COURSEPACK (Fall 2023-24)

SCHEME

The scheme is an overview of work-integrated learning opportunities and gets students out into the real world. This will give what a course entails.

<table>
<tr><td>Course Title</td><td colspan="3">Object-Oriented Programming</td><td colspan="3">Course Type</td><td colspan="4">Comprehensive</td></tr>
<tr><td>Course Code</td><td colspan="3">E2UC201C</td><td colspan="3">Class</td><td colspan="4">B.Tech (CSE) II Sem</td></tr>
<tr><td rowspan="5">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Weekly Hours</td><td colspan="3" rowspan="2">Total Number of Classes per Semester</td><td colspan="4">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>3</td><td>3</td><td colspan="4"></td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td><td rowspan="3">Theory</td><td rowspan="3">Tutorial</td><td rowspan="3">Practical</td><td rowspan="3">Self-study</td><td rowspan="3">CIE</td><td rowspan="3">SEE</td></tr>
<tr><td>Practical</td><td>1</td><td>2</td></tr>
<tr><td>Self-study</td><td>1</td><td>8</td></tr>
<tr><td>Total</td><td>5</td><td>13</td><td>30</td><td>0</td><td>30</td><td>120</td><td>50%</td><td>50%</td></tr>
<tr><td rowspan="3">Names of Course Instructors</td><td>Course Lead</td><td colspan="3">Ms. Vineeta</td><td>Course Coordinator</td><td colspan="3">Shweta Mayor Sabharwal</td></tr>
<tr><td colspan="3">Theory</td><td></td><td colspan="4">Practical</td></tr>
<tr><td colspan="3">Anubhav Kumar
Kumar Manoj
Nipendra Dwivedi
Akhilendra Kumar Khare
Amit Kumar
Rahul
Shabir Ali
Anurag Singh
Ashwini Kumar Pradhan
Dewan Imdadul Islam
Fatima Ziya
Gurpreet Singh
K. Prabu
Manmohan Singh
Neha
Nitin Shukla
Pragati Gupta
Prem Kant
S Karpaga Selvi
R. Muthuganesh
Rahul Anjana
Rajiv Chourasiya
Rakesh Sahu
S P Ramesh
Shwet Ketu
Shweta Mayor Sabharwal
Sonu Kumar Jha
Subash Harizan
Tarachand Verma
Vijaya Choudhary</td><td></td><td colspan="4">Anubhav Kumar
Kumar Manoj
Nipendra Dwivedi
Akhilendra Kumar Khare
Amit Kumar
Rahul
Shabir Ali
Anurag Singh
Ashwini Kumar Pradhan
Dewan Imdadul Islam
Fatima Ziya
Gurpreet Singh
K. Prabu
Manmohan Singh
Neha
Nitin Shukla
Pragati Gupta
Prem Kant
S Karpaga Selvi
R. Muthuganesh
Rahul Anjana
Rajiv Chourasiya
Rakesh Sahu
S P Ramesh
Shwet Ketu
Shweta Mayor Sabharwal
Sonu Kumar Jha
Subash Harizan
Tarachand Verma
Vijaya Choudhary</td></tr>
</table>

	Vineeta Vivek Kumar Sharma Vivek Sharma Yamini Singh Yogendra Kumar		Vineeta Vivek Kumar Sharma Vivek Sharma Yamini Singh Yogendra Kumar

COURSE OVERVIEW

"Object-Oriented Programming" course provides a good understanding of object-oriented concepts and implementation in Python programming. This course will improve the Python programming skills of developers who have a basic understanding of Python. You will learn Object Oriented features of Python programming which will help in providing efficient solutions for software projects.

PREREQUISITE COURSE

Prerequisite course required	Yes (√)	No

COURSE OBJECTIVE

- To teach Object-oriented Concepts and introduce programming basics using Python.
- Guide the students to build simple applications using Object-oriented Programming Concepts using Python Language.

COURSE OUTCOMES(COs)

After the completion of the course, the student will be able to:

Course Outcomes	Upon successful completion of this course, the student will be able to:
E2UC201C.1.	Understand the principles of object-oriented programming in problem-solving.
E2UC201C.2.	Apply inheritance and exception-handling techniques in problem-solving using Python
E2UC201C.3.	Apply Generators, Decorators, comprehensions, and packages involving functional programming in Python
E2UC201C.4.	Apply Tkinter to create GUI Applications.
E2UC201C.5.	Create simple software applications using Python GUI for some potential users.

BLOOM'S LEVEL OF THE COURSE OUTCOMES

Bloom's taxonomy is a set of hierarchical models used for the classification of educational learning objectives into levels of complexity and specificity. The learning domains are cognitive, affective, and psychomotor.

COMPREHENSIVE

CO No.	Bloom's Taxonomy Level(BTL)					
	Remember (**KL1**)	Understand (**KL2**)	Apply (**KL3**)	Analyze (**KL4**)	Evaluate (**KL5**)	Create (**KL6**)
E2UC201C.1	√	√				
E2UC201C.2		√	√			
E2UC201C.3		√	√			
E2UC201C.4		√	√			
E2UC201C.5		√	√	√	√	√

PROGRAM OUTCOMES (POs):

PO1	**Computing Science knowledge:** Apply the knowledge of mathematics, statistics, computing science and information science fundamentals to the solution of complex computer application problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex computing science problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and computer sciences**.**
PO3	**Design/development of solutions:** Design solutions for complex computing problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern computing science and IT tools including prediction and modeling to complex computing activities with an understanding of the limitations.
PO6	**IT specialist and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional computing science and information science practice.
PO7	**Environment and sustainability**: Understand the impact of the professional computing science solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the computing science practice.

PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the IT analyst community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the computing science and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs):

PSO1	Have the ability to work with emerging technologies in computing requisite to Industry 4.0.
PSO2	Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

COURSE ARTICULATION MATRIX

The Course articulation matrix indicates the correlation between Course Outcomes and Program Outcomes and their expected strength of mapping in three levels (low, medium, and high).

CO/PO Mapping (1 / 2 / 3 indicates strength of correlation) 3 - Strong, 2 - Medium, 1 – Low														
COs	**Programme Outcomes (POs)**													
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
E2UC201C.**1**	1	-	1	-	-	-	-	-	-	-	-	-	-	-
E2UC201C.**2**	2	2	1	1	2	-	-	-	-	-	-	-	1	-
E2UC201C.**3**	2	2	1	1	2	-	-	-	-	-	-	-	2	-
E2UC201C.**4**	2	2	1	1	2	-	-	-	-	-	-	-	3	-
E2UC201C.**5**	2	1	1	1	2	1	1	1	3	3	2	2	3	1
1-Low,2-Medium,3-High														

COURSE ASSESSMENT

The course assessment patterns are the assessment tools used both in formative and summative examinations.

Type of Course (C)	**CIE**			**Total Marks**		**Final Marks** **CIE*0.5+SEE*0.5**
	LAB[@] (Work+ Record)	MTE	Course-based Project[^]	CIE	SEE	
COMPREHENSIVE	25	50	25	100	100	100

[@]Lab Work-15 marks + Lab Record-10 marks

[^] **Typical Rubric for the Course-based project**

Type of Assessment Tools	Preliminary Project Plan	Technical Seminar	TRL-1	Viva-voce
Course-based Project Work	05	05	10	05
PPP (Preliminary Project Plan): *The preliminary project plan (PPP) provides an initial, overview of the project and all of its known parameters. It outlines the project's objectives, relevance to the program, merit, and conformity to current industry/government policy, proposed methodology, and expected outcomes. It should also include any known constraints related to the time frame (Gantt Chart), budget, etc.*				
TRL (Technology Readiness Level)-1: *Basic Research: Initial scientific research has been conducted. Principles are qualitatively postulated and observed. Focus is on new discovery rather than applications.*				

COURSE CONTENT

THEORY+ PRACTICAL

THEORY:

INTRODUCTION TO OOP:

Purpose of Object-Oriented Programming-Classes and Objects-Class Attributes -Class variable-methods and Instance variable-methods-working of an object-constructor-type of contractor-destructor-Advantages of using destructor-Pillars_of_OOPS: Encapsulation-Access-specifier: public-private-protected-Polymorphism: function and objects-Overriding (Run time polymorphism)-Overloading-Different between method overloading and overriding.

INHERITANCE AND EXCEPTION HANDLING:

Reusability of classes-Types of Inheritance: single level Inheritance-multilevel-multiple and hierarchal-Super methods-method resolution order-diamond problem-Overriding-Interfaces-Abstract class and Method-properties-Error Handling-Exception Handling-Types of Exception-custom exception

FUNCTIONAL PROGRAMMING:

Lambda function-First class, higher order, and proxy function-Iterators- Performance of Generators: importance and needs Performance of Generators-Decorators: function with parameter and arguments-Class method and static method decorators-Map, Filter, Zip, Reduce, –Regular Expressions -Comprehensions: List, tuples, set, and Dictionary- Modules and packages: import packages, built-in modules, package index, pip install, Virtual Environments.

PYTHON GUI:

Introduction to Tkinter, settling widgets in the window's interior, Numeric Widgets, Boolean Widgets, Selection Widgets, String Widgets, Date Picker, Color Picker, Container Widgets, creating GUI Application with Tkinter, button, canvas, Geometry Management, Binding Function, Working with image.

PRACTICAL: Creating a Class and Object; Class with constructor; Inheritance; Polymorphism; Multiple Inheritance; Abstract base classes; Class Methods and Static Methods; Composition; Operator Overloading; Exception Handling; Composition and aggregation; Method Overriding; Lambda functions; Loan Calculator using Tkinter; Popup Menu based Arithmetic Operations using Tkinter; Display an Image and its Transformation using Tkinter and OpenCV-Python.

LESSON PLAN FOR INTEGRATED COURSES of 3 CREDITS

FOR THEORY 15 weeks * 2 Hours = 30 Classes) (1credit = 1Lecture Hour)
FOR PRACTICAL 15 weeks * 2Hours = 30 Hours lab sessions (1 credit = 2 lab hours)

L-No	Topic for Delivery	Theory / Tutorial / Practical Plan	Skills	Competency
1	**INTRODUCTION TO OOP:** Purpose of Object-Oriented Programming-Classes and Objects-Class Attributes	Theory	Interpret the principles of the object-oriented programming paradigm.	CO1
2	Class variable-methods and Instance variable-methods-working of an object	Theory		CO1
3	Write a program to print the area of two rectangles having sides (a,b) and (c,d) respectively by creating a class named 'Rectangle' with a method named 'Area' which returns the area and length and breadth passed as parameters to its constructor.	**Practical**		CO1, CO2
4		**Practical**		
5	Constructor-type of constructor	Theory	Apply the constructors & destructors technique in program design	CO2
6	Destructor-Advantages of using destructor	Theory		CO2
7	Write a program by creating an 'Employee' class having the following methods and print the final salary. 1 - 'getInfo()' which takes the salary, number of hours of work per day of employee as parameter	**Practical**		CO1, CO2
8	2 - 'AddSal()' which adds $100 to salary of the employee if it is less than $500. 3 - 'AddWorkSal()' which adds $ to salary of employee as an input .	**Practical**		
9	Pillars_of_OOPS: Encapsulation,Inheritance,Polymorphism, Abstraction	Theory	Apply OOPs Concepts in python programs	CO2
10	Encapsulation, Access-specifier: public-private-protected	Theory		CO2

11	Any method for which implementation is unknown can be marked as Abstract method. (prefix keyword abstract). If any method in the class is marked abstract, then the class as well is marked as abstract. Moreover, Objects cant be created for an abstract class thus addressing the above issue. Hmmm.. Lets try it out. We need a method to calculate service tax on every transaction. To keep things simple, let's assume its 10% for Savings, 20% for Checking and 5% for Demat. Make the method abstract in the Account class and override in child classes.	**Practical**		
12	Create a abstract class Account with 3 private data member variables -accountNumber of type String, balance of type double, holderName of type String and methods - display() and an abstract method calculateServiceTax() with an argument of type int and return type as double; Create the class CheckingAccount which extends the class Account Create the class SavingsAccount which extends the class Account . Create the class DematAccount which extends the class Account . Use Appropriate Getters Setters for the above classes. Create a driver class named Main which creates an instance of the above mentioned classes. Service Tax must be calculated seperately for all the base class (value must be round to 2 decimal place).	**Practical**		CO1, CO2
13	Polymorphism: function and objects	Theory	Apply the concept of polymorphism using objects and functions	CO3
14	Overriding (Run time polymorphism)	Theory		CO3
15	Governing agency mandates transaction has to be notified through SMS, Email and a monthly e-statement. that any customer who performs a As expected, we define an interface Notification and three methods as specified below. Lets code this example. Create a Notification interface which has three methods notificationBySms(), notificationByEmail(), notificationByCourier(). Create two class as Icici, Create a BankFactory class getIcici() (returns object (returns object for Hdfc). Create the main class that input and output.	**Practical**		CO1, CO3
16		**Practical**		

17	Overloading-Different between method overloading and overriding	Theory	Implement the exception handling techniques to solve real world problems	CO3
18	Taking shopping cart example covering classes, objects, constructors, destructor	Theory		
19	Extending shopping cart example covering polymorphism, overloading, overriding	Theory		
20	**INHERITANCE AND EXCEPTION HANDLING:** Reusability of classes	Theory		CO3
21	Create the class HDFC which extends the class RBI. Use Appropriate Getters Setters for the above classes. Create a driver class named Main which creates an instance of the above-mentioned classes. The credit score must be calculated separately for all the classes (value must be round to 2 decimal place).	**Practical**		CO1, CO3
22		**Practical**		
23	Types of Inheritance: single level Inheritance-multilevel-mutiple and hierarchal	Theory	Implement the concept of inheritances for a problem solution	CO3
24	Super() methods	Theory		CO3
25	The Matrix class has methods for each of the following: 1 - get the number of rows 2 - get the number of columns 3 - set the elements of the matrix at given position (i,j) 4 - adding two matrices. If the matrices are not addable, "Matrices cannot be added" will be displayed.	**Practical**		CO1,CO3
26	5 - multiplying the two matrices note - print 'invalid' on invalid input, no need to print options and 2x2 matrix is suffice	**Practical**		
27	method resolution order-diamond problem	Theory	Use interface in python programs	CO3
28	Overriding-Interfaces	Theory		CO3

29	Write a program to print the area and circumfrence of a circle having radius r units by creating a class named 'circle' without any parameter in its constructor.	**Practical**		CO1, CO3
30		**Practical**		
31	Abstract class and Method-properties	Theory	Apply functional programming in python programs	CO3
32	Error Handling-Exception Handling			
33	Types of Exception-custom exception	Theory		CO3
34	One such practical scenario is contract between the Banks to perform fund / message transfer. The information has to be confidential and each bank might use its own encryption & decryption techniques. The Reserve Bank might setup an Interface giving the method names and not worrying about the implementation. 1. Create an interface BankTransfers 2. Add two methods with the following prototype -- public String encrypt(String a); -- public String decrypt(String a); 3. Create class ICICI which implements the BankTransfers Interface & implement a simple encryption technique. 4. Create class HDFC which implements the BankTransfers Interface & implement a simple encryption technique. 5. Encrypt technique followed by both banks: ICICI - add 1 with the ASCII value of the character and insert number '1' after every character. HDFC - add 1 with the ASCII value of the character in the even Index and subtract 1 with the ASCII value of the character in the odd Index .It does not encrypt the space. The reverse of both will be decrept message (i.e original text)	**Practical**		CO1, CO3
35		**Practical**		
36	Extending shopping cart example more by covering exception handling	Theory	Implement the exception handling techniques	CO3
37	**FUNCTIONAL PROGRAMMING:** Lambda function-First class, higher order and proxy function	Theory		CO4

38	Write a program to print the area and perimeter of a triangle having sides of a, b and c units by creating a class named 'Triangle' with constructor having the three sides as its parameters.	**Practical**		CO1, CO4
39		**Practical**		
40	Iterators	Theory	Apply the generator in python	CO4
41	Performance of Generators: importance and needs Performance of Generators	Theory		CO4
42	Write a program to print the area and perimeter of a triangle having sides of a, b and c units by creating a class named 'Triangle' without any parameter in its constructor.	**Practical**		CO1, CO4
43		**Practical**		
44	Decorators: function with parameter and arguments	Theory	Use decorators with class and static methods	CO4
45	Class method and static method decorators	Theory		CO4
46	Create a class called 'Matrix' containing constructor that initializes the number of rows and number of columns of a new Matrix object. The Matrix class has the following information: 1 - print number of rows of matrix 2 - print number of columns of matrix 3 - print elements of matrix in the form of 2D array.	**Practical**		CO1, CO4
47		**Practical**		
48	Map, Filter, Zip, Reduce, Regular Expressions	Theory	Develop Comprehensions in python	CO4
49	Comprehensions: List, tuples, set and Dictionary	Theory		CO4
50	Method Overriding: Create a base class called “Vehicle” with a method called “drive.” Implement two subclasses, “Car” and “Bicycle,” that inherit from “Vehicle” and override the “drive” method with their own implementations.	**Practical**		CO1, CO4
51		**Practical**		
52	Modules and packages: import packages	Theory	Apply lambda Functions in	CO4
53	built-in modules, package index, pip install,	Theory		CO4

	virtual Environments		python programs	
54	Lamda functions:	**Practical**		CO1, CO4
55	Write python code to perform the following Lamda function below: (i) Map() (ii) filter() (iii) lambda function to calculate grades for a list of scores. (iv) List Comprehensions. (v) calculates the factorial of a number using a recursive lambda function	**Practical**		
56	**PYTHON GUI:** Introduction to Tkinter, settling widgets in the window's interior	Theory	Use Tkinder in GUI application development	CO4
57	Numeric Widgets, Boolean Widgets, Selection Widgets, String Widgets	Theory		CO4
58	Loan Calculator using Tkinter: (i). Write python GUI application is developed for computing loan payments. The code consists of the following steps: 1.Design the user interface consisting of labels using Label( ), text entry boxes using Entry( ), and a button using Button( ).	**Practical**		CO1, CO4
59	2.Process the event. When the button is clicked, the program invokes a callback functions, using getMonthlyPayment( ) to obtain the user input for the interest rate, number of years, and loan amount from the text entries. The monthly and total payments are computed using computePayment( ) and displayed in the labels.	**Practical**		
60	Date Picker, Color Picker, Container Widgets	Theory	Use numeric , and Boolean widgets in GUI application	CO5
61	Buttons	Theory		CO5
62	Popup Menu based Arithmetic Operations using Tkinter: Creating a popup menu is similar to creating a regular menu. First, you create an instance of Popup menu using Menu( ),	**Practical**		CO5
63	and then you can add items to it using menu.add_command( ). Finally, you bind a widget with an event to pop up the menu. Popup menu commands are used to perform actions to be displayed in a canvas	**Practical**		

	which is created using canvas( ). The canvas widget is used to add the structured graphics to the python application. It is used to display text, images, graph and plots to the python application. The menu items use callback functions to instruct the canvas to perform actions. In code, a popup menu is displayed by clicking the right mouse button followed by binding it with the canvas using canvas.bind( ). Each item in the popup menu is selected to perform the corresponding arithmetic operations. The inputs are obtained in two textboxes and the output is displayed in one textbox.			
64	Canvas	Theory	Create GUI application using widgets	CO5
65	Geometry Management	Theory		
66	Display an Image and its Transformation using Tkinter and OpenCV-Python:	**Practical**		
67	(i).Program to read and display an RGB color image and convert it into grayscale, negative and edge images	**Practical**		
68	Binding Function	Theory	Create GUI applications with image and binding functions	CO5
69	Working with image	Theory		
70	creating GUI Application with Tkinter	Theory		
71	Revision	Theory		
72	Revision	Theory		

BIBLIOGRAPHY

- **Text Book**:
  1. The Complete Reference Python, Martin C. Brown, McGraw Hill, Fourth edition (20 March 2018)
  2. Head First Python: a Brain-Friendly Guide by Paul Barry (Author), Dawn Griffiths, Shroff/O'Reilly; Second Edition (1 January 2012).
- **Reference Books:**
  1. Python In-Depth by Ahidjo Ayeva, Kamon Ayeva, Aiman Saed, BPB Publication (1 OCT 2020).
  2. Introduction to programming using Python, Y. Daniel Liang, Pearson Education; First Edition (26 February 2017).
  3. Mastering Python, Rick Van Hatten, Packet Publishing House (29 April 2016).
  4. Starting out with Python, Tony Gaddis, Pearson, 4th edition (17 May 2017).

- **Webliography:**

1. https://docs.python.org/3/tutorial/

- **SWAYAM/NPTEL/MOOCs Certification:**

1. https://onlinecourses.swayam2.ac.in/cec22_cs20/preview
2. https://onlinecourses.nptel.ac.in/noc22_cs32/preview
3. https://onlinecourses.nptel.ac.in/noc19_cs42/preview

- Optional certifications and online platforms:

1. https://www.codechef.com/practice/python
2. https://skillsforall.com/course/python-essentials-2
3. https://www.coursera.org/learn/object-oriented-programming-and-gui-with-python

PRACTICE PROBLEMS

(Assignments) (Min 45 Problems*)

S.No	Practice Problems	KL
1	Develop a program for creating a class named Students, initializing attributes such as name, age, and grade during object instantiation.	K3
2	Write a program to establish a valid empty class named Students, devoid of any properties.	K3
3	Design a program for creating a child class Teacher that inherits properties from the parent class Staff.	K3
4	Devise a Python program to define a Vehicle class with instance attributes max_speed and mileage.	K3
5	Instantiate a Vehicle class without including any variables or methods.	K3
6	Construct a child class Bus that inherits all variables and methods from the Vehicle class created in question 3.	K3
7	Create a Python program to define a person class with attributes like name, country, and date of birth, and implement a method for determining the person's age.	K3
8	Implement a Python class named BankAccount representing a bank account, incorporating attributes such as accountNumber, name, and balance. Provide a complete code for the BankAccount class with various methods.	K3
9	Discuss the importance of access modifiers (e.g., public, private, protected) in achieving the objectives of Object-Oriented Programming (OOP) in Python.	K3
10	Explain how a destructor can be defined and utilized in a Python class.	K3
11	Explore the usage of the 'private' access modifier in Python and discuss its implications.	K3
12	Enumerate the different types of polymorphism in Object-Oriented Programming (OOP) in Python.	K3
13	Develop a Python program that overloads the operator + and > for a custom class.	K3
14	Demonstrate method overriding in Python to achieve runtime polymorphism with a code example.	K3
15	Discuss the advantages and disadvantages of compile-time polymorphism compared to runtime polymorphism in Python.	K3
16	Explain how method overriding showcases runtime polymorphism in Python.	K3

17	Create a Python code example illustrating data overloading by defining multiple attributes with the same name in a class.	K3
18	Differentiate between data overloading and data overriding in Python.	K3
19	Define data overriding and provide an example demonstrating it in Python.	K3
20	Present a Python code example showcasing data overloading using multiple constructors with different parameters.	K3
21	List and explain the exceptions in Python.	K3
22	Describe how to handle exceptions using try, except, and finally blocks in Python.	K3
23	Explain the purpose of the raise statement in Python.	K3
24	Discuss the use of the except clause without an exception type in Python.	K3
25	Highlight the differences between the except and else blocks in Python exception handling.	K3
26	Explain how to handle multiple exceptions in a single except block in Python.	K3
27	Determine if it is possible to include other statements between 'try,' 'catch,' and 'finally' blocks in Python.	K3
28	Elaborate on the concept of an unreachable catch block error in Python.	K3
29	Differentiate between errors and exceptions in Python.	K3
30	Enumerate and explain the types of exceptions in Python.	K3
31	Explain the purpose of the finally block in Python exception handling.	K3
32	Discuss the concept of custom exceptions in Python.	K3
33	Explain the purpose of the assert statement in Python.	K3
34	Differentiate between a built-in exception and a custom exception.	K3
35	Describe how to re-raise an exception in Python.	K3
36	Clarify the purpose of inheritance in Python.	K3
37	Define the role of the super() function in Python.	K3

38	Differentiate between throw, throws, and throwable in Python.	K3
39	Explain the StackOverflowError in Python.	K3
40	Explore whether it is possible to override a superclass method that throws an unchecked exception with checked exceptions in the sub-class.	K3
41	Identify the class defined as a superclass for all types of errors and exceptions in Python.	K3
42	Discuss the correct combination of try, catch, and finally blocks in Python.	K3
43	Explain the use of the printStackTrace() method in Python.	K3
44	Create a Bus child class that inherits from the Vehicle class, incorporating fare calculations with maintenance charges.	K2
45	Provide examples of checked exceptions in Python.	K3
46	Develop a Python code to create a decorator returning the square of a number.	K3
47	Compare class method and static method, and demonstrate their use with a Python code example.	K3
48	Validate functions such as append(), extend(), pop(), reverse(), remove(), sort(), and insert() of a list in Python.	K3
49	List at least five built-in modules with their functions in Python.	K3
50	Write a Python code to generate a random number between 50 to 99 and shuffle a given list1=[21,22,23,24,25].	K3
51	Describe the process of working with images in Tkinter, covering image loading, resizing, and display within a GUI.	K3
52	Explain the concepts of class method and static method decorators with a suitable example in Python.	K3
53	Differentiate between tuples and dictionaries in Python.	K3
54	Develop a detailed guide on creating a Tkinter application that includes multiple widgets, emphasizing effective organization and user interaction.	K3
55	Define a lambda function with suitable code in Python.	K3
56	Explain the purpose of the sys module in Python.	K3
57	Write a code to find the sum of a given list of numbers (list1=[5,6,7,8,9,12,15,18,21]) using the reduce function in Python.	K3

58	Discuss the key features of a Color Picker and its application in enhancing Tkinter interfaces.	K3
59	Elaborate on the role of Container Widgets, with a specific focus on the Frame widget in Tkinter.	K3
60	Demonstrate the use of regular expressions to count the number of vowels in a given string in Python.	K3

STUDENT-CENTEREDLEARNING (SELF-LEARNING TOWARDS LIFE-LONG-LEARNING)

Self-Learning (it's a typical course-based project to be carried out by a whole class in groups of four students each; they should exhibit higher level KLs)

The students, in a group, are expected to conceive an idea based on the content (objectives/outcomes) and apply the suitable knowledge to demonstrate their learning.

A list of 30-40 project statements can be offered to the students to choose or students can conceive their own ideas (teamwork), design and develop the product/process/service and implement the same.

SOME SAMPLE PROJECT (Psychomotor skills) (30 Projects*)

To enhance their skill set in the integrated course, the students are advised to execute course-based projects. Some sample projects are given below:

A) COURSE-BASED PROJECT (Psychomotor skills)

S. No.	Project Name	Description
1	Employee Payroll System	Build a system to manage employee payroll information including salary calculation, deductions, and bonuses. Include features for adding new employees, updating details, and generating pay-slips.
2	Online Quiz System	Create an online quiz platform with multiple-choice questions on various topics. Implement features like user registration, quiz-taking, and result display.
3	Recipe Management System	Design a system to manage recipes, ingredients, and meal planning. Include features such as adding new recipes, categorizing them, and generating shopping lists.
4	Movie Ticket Booking System	Develop a system for booking movie tickets online. Include features such as movie selection, seat reservation, and payment processing.
5	Social Media Analysis Tool	Create a tool to analyze social media data including trends, sentiments, and user interactions. Utilize APIs to fetch data from platforms like Twitter or Facebook.
6	Expense Tracker Application	Build an application for tracking personal expenses and budgeting. Include features for adding expenses, categorizing them, and generating reports.
7	Online Auction System	Design a system for conducting online auctions. Include features for bidding, auction management, and winner selection.
8	Music Player Application	Develop a music player application with features like playlist creation, shuffle, and repeat. Include support for various audio formats.
9	Parking Management System	Build a system for managing parking spaces in a parking lot. Implement features for vehicle entry/exit tracking, fee calculation, and slot availability.
10	Gym Management System	Design a system to manage gym memberships, schedules, and workout routines. Implement features for member registration, attendance tracking, and progress monitoring.
11	Appointment Scheduling System	Build a system for scheduling appointments for services like doctor visits, salon appointments, etc. Include features for appointment booking, reminders, and cancellations.
12	Student performance monitoring system using Python	Student performance monitoring system using Python
13	Tic-Tac-Toe Game	Develop a simple Tic-Tac-Toe game with a graphical user interface. Include features for player vs. player and player vs. computer modes.

14	Chat Application	Build a real-time chat application for communication between multiple users. Implement features like private messaging, group chats, and online status indicators.
15	Online Voting System Project Python	We have designed an Online Voting System in Python to tackle the existing problem of elections in various spaces. It is a web-based voting system that will help to manage elections easily and securely. This voting system can be used for casting votes during elections held in universities and organizations. In this system, the voter does not have to go to the polling booth to cast their vote. They can use their personal computer to do so. The system can be used anywhere. It is not just employed for national elections; it may also be used for any sort of elections required. The system comprises 2 modules: User and Admin.
16	College Placement System Using Python	To address the above-mentioned problem, we have developed a College Placement System. Our system can help to resolve the issue of manual work that makes the process slow and other problems such as inconsistency and ambiguity in operations. The system intends to assist college faculties and recruiting companies in analyzing the placement process more efficiently. The system comprises 3 modules: Students, Placement Department/Admin, and College Management.
17	Library Management System	Library Management System Using Python To tackle this problem, we have designed a Library Management System using Python. Library management means efficient and effective management of material (information sources), machinery, men (human resources), technology, and money to meet the objectives of the library. Our system can help librarians to work easily. This computerization of the library helps in many instances of its maintenance. It reduces the workload of management as most of the manual work done is reduced.
18	Ambulance Booking System	Ambulance Booking System using Python In our Ambulance Booking System, people can easily book an ambulance. There are three major modules namely User, Ambulance, and Hospital. Users can register and log in using credentials. Users can edit their profile and change their password in an emergency. Any Upcoming Ambulance Booking details if anyone wants to Book an Ambulance or if there is an Emergency. For booking an ambulance users have to select ambulance size, pick-up point & hospital, and date and time. In an emergency will automatically book the nearest ambulance and hospital. Users will get a list of All the bookings of Ambulances. The front end involves HTML, CSS, and JavaScript and the back end involves Python. The framework used is Django and the database is MySQL.
19	Bulk File Rename/ Image Resize Application	Bulk File Rename/ Image Resize Application This is an advanced project which needs you to be well-versed in Machine Learning. We will begin by teaching the program how to pre-process data, then perform a few resize and rename images tasks. As the program starts learning, it can handle bulk functions at once.
20	Weather Monitoring System	Build a weather monitoring system that collects data from various sensors and displays weather information such as temperature, humidity, and pressure.
21	Real Estate Management System	Create a system for managing real estate properties, listings, and inquiries. Implement features for property search, viewing appointments, and contact management.
22	Generator for Mad Libs	Working on Mad Libs Generator is one of the finest ways to start exploring hands-on projects on python for kids. This project is ideal for those who are just getting started with software programming. This project will show you how to modify user-inputted data by focusing on Strings, Variables, and Concatenation. The program is set up in such a way that users will be asked to submit a sequence of inputs that will be used to create a Mad Lib. Mab lib is a beginner-friendly Python project.

23	Dice Rolling Simulator	We'll be emulating a rolling dice, as the program's name suggests. This is one of the more fascinating Python projects since, with each dice roll, it generates a random number. The user is free to roll the dice as many times as he likes. When the client rolls the dice, the application generates random numbers in between 1 and 6. The client will then see the number displayed. Clients will also be asked if they want to roll the dice again. A function that may randomly select and print a number between 1 and 6 should also be included in the programme. This collection of beginner-level Python projects will aid in the development of a solid foundation in essential programming ideas.
24	Python Story Generator	This is a python project that is both entertaining and fascinating and will delight children. In a nutshell, the computer Users will be asked for input by the computer such as a place's name, an activity, and so on, and then create a tale based on the information. The plot will always be the same, with just minor variations in the input.
25	YouTube Video Downloader-	Working on a YouTube video downloader is one of the finest ways to get started with your hands-on projects on python for students. Every month, about a billion people watch YouTube. On occasion, we enjoy downloading videos endlessly. Although YouTube does not offer this feature, you can make a simple app with a user interface that allows you to download videos on YouTube in a variety of formats and quality levels. This project appears complicated at first, but it becomes simple once you get started.
26	What's the word?	The focus of this name is on the user guessing the randomly produced word. You can make a list of words from which the word must be selected, as well as a limit on the number of guesses that can be made. After that, you're free to make your own rules! When the user submits the word, you can specify whether or not the letter written in this location appears. You'll need a function that detects whether the user is typing letters or numbers and displays appropriate error messages if so.
27	Dictionary App	Build a simple dictionary application that fetches word definitions and synonyms from an online dictionary API. A Dictionary App based project in Python is a useful program that allows users to look up the meanings, definitions, translations, and related information of words or phrases.
28	Language Translator	Build a program that translates text from one language to another using a translation API. A Language Translator project using Python is an application that allows users to input text in one language and receive a translation of that text into another language of their choice. This project typically involves the integration of translation APIs or libraries to perform the actual translation.
29	E-commerce Cart	Develop a basic e-commerce shopping cart system where users can add, remove, and purchase items. Remember that the complexity of these projects can vary depending on your skill level, and you can always add more features and polish to make them more advanced if you'd like. Additionally, these projects can serve as excellent portfolio pieces or practice to enhance your Python programming skills.
30	Appointment Scheduling System	Build a system for scheduling appointments for services like doctor visits, salon appointments, etc. Include features for appointment booking, reminders, and cancellations.

SELF-LEARNING THROUGH MOOCs (Cognitive Skills):

Certification

1. Certification from Guvi
2. https://www.codechef.com/practice/python
3. https://skillsforall.com/course/python-essentials-2