**100K L Deemed to be University Department of Computer Science & Engineering**

**Course Handout for Y20 Batch**

A.Y.2020-21, Even Semester

|  |  |
| --- | --- |
| **Course Title** | : Object Oriented Programming System |
| **Course Code** | : 20AD1202 |
| **L-T-P-S Structure** | : 3-0-2-3 |
| **Credits** | :4.75 |
| **Pre-requisite** | : NIL |
| **Course Coordinator** | : MR. V.UDAY KUMAR |

**Course Instructors** : MR.V.UDAY KUMAR

**Course Objective:** The objective of the course is to equip the student with problem solving skills using Object Oriented Programming language – Python and details about the essential ingredients of the programming language and its fundamentals with a rich set of examples.

**Course Rationale:** The course takes an imperative view of problem-solving using Python programming language. This necessitates a firm foundation on the principles of Object-Oriented Programming (OOP). Student is professionally trained in OOP principles. The students are made to write Python programs on their own for sets of both mathematical and other engineering problems after exposing them to the different constructs of Python language namely abstract classes, Interfaces, packages and multithreading. Finally, the student is acquainted with basic knowledge of the collection framework.

## Course Outcomes:

|  |  |  |  |
| --- | --- | --- | --- |
| **CO#** | **Course Outcome** | **PO** | **BTL** |
| CO1 | Apply the concepts of Basic Data types, Operators, Decision and Looping Control Statements, Strings. | PO3, PO5 | 3 |
| CO2 | Apply the concepts of Lists, Tuples, Dictionaries. Functions, Modules, Class, Object, OOPS principles. | PO3, PO5 | 3 |
| CO3 | |  |  |  |  | | --- | --- | --- | --- | | Apply Concepts of OOP principles, classes and objects, Call by value vs. Call by reference, recursion, and Nested classes. |  | | PO1,  PO2/PSO2 | |  | |  | | | PO3, PO5 | 3 |
| CO4 | Apply Concepts of Files, Interfaces, Packages, Threads. | PO3, PO5 | 3 |
| CO5 | Apply object-oriented programming concepts to write programs and Analyses requirements and design to implement lab-based project with SDLC in a group of students. | PO7, PO9, PO9  An ability to function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings. | 4 |

**Course Outcome Indicators (COIs):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CO No.** | **Highest**  **BTL** | **COI-1**  **(BTL-1,2)** | **COI-2**  **(BTL-3)** | **COI-4**  **(BTL-3)** |
| **CO-1** | 3 | Understanding the datatypes, | Illustrate the concepts of looping |  |
| **CO-2** | 3 | Understanding  Lists, Tuples,  Dictionaries. | Implement  Modules, Class, Object, OOPS principles. |  |
| **CO-3** | 3 | Understand Concepts of OOP principles, classes and objects, Call by value vs. Call by reference, | Solve problems using  recursion, and Nested classes. |  |
| **CO-4** | 3 | Understand the need for Files, Interfaces | Apply exception handling, Packages, Threads. |  |
| **CO-5** | 4 | The students will be able to develop real world projects in python | | |

**Program Objectives & Program Specific Objectives (POs/PSOs)**

**Program Objectives**

|  |  |
| --- | --- |
| **PO3** | An ability to design solutions for complex engineering problems and system component or processes that meet the specified needs considering public health & safety and cultural, societal & environment. |
| **PO5** | Ability to create, select and apply appropriate techniques, resources, and modern engineering activities, with an understanding of the limitations. |

**PO7** Ability to demonstrate the knowledge of engineering solutions, contemporary

issues understanding their impacts on societal and environmental contexts,

leading towards sustainable development.

**PO9** An ability to function effectively as an individual, and as a member or leader

in diverse teams and in multi-disciplinary settings.

**PO10** Ability to communicate effectively oral, written reports and graphicalforms

on complex engineering activities.

**Program Specific Objectives**

|  |  |
| --- | --- |
| **PSO1** | An ability to design and develop software projects, as well as to analyze and test user requirements. |
| **PSO2** | Working knowledge on emerging software tools and technologies. |

**Syllabus:**

Python interpreter and interactive mode; values and types: int, float, Boolean, string, and list, variables, expressions, statements, tuple assignment, precedence of operators, comments .Conditionals: Boolean values and operators, conditional (if), alternative (if-else), chained conditional (if-elif-else); Iteration: state, while, for, break, continue, pass;. Fruitful functions: functions, function definition and use, flow of execution, parameters and arguments return values, parameters, local and global scope, recursion; Strings: string slices, immutability, string functions and methods, string module; Classes, Object: command line arguments, Class, object, methods, constructors, oop’s principles. Introduction to OOPs: **Introduction to OOP’s Principles**, Difference between Methods and constructors, Polymorphism: Static and Dynamic Polymorphism (Overloading, Overriding), Inheritance – Super classes- sub classes –Protected members – constructors in sub classes, Super keyword, Encapsulation: Data Encapsulation and Data Abstraction, Object as argument and return value: Call by value vs. Call by reference, recursion, and Nested classes. Files, Interfaces, Packages, and Threads: Introduction To files: Create, Read Write, Append, Delete Interfaces – defining an interface, implementing interface, differences between classes and interfaces and extending interfaces, Packages, importing packages, differences between classes and interfaces, Implementing & Applying interface. Exception Exceptions – exception hierarchy – throwing and catching exceptions – built-in exceptions, creating own exceptions fundamentals, Threads: Difference between multi-threading and Multitasking, Different Thread objects.

Text Books:

* Guido van Rossum and Fred L. Drake Jr, “An Introduction to Python – Revised and updated for Python 3.2, Network Theory Ltd., 2011.
* Timothy A. Budd, “An Introduction to Object-Oriented Programming”, 3/e, Pearson, 2008.

**Reference Books**

1. Charles Dierbach, “Introduction to Computer Science using Python: A Computational Problem-Solving Focus, Wiley India Edition, 2013.
2. Robert Sedgewick, Kevin Wayne, Robert Dondero, “Introduction to Programming in Python: An Inter-disciplinary Approach, Pearson India Education Services Pvt. Ltd., 2016.

## Course Delivery Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Session. No.** | **CO** | **COI** | **Topic (s)** | **Book No [CH No]**  **[Page No]** | **Teaching- Learning**  **Methods** | **Evaluation Components** |
| 1 | 1 | 1 | Introducing the course handout.  Python interpreter and interactive mode; | TBOOK[1],CH1,  Pageno-22 | Lecturing, Discussion | SIE-1/SEE |
| 2 |  | 1 | Data types  Conditionals: Boolean values and operators, operator precedence and comments  variables, expressions, statements | TBOOK[1],CH2,  Pageno-39-63 | Lecturing, Discussion | SIE-1/SEE |
| 3 |  | 1 | conditional (if), alternative (if-else), chained conditional (if-elif-else); | TBOOK[1],CH3,  Pageno-89 | Lecturing, Discussion | SIE-1/SEE |
| 4 |  | 2 | Iteration: state, while, for  break, pass | TBOOK[1],CH3,  Pageno-97 | Lecturing, Discussion | SIE-1/SEE |
| 5 |  | 1 | List | TBOOK[1],CH4,  Pageno-130 | Lecturing, Discussion | SIE-1/SEE |
| 6 |  | 2 | List, Set | TBOOK[1],CH4,  Pageno-130 | Lecturing,  Discussion | SIE-1/SEE |
| 7 |  | 2 | Tuple ,Dictionary | TBOOK[1],CH4,  CH9,Pageno-131,338 | Lecturing, Discussion | SIE-1/SEE |
| 8 |  | 1 | Comparison between Lists, Tuple,Set, Dictionary | TBOOK[1],CH4,  CH9,Pageno-131,338 | Lecturing, Discussion | SIE-1/SEE |
| 9 |  | 1 | Functions, Strings | TBOOK[1],CH5,  Pageno-169 | Lecturing, Discussion | SIE-1/SEE |
| 10 | 2 | 1 | Introduction to OOP principles command line arguments | TBOOK[1],CH10,  Pageno-384 | Lecturing,  Discussion | SIE-1/SEE |
| *11* |  | 1 | Class, object, methods, | TBOOK[1],CH10,  Pageno-384 | Lecturing, Discussion | SIE-1/SEE |
| *12* |  | 1 | Access specifiers –public ,private Constructors, Difference between Methods and constructors, | TBOOK[1],CH10,  Pageno-395 | Lecturing, Discussion | SIE-1/SEE |
| *13* | 3 | 1 | Method overloading | TBOOK[1],CH10,  Pageno-411 | Lecturing, Discussion | SIE-1/SEE |
| 14 |  | 2 | Inheritance and its types | TBOOK[1],CH10,  Pageno-401 | Lecturing,  Discussion | SIE-1/SEE |
| 15 |  | 2 | Inheritance – Super classes- sub classes –Protected members | TBOOK[1],CH10,  Pageno-406 | Lecturing, Discussion | SIE-1/SEE |
| 16 |  | 2 | Polymorphism: Static Polymorphism , Dynamic Polymorphism (Overriding), | TBOOK[1],CH10,  Pageno-412 | Lecturing, Discussion | SIE-1/SEE |
| 17 |  | 2 | Revision |  | Lecturing,  Discussion | SIE-1/SEE |
| 18 |  | 1 | Object as argument and return value | TBOOK[1],CH10,  Pageno-415 | Lecturing, Discussion | SIE-2/SEE |
| 19 |  | 1 | Recursion | TBOOK[1],CH11,  Pageno-461 | Lecturing, Discussion | SIE-2/SEE |
| 20 |  | 2 | Nested classes | TBOOK[1],CH11,  Pageno-475 | Lecturing, Discussion | SIE-2/SEE |
| 21 |  | 2 | Introduction To files: Create, Read Write, | TBOOK[1],CH08,  Pageno-290 | Lecturing, Discussion | SIE-2/SEE |
| *22* |  | 2 | Append, Delete from file | TBOOK[1],CH08,  Pageno-297 | Lecturing, Discussion | SIE-2/SEE |
| *23* |  | 2 | Interfaces – defining an interface, implementing interface,  Differences between classes and interfaces | TBOOK[1],CH11,  Pageno-432 | Lecturing, Discussion | SIE-2/SEE |
| *24* |  | 2 | Extending interfaces | TBOOK[1],CH11,  Pageno-439 | Lecturing, Discussion | SIE-2/SEE |
| *25* |  | 2 | Packages | TBOOK[1],CH11,  Pageno-441 | Lecturing, Discussion | SIE-2/SEE |
| *26* |  | 2 | Importing packages | TBOOK[1],CH11,  Pageno-441 | Lecturing, Discussion | SIE-2/SEE |
| *27* | 4 | 1 | Exceptions – exception hierarchy | TBOOK[1],CH12,  Pageno-445 | Lecturing, Discussion | SIE-2/SEE |
| *28* |  | 1 | Raising exceptions | TBOOK[1],CH12,  Pageno-450 | Lecturing, Discussion | SIE-2/SEE |
| *29* |  | 2 | Built-in exceptions | TBOOK[1],CH12,  Pageno-453 | Lecturing, Discussion | SIE-2/SEE |
| *30* |  | 2 | Creating own exceptions | TBOOK[1],CH12,  Pageno-456 | Lecturing, Discussion | SIE-2/SEE |
| *31* |  | 2 | Multithreading introduction | TBOOK[1],CH12,  Pageno-458 | Lecturing, Discussion | SIE-2/SEE |
| *32* |  | 2 | Different Thread objects. | TBOOK[1],CH12,  Pageno-454 | Lecturing,  Discussion | SIE-2/SEE |
| *33* |  | 2 | Difference Between Multithreading vs Multiprocessing in Python | TBOOK[1],CH12,  Pageno-458 | Lecturing, Discussion | SIE-2/SEE |
| *34* |  | 2 | Revision | TBOOK[1],CH12,  Pageno-460 | Lecturing,  Discussion | SIE-2/SEE |
| *35* |  | 2 | Revision | TBOOK[1],CH12,  Pageno-462 | Lecturing,  Discussion | SIE-2/SEE |
| *36* |  | 2 | Revision | TBOOK[1],CH12,  Pageno-462 | Lecturing,  Discussion | SIE-2/SEE |

**Session Wise Teaching/Learning Plan:**

**Session: 01**

**Session Outcome:** At the end of this session, Students will be able to understand

1. Importance of Python Programming
2. Basic Concepts on Python

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 15 | Course handout discussion:  Regarding Syllabus, Textbooks, References, MOOC’s, Evaluation Pattern, Division of course  Competencies. |  |  |  |
| 5 | Introduction/Recap of Procedural Oriented  Programming | **2** | Lecturing,  Discussion |  |
| 20 | Python interpreter and interactive mode | **2** | Lecturing,  Discussion |  |
| 10 | Basic Example programs | **2** | Lecturing,  Discussion |  |

## SESSION NUMBER: 02

**Session Outcome:** At the end of this session, Students will be able to understand

1. Students will be able to understand the basic data types and understand them

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Attendance |  |  |  |
| 5 | Write a program to print “WELCOME TO PYTHON WORLD” | **2** | Lecturing |  |
| 20 | Introduction to Basic Data types ,integer, float ,complex numbers, string, Boolean, special characters, | **2** | Lecturing |  |
| 20 | Explanation about Conditional operator and other operators with precedence  Explanation of various basic expression statements | 2 | Lecturing |  |

## SESSION NUMBER: 03

**Session Outcome:** At the end of this session,

1. Students will be able to understand the conditional (if), alternative (if-else),
2. Students will be able to understand chained conditional (if-elif-else)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explanation of syntax of  1.simple-if  2.alternate(if-else)  3. chained conditional (if-elif-else) | **2** | Lecture/Discussion |  |
| 20 | (Financial: compare costs) Suppose you shop for rice and find it in two different sized  packages. You would like to write a program to compare the costs of the  packages. The program prompts the user to enter the weight and price of each  package and then displays the one with the better price. Here is a sample run:  Enter weight and price for package 1: 50, 24.59  Enter weight and price for package 2: 25, 11.99  Package 1 has the better price. | 3 | Lecture/Discussion |  |
| 10 | (Check a number) Write a program that prompts the user to enter an integer and  checks whether the number is divisible by both 5 and 6, divisible by 5 or 6, or just  one of them (but not both). Here is a sample run:  Enter an integer:10  Is 10 divisible by 5 and 6? False  Is 10 divisible by 5 or 6? True  Is 10 divisible by 5 or 6, but not both? True | 2 | Lecture/Discussion |  |

## SESSION NUMBER: 04

**Session Outcome:** At the end of this session,

1. Students will be able to understand Iteration: state, while, for break, pass

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/ Learning Methodology** | **Active Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain syntax for  1.while  2.for | **3** | Lecturing,  Discussion |  |
| 10 | **Program-1:**  The program generates the magic number and prompts the user to enter a guess continuously .For each guess, the program determines whether the user’s number is correct, too high, or too low .When the guess is correct, the program exits the loop .Note that **guess** is initialized to **-1**. This is to avoid initializing it to a value between **0** and **100**, because that could be the number to be guessed. | **3** | Lecturing, Discussion |  |
| 10 | **Practise-1:**  Can you always convert any while loop into a for loop? Convert the following while loop into a for loop.  i = 1  sum = 0  while sum < 10000:  sum = sum + i  i += 1  **Program-2:**  write a program to find the smallest factor other than 1 for an integer n (assume n >= 2).(use break)  **Program-3:**  (Find the two highest scores) Write a program that prompts the user to enter the number of students and each student’s score, and displays the highest and second highest scores. | **3** |  | Group Discussion |
| 5 | Conclusion & summary |  |  |  |
| Homework:  1. (*Financial application: compute future tuition*) Suppose that the tuition for a university is $10,000 this year and increases 5% every year. Write a program that computes the tuition in ten years and the total cost of four years’ worth of tuition starting ten years from now. | | | | |

## SESSION NUMBER: 05

**Session Outcome:** At the end of this session,

1. Students will understand the usage List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/ Learning**  **Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain the usage and syntax of list  **Program-1:**  Create a list of numbers and display all those numbers which are the above the average of all the numbers in the list | **3** | Lecturing,  Discussion |  |
| 30 | Explain the use of index operator and list slicing  Explain the use of in/not operators in list.  Explain the list functions:  append(x: object): None  count(x: object): int  extend(l: list): None  index(x: object): int  insert(index: int, x: object): None  pop(i): object  remove(x: object): None  reverse(): None  sort(): None | **3** |  | Think/Pair/Sha re |
| 5 | Conclusion & summary |  |  |  |

## SESSION NUMBER: 06

**Session Outcome:** At the end of this session,

1. Students will understand List and Set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/**  **Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain the usage of split() method with an example | 3 | Lecture |  |
| 10 | Given **lst = [30, 1, 2, 1, 0]**, what is the return value of each of the following  statements?  lst.index(**1**)  lst.count(**1**)  len(lst)  max(lst)  min(lst)  sum(lst) | 2 | Lecture |  |
| 10 | Create a set and explain  1.add()  2.remove()  3.len()  4.min()  5.max()  6.sum() | **3** | Lecture |  |
| 10 | Explain the methods  1.issubset()  2.issuperset()  3.union()  4.intersection()  5.symmetricdifference() |
| 5 | Conclusion & Summary |  |  |  |
| Homework:  1.(Print distinct numbers) Write a program that reads in numbers separated by a space in one line and displays distinct numbers (i.e., if a number appears multiple times, it is displayed only once). (Hint: Read all the numbers and store them in **list1**. Create a new list **list2**. Add a number in **list1** to **list2**. If the number is already in the list, ignore it.) Here is the sample run of the program:  Enter ten numbers: 1 2 3 2 1 6 3 4 5 2  The distinct numbers are: 1 2 3 6 4 5 | | | | |

## SESSION NUMBER:07

## Session Outcome:

## 1. Student will be able to understand the usage of tuple

## 2. Student will be able to understand the usage of Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/ Learning**  **Methodology** | **Active Learning**  **Methods** |
| **5** | Recap/Introduction |  |  |  |
| **5** | Explain the difference between tuple and a list |  |  |  |
| **10** | Explain the following functions for tuple  ( len , max, min, sum) |  |  |  |
| **10** | Explain how to  1.Combine two tuples  2.slice a tuple  3.Duplicate a tuple  4.Make a list out of tuple |  |  |  |
| **15** | Explain how to Create a Dictionary and Adding, Modifying, and Retrieve Values from Dictionary  Explain get() method |  |  |  |

## SESSION NUMBER: 08

**Session Outcome:**

1. Students will understand and compare between List,Tuple and Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/ Learning**  **Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Write a program that reads a list of scores and then assigns grades  based on the following scheme:  The grade is A if score is best >= 10.  The grade is B if score is best >=20.  The grade is C if score is best >=30.  The grade is D if score is best >=40.  The grade is F otherwise.  Here is a sample run:  Enter scores: 40 55 70 58  Student 0 score is 40 and grade is C  Student 1 score is 55 and grade is B  Student 2 score is 70 and grade is A  Student 3 score is 58 and grade is B | **3** | Lecture |  |
| 10 | Show the printout of the following code:  t = (**1**, **2**, **3**, **7**, **9**, **0**, **5**)  print(t)  print(t[**0**])  print(t[**1**: **3**])  print(t[**-1**])  print(t[ : **-1**])  print(t[**1** : **-1**]) | **3** | Lecture |  |
| 20 | Suppose a dictionary named students is {"john":3, "peter":2}. What do the following statements do?  (a) students["susan"] = 5  (b) students["peter"] = 5  (c) students["peter"] += 5  (d) del students["peter"]  Suppose a dictionary named students is {"john":3, "peter":2}. What do the following statements do?  (a) print(len(students))  (b) print(students.keys())  (c) print(students.values())  (d) print(students.items()) | **3** | Lecture |  |
| 5 | Conclusion & Summary |  |  |  |

## SESSION NUMBER: 09

**Session Outcome:** At the end of this session,

1. Students will understand Strings: string slices, immutability, string functions and methods, string module;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/**  **Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain the concept String slicing and introduce basic string functions(capitalize(),count(),find(),split(),isupper(),join()) | 3 | Lecture |  |
| 20 | Define a function that returns the maximum of two values  Predict the output:  Identify and correct the errors in the following program:  def function1(n, m):  function2(3.4)  def function2(n):  if n > 0:  return 1  elif n == 0:  return 0  elif n < 0:  return –1  function1(2, 3) |  |  |  |
| (Display an integer reversed) Write the following function to display an integer in reverse order:  def reverse(number):  For example, reverse(3456) displays 6543. Write a test program that prompts the user to enter an integer and displays its reversal. | 3 |  | Think/Pair/Sha  re |
| 5 | (*Display characters*) Write a function that prints characters using the following header:  **def** printChars(ch1, ch2, numberPerLine):  This function prints the characters between **ch1** and **ch2** with the specified numbers per line. Write a test program that prints ten characters per line from **1** to **Z**. | **3** | Lecture |  |
| 5 | Conclusion & Summary |  |  |  |
| Homework:  1. (*Twin primes*) *Twin primes* are a pair of prime numbers that differ by 2. For example, 3 and 5, 5 and 7, and 11 and 13 are twin primes. Write a program to find all twin primes less than 1,000. Display the output as follows:  (3, 5)  (5, 7)... | | | | |

## SESSION NUMBER: 10

**Session Outcome:** At the end of this session,

1.Student will be able to understand OOP principles

2.Student will be able to use Command line arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/**  **Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 15 | Differences between Procedure oriented & Object-Oriented Programming. | 3 | Lecture |  |
| 20 | Explain various ways of dealing Python command line arguments  Using sys.argv  Using getopt module  Using argparse module | 3 | Lecture |  |
| 10 | Write a Program to read set of integers through command line arguments, use for each loop to print the data and sum of all values |  |  |  |

## SESSION NUMBER: 11

**Session Outcome:** At the end of this session,

1. Students will understand the need for Class, object, methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/ Learning**  **Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Write a Cuboid class with 3 static variables length, breadth and height of type double, and a static method volume (), access them using main () method within the same class. | 2 | Lecture |  |
| 20 | Explain the need of a class as a template (Encapsulate data and methods)  Syntax – Define a class,  Explain self keyword. | 3 | Lecture |  |
| 10 | Modularize the above Cuboid class  Write a Cuboid class with 3 static variables length, breadth and height of type double, and a static method volume (), access them using main () method within another class Demo. |  |  |  |
| 5 | Conclusion & Summary |  |  |  |

## SESSION NUMBER: 12

**Session Outcome:** At the end of this session,

1. Students will understand the need for Constructors, Difference between Methods and constructors,.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/**  **Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 15 | Explain the usage of access modifiers – private and public | 3 | Lecture |  |
| 15 | Explain different types of constructors  1.Parameterized Constructor  2.Non-parameterized Constructor | 3 | Lecture |  |
| 10 | Predict the output of the following  What problem arises in running the following program? How do you fix it?  class A:  def \_\_init\_\_(self, i):  self.i = i  def main():  a = A()  print(a.i)  main() # Call the main function  What is wrong with the following programs?  class A:  # Construct an object of the class  def A(self):  radius = 3  class A:  # Construct an object of the class  def \_\_init\_\_(self):  radius = 3  def setRadius(radius):lf.radius = radius |  |  |  |
| 5 | Conclusion and Summary |  |  |  |

## SESSION NUMBER: 13

**Session Outcomes:** At the end of this session,

1. To be able to understand Method overloading

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active Learning**  **Methods** |
| 5 | Recap |  |  |  |
| 15 | Explain method overloading with an example  1. Built-in function.  2. User defined function. | **3** | Lecturing, Discussion |  |
| 15 | Create the class area with a private members length,breadth.Overload the method find\_area().Display its area using str method . | **3** | Lecturing, Discussion |  |
| 10 | Write a program to overload the area method by using importing a package math | **3** | Lecturing, Discussion |  |
| 5 | Conclusion and Summary |  |  |  |

## SESSION NUMBER: 14

**Session Outcomes:** At the end of this session,

**1.** Inheritance – Super classes- sub classes Protected members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learni ng Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction**:** |  |  |  |
| 10 | Need for Inheritance with an example and Syntax, Terminology | **2** | Lecturing |  |
| 10 | Types of Inheritance  1. Simple Inheritance  Explain the using GeometricShape Class with attributes borderColor (String), filled (Boolean type). This is inherited by Rectangle Class with length and width as attributes. Add mutators, accessors and str methods |  |  | Brainstorming |
| 10 | 2. Multilevel Inheritance  Enhance the above design where Cuboid class with height field inherits the Rectangle class. |  |  | Brainstorming |
| 10 | 3. Hierarchy Inheritance  Enhance the design of Simple Inheritance where Circle class with radius field also inherits from GeometricShape. |  |  |  |
| 5 | Conclusion and Summary |  |  |  |

## SESSION NUMBER: 15

**Session Outcomes:** At the end of this session,

**1.** Understand member access protected

2. Apply Super method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap |  |  |  |
| 10 | Explain protected and usage | **3** |  |  |
| 20 | Enhance the designs using protected keyword | **3** |  | Think/Pair/S  hare |
| 10 | Explain how to access super class fields and methods using super () | **3** |  |  |
| 5 | Conclusion and Summary |  |  |  |

## SESSION NUMBER: 16

**Session Outcomes:** At the end of this session,

1. Students will be able to understand Polymorphism: Static Polymorphism

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active Learning Methods** |
| 5 | Recap |  |  |  |
| 10 | Explain about overriding | **3** | Lecture |  |
| 15 | Show the printout of the following program:  **class** A:  **def** \_\_init\_\_(self, i = **0**):  self.i = i  **def** m1(self):  self.i += **1**  **class** B(A):  **def** \_\_init\_\_(self, j = **0**):  super().\_\_init\_\_(**3**)  self.j = j  **def** m1(self):  self.i += **1**  **def** main():  b = B()  b.m1()  print(b.i)  print(b.j)  main() # Call the main function | **3** |  | 15 min paper |
| 15 | Explain the polymorphism with isinstance() | **3** | Lecture |  |
|  | Program-1:  (The Location class) Design a class named Location for locating a maximal value and its location in a two-dimensional list. The class contains the public data fields row, column, and maxValue that store the maximal value and its indexes in a two-dimensional list, with row and column as int types and maxValue as a float type.  Write the following method that returns the location of the largest element in a two dimensional list.  def Location locateLargest(a):  The return value is an instance of Location. Write a test program that prompts the user to enter a two-dimensional list and displays the location of the largest element in the list. Here is a sample run:  Enter the number of rows and columns in the list:  Enter row 0:  Enter row 1:  Enter row 2:  The location of the largest element is 45 at (1, 2) |  |  |  |
| 5 | Conclusion and Summary |  |  |  |

## SESSION NUMBER: 17

**Session Outcomes:** Revision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap |  |  |  |
| 20 | (*The* **Stock** *class*) Design a class named **Stock** to represent a company’s stock  that contains:  ■ A private string data field named **symbol** for the stock’s symbol.  ■ A private string data field named **name** for the stock’s name.  ■ A private float data field named **previousClosingPrice** that stores the stock price for the previous day.  ■ A private float data field named **currentPrice** that stores the stock price for the current time.  ■ A constructor that creates a stock with the specified symbol, name, previous  price, and current price.  ■ A get method for returning the stock name.  ■ A get method for returning the stock symbol.  ■ Get and set methods for getting/setting the stock’s previous price.  ■ Get and set methods for getting/setting the stock’s current price.  ■ A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice**.  Draw the UML diagram for the class, and then implement the class. Write a test program that creates a **Stock** object with the stock symbol INTC, the name Intel  Corporation, the previous closing price of **20.5**, and the new current price of **20.35**, and display the price-change percentage. |  | Lecture |  |
| 10 | What problem arises in running the following program? How do you fix it?  **class** A:  **def** \_\_init\_\_(self, i):  self.i = i  **def** main():  a = A()  print(a.i)  main() # Call the main function |  | Lecture |  |
| 10 | Show the output of the following program:  **class** Count:  **def** \_\_init\_\_(self, count = **0**):  self.count = count  **def** main():  c = Count()  times = **0**  **for** i **in** range(**100**):  increment(c, times)  print(**"count is"**, c.count)  print(**"times is"**, times)  **def** increment(c, times):  c.count += **1**  times += **1**  main() # Call the main function |  | Lecture |  |
| 5 | Recap |  | Lecture |  |

## SESSION NUMBER: 18

**Session Outcomes:** At the end of this session, Students will be able

**1.** To be able to Object as argument and return value

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap |  |  |  |
| 15 | Passing an object as an argument to a method  Practical session 1:  Create a class Test with equals () method which compares two objects for equality and returns the result i.e., either true or false (Note: equals () methods should take an object as an argument) | **3** | Lecture |  |
| 15 | Returning object from a method  Practical session 1:  Create a class Test with incrByTen () method which returns an object after incrementing the value by 10 than the value in the invoking object. |  |  |  |
| 25 | Call by value and Call by reference | **3** | Discussion |  |
| 5 | Conclusion & Summary |  |  |  |

## Session: 19

**Session Outcome:** At the end of this session, Students will be able to understand and apply

1. Recursion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain the difference between Iteration and Recursion | **3** | Lecture |  |
| 10 | (Compute greatest common divisor using recursion).The **gcd(m, n)** can also be defined recursively as follows:  ■ If **m % n** is **0**, **gcd(m, n)** is **n**.  ■ Otherwise, **gcd(m, n)** is **gcd(n, m % n)**.  Write a recursive function to find the GCD. Write a test program that prompts the user to enter two integers and displays their GCD. | **3** | Lecture |  |
| 10 | (*Occurrences of a specified character in a string*) Write a recursive function that finds the number of occurrences of a specified letter in a string using the following  function header.  **def** count(s, a):  For example, **count("Welcome", 'e')** returns **2**. Write a test program that prompts the user to enter a string and a character, and displays the number of occurrences for the character in the string. | **3** | Lecture |  |
| 10 | (*Find the largest number in a list*) Write a recursive function that returns the largest integer in a list. Write a test program that prompts the user to enter a list of integers and displays the largest element. | **3** | Lecture |  |
| 5 | Conclusion & Summary |  |  |  |

## Session: 20

**Session Outcome:** At the end of this session, Students will be able to

1. Understand the concept of inner class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 5 | Explain the concept of inner class and accessibilty | **3** | Lecture |  |
| 15 | Identify the output:  # create a Color class  class Color:      # constructor method    def \_\_init\_\_(self):      # object attributes      self.name = 'Green'      self.lg = self.Lightgreen()      def show(self):      print("Name:", self.name)      # create Lightgreen class    class Lightgreen:       def \_\_init\_\_(self):          self.name = 'Light Green'          self.code = '024avc'         def display(self):          print("Name:", self.name)          print("Code:", self.code)    # create Color class object  outer = Color()    # method calling  outer.show()    # create a Lightgreen  # inner class object  g = outer.lg    # inner class method calling  g.display() | **3** |  | Think/Pair/  Share |
| 10 | Create a human class with head class as its inner class and access the members of inner class through the reference of outer class | **3** | Lecture |  |
| 10 | Enhance the concept with multiple inner classes | **3** | Lecture |  |
| 5 | Conclusion & Summary |  |  |  |

## Session: 21

**Session Outcome:** At the end of this session, Students will be able to

1. Understand how to Create, Read from files and Write into files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active**  **Learning Methods** |
| 5 | Recap/Introduction |  |  |  |
| 10 | Explain the method open() with modes r, w, a, rb, wb | **3** |  | Think/Pair/ Share |
| 15 | Write a program that removes all the occurrences of a specified string from a text file. Your program should prompt the user to enter a filename and a string to be removed. Here is a sample run:  Enter a filename: test.txt  Enter the string to be removed: morning | **3** | Lecture |  |
| 15 | (*Replace text*) Write a program that replaces text in a file. Your program should prompt the user to enter a filename, an old string, and a new string. Here is a sample run:  Enter a filename: test.txt  Enter the old string to be replaced: morning  Enter the new string to replace the old string:  afternoonDone |
| 5 | Conclusion & Summary |  |  |  |

## Session: 22

**Session Outcome:** At the end of this session, Students will be able to

1. Understand Append and Delete form files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **BTL** | **Teaching/Learning Methodology** | **Active Learning**  **Methods** |
| 5 | Recap/Introduction |  |  |  |
| 20 | |  | | --- | | (*Write/read data*) Write a program that writes 100 integers created randomly into a file. Integers are separated by a space in the file. Read the data back from the file and display the sorted data. Your program should prompt the user to enter a filename. If the file already exists, do not override it. Here is a sample run:  Enter a filename: test.txt  The file already exists | |  |  |  |
| 20 | (*Count words*) Write a program that counts the number of words in PresidentAbraham Lincoln’s Gettysburg Address from <http://cs.armstrong.edu/liang/data> Lincoln.txt. |  |  |  |
| 5 | Conclusion & Summary |  |  |  |

**SESSION PLAN 23:**

**Session Outcome:** At the end of this session students will be able to understand

1. Interfaces – defining an interface, implementing interface,

2. Differences between classes and interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Minutes** | **Topic: Abstract Classes** | **BTL** | **Teaching Methodology** | **ALM** |
| 05 | **Recap / Introduction:** | 1 |  |  |
| 20 | |  | | --- | | Understand the concept of Differences between classes & interfaces |   . | 2 | Chalk & talk |  |
| 20 | Understand the concept of interface  And implement the concept of interfaces. | 3 | Discussion | Brain Storming |
| 05 | **Summary and Conclusion** |  |  |  |

**SESSION PLAN 24:**

**Session Outcome:** At the end of this session on Extending interfaces,, Students will be able:

* 1. To solve exercise problems on interfaces.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Minutes** | **Topic** | **BTL** | **Teaching Methodology** | **ALM** |
| 05 | Recap / Introduction |  |  |  |
| 30 | Explain about   1. Informal interfaces 2. Formal interfaces | 3 | Discussion | Case study |
| 10 | Write a program for computing sum of two integers and floats using interface? | 3 | Discussion |  |
| 05 | Conclusion & Summary |  |  |  |

**SESSION PLAN 25:**

**Session Outcome:** At the end of this session on **Packages**, Students will be able:

1. Student should be able to create packages, add classes to packages, access packages and use a package

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Minutes** | **Topic** | **BTL** | **Teaching Methodology** | **ALM** |
| 05 | Recap / Introduction |  |  |  |
| 20 | Sub-topic 1(Lecture):  Explain how to create user defined packages to group different classes and/or interfaces  Explain how to add more classes to a package | 1 | Chalk & talk |  |
| 10 | :Create a package named animal. Add Mammal class to animal package. Mammal class must contain eat(), travel() methods. | 3 | Discussion |  |
| 10 | Create college, staff and cse packages such that cse is present in staff and staff is present college. | 3 | Discussion |  |
| 05 | Conclusion & Summary |  |  |  |

**SESSION PLAN: 26**

**Session Outcome:** At the end of this session on **Importing predefined packages and usage,**

1. **Students will be able to use the methods available in predefined packages available**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Minutes** | **Topic** | **BTL** | **Teaching Methodology** | **ALM** |
| 05 | Recap / Introduction |  |  |  |
| 20 | **Sub-topic 1(Lecture):**  Understand the concept of packages, Types of predefined packages, | 1 | Chalk & talk |  |
| 10 | **Sub-topic 2(Lecture):**  Importing available predefined packages | 1 | Chalk & talk |  |
| 10 | Print the details of the student like name, age, marks percentage, branch by using package | 3 | Discussion |  |
| 05 | Conclusion & Summary |  |  |  |
|  | **Home Assignments:**   1. **Practice session-2:**   program to demonstrate accessing of members when corresponding classes are imported and not imported Write about predefined packages with an example. |  |  |  |

**SESSION NUMBER: 27**

# **Session Outcome:** At the end of this session students will be able to learn Exception Exceptions – exception hierarchy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **B T L** | **Teaching/Lear ning Methodology** | **Active Learnin g**  **Methods** |
| 5 | Recap/Introduction**:** | 1 |  |  |
| 20 | Explain about The Standard Exception Hierarchy | **3** | Lecturing,  Discussion |  |
| 10 | Explain about  1.try  2.except | **3** | Lecturing, Discussion |  |
| 10 | Suppose that **statement2** causes an exception in the following statement:  **try**:  statement1  statement3  **except** Exception1:  # Handle exception  **except** Exception2:  # Handle exception  **except** Exception3:  # Handle exception  **finally**:  statement4  statement5  Answer the following questions:  ■ Will **statement5** be executed if the exception is not caught?  ■ If the exception is of type **Exception3**, will **statement4** be executed and will **statement5** be executed? |
| 5 | Conclusion & Summary |  |  |  |

**SESSION NUMBER: 28**

# **Session Outcome:** At the end of this session students will be able to learn about Throwing and catching exceptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **B T L** | **Teaching/Lear ning Methodology** | **Active Learnin**  **g Methods** |
| 5 | Recap/Introduction**:** | 1 |  |  |
| 20 | Explain about  1.else  2.finally  3.raise | **2** | Lecturing, Discussion |  |
| 10 | Explain about multiple excepts | **2** | Lecturing, Discussion |  |
| 10 | Explain how to process exceptions using exception objects |
| 5 | Conclusion & Summary |  |  |  |

**SESSION NUMBER: 29**

**Session Outcome: To implement Exception handling using** Built-in exceptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Minutes** | **Topic** | **BT L** | **Teaching–**  **Learning Method** | **Active**  **Learning Methods** |
| 5 | Recap / Introduction: |  | Lecturing, Discussion |  |
| 20 | Explain about  1. ArithmeticError  2. SyntaxError  3. ZeroDivisionError  4. IndentationError | 2 |  |
| 3 | Quiz |
| 20 | What is displayed when the following program is run?  def main():  try:  f()  print("After the function call")  except ZeroDivisionError:  print("Divided by zero!")  except:  print("Exception")  def f():  print(1 / 0)  main() # Call the main function | 3 | Surprise Test |
|  | What is displayed when the following program is run?  **try**:  lst = **10** \* [**0**]  x = lst[**10**]  print(**"Done "**)  **except** IndexError:  print(**"Index out of bound"**)  **else**:  print(**"Nothing is wrong"**)  **finally**:  print(**"Finally we are here"**)  print(**"Continue"**) |  |  |  |
| **5** | Conclusion |  |  |

**SESSION NUMBER: 30**

**Session Outcome:** At the end of this session students will be able create own exceptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (Min)** | **Topic** | **B T L** | **Teaching/Lear ning Methodology** | **Active Learnin g**  **Methods** |
| 5 | Recap/Introduction: | 1 |  |  |
| 10 | Create a custom exception by extending a base exception | 3 | Lecturing, Discussion |  |
| 10 | What is wrong in the following code?  **try**:  # Some code here  ...  **except** ArithmeticError:  print(**"ArithmeticError"**)  **except** ZeroDivisionError:  print(**"ZeroDivisionError"**)  print(**"Continue"**) | 3 | Lecturing, Discussion |  |
| 10 | Create custom exception class that extends **RuntimeError** .The initializer simply invokes the superclass’s initializer and sets the radius in the data field It contains **setRadius(radius)** method in the **Circle** class to raise an **InvalidRadiusException** if the radius is negative. When creating a **Circle** object with a negative radius an **InvalidRadiusException** is raised. The exception is caught in the **except** clause |
| 10 | Suppose that **statement2** causes an exception in the following **try**-**except**  block:  **try**:  statement1  statement3  **except** Exception1:  # Handle exception 1  **except** Exception2:  # Handle exception 2  statement4  Answer the following questions:  ■ Will **statement3** be executed?  ■ If the exception is not caught, will **statement4** be executed?  ■ If the exception is caught in the **except** block, will **statement4** be executed? |  |  |  |
| 5 | Conclusion & Summary |  |  |  |

**Session Number: 31**

**Session Outcome: 1. Understand the Multithreading**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time (min)** | **Topic** | **BTL** | **Teac hing – Lear ning Meth od** | **Activ e Lear ning Meth ods** | |
| 05 | Recap the previous topic |  |  | |  |
| 15 | Explain about the Threading Module1. threading.activeCount ()2. threading.currentThread ()3. threading.enumerate () | 3 | Chalk and talk | |  |
| 15 | Explain the methods of the thread class  1.run()  2.start()  3.join([time])  4.isAlive()  5.getName()  6.setName() | 3 | Chalk and talk | | Group Discu ssion |
| 10 | Create a thread using Thread module and override\_\_init\_\_(self,args) and override run(self,args[])to implement which thread to start | 3 | talk | |  |
| 05 | Conclusion |  |  | |  |

**SESSION NUMBER: 32**

**Session Outcome: Understanding concept of Synchronizing threads**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time(min)** | **Topic** | **BTL** | **Teaching – Learning Method** | **Active Learning Methods** |
| 05 | Recap the previous topic |  |  |  |
| 10 | Explain about threading.enumerate () | 2 | Chalk and talk |  |
| 15 | Explain the methods of the thread class  1.isAlive()  2.getName()  3.setName() | 2 | Chalk and talk |  |
| 15 | Explain about  1.lock()  2.release() | 2 | Chalk and Talk |  |
| 05 | Conclusion |  |  |  |

## Session - 33

**Session Outcome: 1. Difference Between Multithreading vs Multiprocessing in Python**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time(min)** | **Topic** | **BTL** | **Teaching – Learning Method** | **ALM** |
| 05 | Recap the previous topic |  |  |  |
| 20 | Explain about  1.program  2.process  3.thread  4.Multithreading  5.Multiprocessing | 3 | Chalk and talk |  |
| 20 | Explain about  1.I/O bound  2.CPU bound | 3 | Chalk and talk | seminar |
| 05 | Conclusion |  |  |  |

## Session Number: 34

**Session Outcome:** Student will be able to understand str class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **Teaching** | **Active** |
| **Time(min)** | **Topic** | **BTL** | **–**  **Learning** | **Learning Methods** |
|  |  |  | **Method** |  |
| 5 | Recap the previous topic |  |  |  |
| 20 | Explain **str** class methods  1.isalnum(): bool  2.isalpha(): bool  3.isdigit(): bool  4,isidentifier(): bool  5.islower(): bool  6.isupper(): bool  7.isspace(): bool | 2 | Chalk and talk, PPT | Role play |
| 20 | Explain about  endswith(s1: str): bool  startswith(s1: str): bool  find(s1): int  rfind(s1): int  count(substring): int | 2 | Chalk and talk, PPT |  |
| 5 | Conclusion |  |  |  |

## Session Number: 35

**Session Outcome:** Student will be able to understand and apply Multi dimensional Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **Teaching** | **Active** |
| **Time(min)** | **Topic** | **BTL** | **–**  **Learning** | **Learning Methods** |
|  |  |  | **Method** |  |
| 5 | Recap the previous topic |  |  |  |
| 15 | Explain A two-dimensional list with example  1. Initializing Lists with Input Values  2. Initializing Lists with Random Values  3. Printing Lists  4. Summing All Elements  5. Summing Elements by Column | 2 | Chalk and talk, PPT | Role play |
| 10 | Explain A two-dimensional list with example  1. Summing All Elements  2. Summing Elements by Column |  |  |  |
| 10 | What is the output of the following code?  matrix = []  matrix.append(**3** \* [**1**])  matrix.append(**3** \* [**1**])  matrix.append(**3** \* [**1**])  matrix[**0**][**0**] = **2**  print(matrix)  What is the output of the following code?  matrix = []  matrix.append([**1**, **2**, **3**])  matrix.append([**4**, **5**])  matrix.append([**6**, **7**, **8**, **9**])  print(matrix) | 2 | Chalk and talk, PPT |  |
| 5 | Conclusion |  |  |  |

## Session Number: 36

**Session Outcome:** Student will be able to implement Case Studies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **Teaching** | **Active** |
| **Time(min)** | **Topic** | **BTL** | **–Learning** | **Learning Methods** |
|  |  |  | **Method** |  |
| 5 | Recap |  |  |  |
| 15 | **Counting Keywords**  For each word in a Python source file, we need to determine whether the word is a keyword.To handle this efficiently, store all the keywords in a set and use the **in** operator to test if aword is in the keyword set. | 2 | Chalk and talk, PPT | Role play |
| 15 | **Occurrences of Words**  The program determines whether each word is already a key in the dictionary. If not,the program adds a dictionary item with the word as the key and the value **1**. Otherwise, the program increases the value for the word (key) by **1** in the dictionary. Assume the words are case-insensitive (for example, **Good** is treated the same as **good**). The program displays the ten most frequently used words in the file in decreasing order of their count. | 2 | Chalk and talk, PPT |  |
| 10 | (*The* **Location** *class*) Design a class named **Location** for locating a maximal value and its location in a two-dimensional list. The class contains the public data  fields **row**, **column**, and **maxValue** that store the maximal value and its indexes in a two-dimensional list, with **row** and **column** as **int** types and **maxValue** as a float type.  Write the following method that returns the location of the largest element in a two-dimensional list.  **def** Location locateLargest(a):  The return value is an instance of **Location**. Write a test program that prompts the user to enter a two-dimensional list and displays the location of the largest element  in the list. Here is a sample run:  Enter the number of rows and columns in the list: 3, 4  Enter row 0: 23.5 35 2 10  Enter row 1: 4.5 3 45 3.5  Enter row 2: 35 44 5.5 12.6  The location of the largest element is 45 at (1, 2) |  |  |  |
| 5 | Conclusion and Summary |  |  |  |

## EVALUATION PLAN:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Evaluation Type** | **Evaluation Component** | **Weightage/Marks** | | **Assessment Dates** | **Duration (Hours)** | **CO1** | **CO2** | **CO3** | **CO4** | **CO5** |
| **Blooms Taxonomy Level** | | | | | | **1** | **2** | **3** | **4** |  |
| **Semester In Summative Evaluation Total = 34 %** | **Semester-in Exam-I** | Weightage | 12% | Semester-in Exam-I Dates | 2 | 6% | 6% |  |  |  |
| Max Marks | 50M | 25 | 25 |  |  |  |
| **Semester-in Exam –II** | Weightage | 12 % | Semester-In Exam-II Dates | 2 |  |  | 6% | 6% |  |
| Max Marks | 50M |  |  | 25 | 25 |  |
| **Lab Semester-in Exam** | Weightage | 5 % | Lab Semester-In Exam Dates | 1 ½ |  |  |  |  | 5% |
| Max Marks | 50M |  |  |  |  | 50 |
| **Skill Semester-in Exam** | Weightage | 5 % | Skill Semester-In Exam Dates | 1 ½ | 1% | 2% | 2% |  |  |
| Max Marks | 50M | 10 | 20 | 20 |  |  |
| **Formative Evaluation Total = 26 %** |  | | | | | | | | | |
| **ALMs/Surprise Quiz** | Weightage | 8% | Continuous Evaluation | | 2% | 2% | 2% | 2% |  |
| Max Marks | 50M | 12.5 | 12.5 | 12.5 | 12.5 |  |
| **Home Assignment** | Weightage | 4% | Continuous Evaluation | | 1 % | 1 % | 1% | 1 % |  |
| Max Marks | 50M | 12.5 | 12.5 | 12.5 | 12.5 |  |
| **Lab Continuous Evaluation** | Weightage | 6% | Continuous evaluation | |  |  |  |  | 6% |
| Max Marks | 50M |  |  |  |  | 50 |
| **Skill Continuous Evaluation** | Weightage | 4% | Continuous evaluation | | 1% | 1% | 1% | 1% |  |
| Max Marks | 50M | 12.5 | 12.5 | 12.5 | 12.5 |  |
| **Ratings on Global Platforms** | Weightage | 4% | Continuous evaluation | | | | | | |
| Max Marks | 50M |
| **Semester End Summative Evaluation Total = 40 %** | **SE Lab Exam** | Weightage | 8% | Lab External  Dates | 2 |  |  |  |  | 8% |
| Max Marks | 50M |  |  |  |  | 50 |
| **SE Skill Exam** | Weightage | 8% | Skill External  Dates | 2 |  |  |  |  | 8% |
| Max Marks | 50M |  |  |  |  | 50 |
| **Semester End Exam** | Weightage | 24% | Semester End Exam Dates | 3 | 6% | 6% | 6% | 6% |  |
| Max Marks | 100M | 25 | 25 | 25 | 25 |  |

**ATTENDANCE POLICY**

Every student is expected to be responsible for regularity of his/her attendance in classrooms and laboratories, to appear in scheduled tests and examinations and fulfil all other tasks assigned to him/her in every course. In every course, student has to maintain a minimum of 85% attendance to be eligible for appearing in Semester end examination of the course, for cases of medical issues and other unavoidable circumstances the students will be condoned if their attendance is between 75% to 85% in every course, subjected to submission of medical certificates, medical case file and other needful documental proof to the concerned departments/Dean.

## DETENTION POLICY

In any course, a student must maintain a minimum of 85% attendance to be eligible for appearing to the Semester End Examination, failing to fulfil these conditions will deem such student to have been detained in that course.

## COURSE TEAM MEMBERS, CHAMBER CONSULTATION HOURS AND CHAMBER VENUE DETAILS:

Each instructor will specify his / her chamber consultation hours during which the student can contact him / her in his / her chamber for consultation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Name of Faculty** | **Chamber Consultation Day (s)** | **Chamber Consultation Timings for each**  **day** | **Chamber Consultation Room No:** | **Signature of Course faculty** |
| 1 | V.UDAY KUMAR BES-1 | All working days | 2:00P.M to 3:20 P.M | F206 |  |

## GENERAL INSTRUCTIONS

Students should come prepared for classes and carry the textbook(s) or material(s) as prescribed by the Course Faculty to the class.

## NOTICES.

All notices will be communicated through the institution email.

All notices concerning the course will be displayed on the respective Notice Boards.

## Signature of COURSE COORDINATOR:

**Signature of Department Prof. In charge Academics & Vetting Team Member:**

**HEAD OF DEPARTMENT: Approval from: DEAN-ACADEMICS**

**(Sign with Office SEAL)**