass	NIL

Course Objectives:

This course is designed to introduce programming concepts using C++ to students. The course aims to develop structured as well as object-oriented programming skills using C++ programming language. The course also aims to achieve competence amongst its students to develop correct and efficient C++ programs to solve problems spanning multiple domains.

Learning outcomes

On successful completion of the course, students will be able to:

- Write simple programs using built-in data types of C++.
- Implement arrays and user defined functions in C++.
- Write programs using dynamic memory allocation, handling external files, interrupts and exceptions.
- Solve problems spanning multiple domains using suitable programming constructs in C++.
- Solve problems spanning multiple domains using oriented programming concepts in C++.

Syllabus

Unit-1 **(3 hours)**

Introduction to C++: Overview of Procedural and Object-Oriented Programming, Using main() function, Header Files, Compiling and Executing Simple Programs in C++.

Unit-2 **(12 hours)**

Programming Fundamentals: Data types, Variables, Operators, Expressions, Arrays, Keywords, Decision-making constructs, Iteration, Type Casting, Input-output statements, Functions, Command Line Arguments/Parameters

Unit-3 **(15 hours)**

Object Oriented Programming: Concepts of Abstraction, Encapsulation. Creating Classes and objects, Modifiers and Access Control, Constructors, Destructors, Implementation of Inheritance and Polymorphism, Template functions and classes

Unit-4 **(9 hours)**

Pointers and References: Static and dynamic memory allocation, Pointer and Reference Variables, Implementing Runtime polymorphism using pointers and references

Unit-5 **(6 hours)**

Exception and File Handling: Using try, catch, throw, throws and finally; Nested try, creating user defined exceptions, File I/O Basics, File Operations

Practical **(30 hours)**

1. Write a program to compute the sum of the first n terms of the following series: The number of terms n is to be taken from the user through the command line. If the command line argument is not found then prompt the user to enter the value of n.
2. Write a program to remove the duplicates from an array.
3. Write a program that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.
4. Write a menu driven program to perform string manipulation (without using inbuilt string functions):
   - **a.** Show address of each character in string
   - **b.** Concatenate two strings.
   - **c.** Compare two strings
   - **d.** Calculate length of the string (use pointers)
   - **e.** Convert all lowercase characters to uppercase
   - **f.** Reverse the string
   - **g.** Insert a string in another string at a user specified position
5. Write a program to merge two ordered arrays to get a single ordered array.

6. Write a program to search a given element in a set of N numbers using Binary search
   a. with recursion
   b. without recursion.
7. Write a program to calculate GCD of two numbers
   a. with recursion
   b. without recursion.
8. Create a Matrix class. Write a menu-driven program to perform following Matrix operations (exceptions should be thrown by the functions if matrices passed to them are incompatible and handled by the main() function):
   a. Sum
   b. Product
   c. Transpose
9. Define a class Person having name as a data member. Inherit two classes Student and Employee from Person. Student has additional attributes as course, marks and year and Employee has department and salary. Write display() method in all the three classes to display the corresponding attributes. Provide the necessary methods to show runtime polymorphism.
10. Create a Triangle class. Add exception handling statements to ensure the following conditions: all sides are greater than 0 and sum of any two sides are greater than the third side. The class should also have overloaded functions for calculating the area of a right angled triangle as well as using Heron's formula to calculate the area of any type of triangle.
11. Create a class Student containing fields for Roll No., Name, Class, Year and Total Marks. Write a program to store 5 objects of Student class in a file. Retrieve these records from the file and display them.
12. Copy the contents of one text file to another file, after removing all whitespaces.

Essential/recommended readings

1. Stephen Prata, C++ Primer Plus, 6th Edition, Pearson India, 2015.
2. E Balaguruswamy, Object Oriented Programming with C++, 8th edition, McGraw- Hill Education, 2020. 3. D.S. Malik, C++ Programming: From Problem Analysis to Program Design, 6th edition, Cengage Learning, 2013.

Suggestive Readings

1. Schildt, H. C++: The Complete Reference, 4th edition, McGraw Hill, 2003

2. Forouzan, A. B., Gilberg, R. F. Computer Science: A Structured Approach using C++, 2nd edition, Cengage Learning, 2010

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

DSC02/DSC03/GE2c: COMPUTER SYSTEM ARCHITECTURE

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
Computer System Architecture	4	3	0	1	Class XII pass	NIL

Course Objectives

The objectives of this course are as follows:

• Introduces the students to the fundamental concepts of digital computer organization, design and architecture.
• Develop a basic understanding of the building blocks of the computer system and highlight how these blocks are organized together to architect a digital computer system.

Learning Outcomes

On successful completion of the course, students will be able to:

• Design Combinational Circuits using basic building blocks. Simplify these circuits using Boolean algebra and Karnaugh maps. Differentiate between combinational circuits and sequential circuits.
• Represent data in binary form, convert numeric data between different number systems and perform arithmetic operations in binary.
• Determine various stages of instruction cycle, pipelining and describe interrupts and their handling.
• Describe how CPU communicates with memory and I/O devices.

Annexure I

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSC 01 **Object Oriented Programming using Python**	**4**	**3**	**0**	**1**	Pass in Class XII	

Course Objective

This course is designed as the first course that introduces object oriented programming concepts using Python to Computer Science students. The course focuses on the development of Python programming to solve problems of different domains using object- oriented programming paradigm.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Understand the basics of programming language
2. Develop, document, and debug modular Python programs.
3. Apply suitable programming constructs and built-in data structures to solve a problem.
4. Use and apply various data objects in Python.
5. Use classes and objects in application programs and handle files.
6. apply OOPs concepts such as encapsulation, inheritance and polymorphism in writing programs.

Syllabus

Unit 1 **(4 hours)**

Introduction to Programming: Problem solving strategies; Structure of a Python program; Syntax and semantics; Executing simple programs in Python.

Unit 2 **(10 hours)**

Creating Python Programs: Identifiers and keywords; Literals, numbers, and strings; Operators; Expressions; Input/output statements; Defining functions; Control structures (conditional statements, loop control statements, break, continue and pass, exit function), default arguments.

Unit 3 **(15 hours)**

Built-in data structures: Mutable and immutable objects; Strings, built-in functions for string, string traversal, string operators and operations; Lists creation, traversal, slicing and splitting operations, passing list to a function; Tuples, sets, dictionaries and their operations.

Unit 4 **(10 hours)**

Object Oriented Programming: abstraction, encapsulation, objects, classes, methods, constructors, inheritance, polymorphism, static and dynamic binding, overloading, abstract classes, interfaces and packages.

Unit 5 **(6 hours)**

File and exception handling: File handling through libraries; Errors and exception handling.

References

1. Allen B. Downey,**Think Python: How to Think Like a Computer Scientist,** O'Reilly Media, 2024.
2. J.V. Guttag, **Introduction to Computation and Programming Using Python: With Application to Understanding Data**, MIT Press, 2016.
3. Robert Sedgewick , Kevin Wayne, Robert Dondero, **Introduction to Programming in Python: An Interdisciplinary Approach,** Addison-Wesley Professional, 2015
4. Tony Gaddis, **Starting Out with Python**, Pearson, 2021.

Additional References

(i) Brown, Martin C. *Python: The Complete Reference*, 2nd edition, McGraw Hill Education, 2018.

Suggested Practical List

1. WAP to find the roots of a quadratic equation
2. WAP to accept a number 'n' and
    a. Check if 'n' is prime
    b. Generate all prime numbers till 'n'
    c. Generate first 'n' prime numbers

    This program may be done using functions
3. WAP to create a pyramid of the character '*' and a reverse pyramid

```
    *
   ***
  *****
 *******
*********
```

```
*********
 *******
  *****
   ***
    *
```

4. WAP that accepts a character and performs the following:
    a. print whether the character is a letter or numeric digit or a special character
    b. if the character is a letter, print whether the letter is uppercase or lowercase
    c. if the character is a numeric digit, prints its name in text (e.g., if input is 9, output is NINE)
5. WAP to perform the following operations on a string
    a. Find the frequency of a character in a string.
    b. Replace a character by another character in a string.
    c. Remove the first occurrence of a character from a string.
    d. Remove all occurrences of a character from a string. WAP to swap the first n characters of two strings.
6. Write a function that accepts two strings and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string then it should return -1.
7. WAP to create a list of the cubes of only the even integers appearing in the input list (may have elements of other types also) using the following:
    a. 'for' loop
    b. list comprehension

8. WAP to read a file and

   a. Print the total number of characters, words and lines in the file.
   b. Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.
   c. Print the words in reverse order.
   d. Copy even lines of the file to a file named 'File1' and odd lines to another file named 'File2'.

9. Define a class *Employee* that stores information about employees in the company. The class should contain the following:

   (1) data members- count (to keep a record of all the objects being created for this class) and for every employee: an employee number, Name, Dept, Basic, DA and HRA.

   (2) function members:

      a. __init__ method to initialize and/or update the members. Add statements to ensure that the program is terminated if any of Basic, DA and HRA is set to a negative value.
      b. function salary, that returns salary as the sum of Basic, DA and HRA.
      c. __del__ function to decrease the number of objects created for the class
      d. __str __ function to display the details of an employee along with the salary of an employee in a proper format.

10. Write a program to define a class "2DPoint" with coordinates x and y as attributes. Create relevant methods and print the objects. Also define a method distance to calculate the distance between any two point objects.

11. Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are cubes of the keys.

12. Inherit the above class to create a "3Dpoint" with additional attribute z. Override the method defined in "2DPoint" class , to calculate distance between two points of the " 3DPoint" class.

13. Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). WAP to perform following operations:

   a. Print half the values of the tuple in one line and the other half in the next line.

b. Print another tuple whose values are even numbers in the given tuple.

c. Concatenate a tuple t2=(11,13,15) with t1.

d. Return maximum and minimum value from this tuple

14. WAP to accept a name from a user. Raise and handle appropriate exception(s) if the text entered by the user contains digits and/or special characters.