COURSE PACK

1. THE SCHEME

<table>
<tr><td>Course Title</td><td colspan="3">Python Programming</td><td colspan="2">Course Type</td><td colspan="4">Theory</td></tr>
<tr><td>Course Code</td><td colspan="3">R1UC427B</td><td colspan="2">Class</td><td colspan="4">B-Tech Core and All specialization (II-YR)</td></tr>
<tr><td rowspan="6">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Credit Hours</td><td colspan="4" rowspan="2">Total Number of Classes per Semester</td><td colspan="2">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>1</td><td>1</td><td rowspan="5">Theory</td><td rowspan="5">Tutorial</td><td rowspan="5">Practical</td><td rowspan="5">Self-study</td><td rowspan="5">LAB TEST</td><td rowspan="5">LAB EXAM</td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td></tr>
<tr><td>Practical</td><td>1</td><td>2</td></tr>
<tr><td>Self-study</td><td>0</td><td>0</td></tr>
<tr><td>Total</td><td>2</td><td>3</td></tr>
<tr><td>Course Lead</td><td colspan="2">S.P.Ramesh</td><td>Course Coordinator</td><td colspan="6">Deepak</td></tr>
<tr><td rowspan="2">Names Course Instructors</td><td colspan="3">Theory</td><td colspan="6">Practical</td></tr>
<tr><td colspan="3">1. Ankita Gupta
2. Dr. AMIT BATRA
3. Dr. Arvind Yadav
4. Kanika Thakur
5. Mahesh Kumar Chouhan
6. Manish Verma
7. Mr. Deepak
8. Ms. Rajeshwari Sisodia
9. Neha Kumari
10. Piyoush Kumar
11. PREM KUMARI VERMA
12. Priya Pandey
13. S.P.RAMESH
14. Sapna Sharma
15. Vaibhav Kumar Singh
16. Vivek Sharma</td><td colspan="6">1. Abhishek Chandra
2. Ankita Gupta
3. Ashish Shrivastava
4. CHANDRAMALA AMARJI
5. Dr. Arvind Yadav
6. Ganesh Kumar Mahato
7. Hariprasath K
8. Kanika Thakur
9. Lenin Narengbam
10. Mahesh Kumar Chouhan
11. Manoj Kumar Tyagi
12. MONIKA
13. Mr. Rahul Kumar
14. Munish Khanna
15. MUZAFAR MEHRAJ MISGAR
16. Nisha Vasudeva
17. Rishav Raj
18. Shabir Ali
19. Sumit Kumar Mishra
20. UPPILIRAJA
21. Vaibhav Kumar Singh
22. Vimal Singh</td></tr>
</table>

2. COURSE OVERVIEW

This course introduces the basics of Python programming and familiarizes learners with Python syntax, data types, operators, user input, conditional statements, and looping structures. It covers essential data structures such as strings, lists, tuples, sets, and dictionaries along with functions, modules, and file handling. The course also explains object-oriented programming concepts including classes, objects,

constructors, inheritance, encapsulation, abstraction, and polymorphism. In addition, learners gain basic exposure to Python libraries such as NumPy, Pandas, and Matplotlib for simple data analysis and data visualization.

3. COURSE OBJECTIVES

- To introduce the fundamentals of Python programming and its syntax.
- To develop problem-solving skills using Python control structures and data types.
- To understand and apply Python data structures, functions, and file handling.
- To explain object-oriented programming concepts such as classes, inheritance, encapsulation, and polymorphism.
- To provide basic exposure to Python libraries for data processing and visualization.

4. PREREQUISITE COURSE

PREREQUISITE COURSE REQUIRED	No	
If, yes please fill in the Details	Course code	Course Title
	NA	NA

5. PROGRAM OUTCOMES (POs):

PO No.	Description of the Program Outcome
PO1	Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	Problem Analysis: Identify, formulate, review research literature and analyse complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4).
PO3	Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5).

PO4	Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6).
PO6	The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO7	Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9).
PO8	Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO9	Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.
PO10	Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO11	Life-Long Learning: Recognize the need for, and have the preparation and ability for: i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8).

6. PROGRAM SPECIFIC OUTCOMES (PSOs):

Program Specific Outcomes (PSO) are statements that describe what the graduates of a discipline-specific program should be able to do. Two to Three PSOs per program should be designed.

PO No.	Description of the Program-Specific Outcome
PSO1	Have the ability to work with emerging technologies in Computer Science and Engineering requisite to Industry 4.0.
PSO2	Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

7. COURSE CONTENT

(THEORY + PRACTICAL)

Introduction to Python
Introduction to Python, Features of Python, Python versions, Install Python and Environment, Setup, Python Identifiers, Keywords and Indentation, Python Data Types, Variables, Operators in Python, Assignment, Logical, Arithmetic, etc. Taking User Inputs, Conditional Statements If else and Nested If else and Elif. For Loop, While Loop & Nested Loops.
Python Data Structures and Functions
Introduction to Strings, Creation and Indexing, String Slicing, Built-in String Methods, List, Tuple, Sets and Dictionary, Basic Operations, Slicing & Functions and Methods, User Defined Functions, Lambda Function, Importing Modules, Math Module, Files
Object-Oriented Programming in Python
Basics of Object-Oriented Programming, Creating Class and Object, Constructors in Python, Inheritance in Python, Inbuilt class methods and attributes, Inheritance: Single, Multi Level and Multiple, Data Abstraction, Encapsulation.
Python Libraries and Data Visualization
Polymorphism: Overriding and Overloading, Difference between method overloading and overriding. Basics of NumPy, Panda, Matplotlib.

8. COURSE OUTCOMES (COs)

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

CO No.	Description of the Course Outcome
R1UC427B . 1	Identify the basics of Python programming and its syntax.
R1UC427B . 2	Apply Python data structures such as lists, tuples, sets, and dictionaries to solve problems.
R1UC427B . 3	Develop programs using object-oriented concepts in Python.
R1UC427B . 4	Use basic Python libraries for simple data analysis and visualization.

9. TAXONOMY LEVEL OF THE COURSE OUTCOMES

Mapping of COs with Bloom's Level

CO No.	Remember **KL1**	Understand **KL 2**	Apply **KL 3**	Analyse **KL 4**	Evaluate **KL 5**	Create **KL 6**
R1UC427B . 1	√	√				
R1UC427B . 2		√	√			
R1UC427B . 3			√	√		
R1UC427B . 4				√	√	√

10. COURSE ARTICULATION MATRIX

COs#/ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
R1UC427B . 1	2	1			1						1	1	
R1UC427B. 2	2	2	1		2						1	2	
R1UC427B . 3	1	2	2	1	1						1	2	1
R1UC427B . 4	1	2	2	1	2			1	1		2	2	1

Note: 1-Low, 2-Medium, 3-High \ *first semester first course and first Course Outcome

11. TYPICAL EXAMPLE OF COURSES, CREDIT HOURS AND TEACHING HOURS

Type of Course	Credits Hours					Hours of engagement/ Week					12 weeks/ semester	Remarks
	Theory	Tutorial	Practical	Self-study	Total	Theory	Tutorial	Practical	Self-study	Total	Total no. of classes	
Theory + Practical	1	0	1	0	2	1	0	2	0	3	40	25 classes for lab

L-No	Topic for Delivery	Tutorial / Practical Plan	Skill	Competency
1	Introduction to Python, Features of Python, Python versions, Install Python and Environment Setup	Tutorial	Develop basic Python programs using correct syntax, data types, operators, user input, and control flow constructs.	CO1
2	Python Identifiers, Keywords and Indentation, Python Data Types, Variables,	Tutorial		
3	Operators in Python, Assignment, Logical, Arithmetic, etc. Taking User Inputs,	Tutorial		
4	Conditional Statements If else and Nested If else and Elif. For Loop, While Loop & Nested Loops.	Tutorial		
5	Python Data Types & Variables	Practical		
6	Operators in Python	Practical		
7	Conditional Statements	Practical		
8	Looping Statements	Practical		
9	Pattern Printing	Practical		
10	Introduction to Strings, Creation and Indexing, String Slicing, Built-in String Methods,	Tutorial	Ability to work with strings and Python data structures, apply built-in and user-defined functions, use modules and lambda expressions, and perform basic	CO2
11	List, Tuple, Sets and Dictionary,	Tutorial		
12	Basic Operations, Slicing & Functions and Methods, User Defined Functions,	Tutorial		
13	Lambda Function, Importing Modules, Math Module, Files	Tutorial		
14	Strings	Practical		
15	Strings Slicing	Practical		
16	Lists	Practical		
17	Tuples Slicing	Practical		
18	Dictionaries	Practical		

19	Functions	Practical	file handling operations.	
20	Lambda Function, Importing Modules, Math Module	Practical		
21	Creating Class and Object	Tutorial	to design and implement object-oriented Python programs using classes, objects, constructors, inheritance, abstraction, and encapsulation.	CO3
22	Basics of Object-Oriented Programming, Creating Class and Object,	Tutorial		
23	Constructors in Python, Inheritance in Python,	Tutorial		
24	Inheritance: Single, Multi Level and Multiple,	Tutorial		
25	Data Abstraction, Encapsulation.	Tutorial		
26	Creating Class and Object	Practical		
27	Constructors in Python	Practical		
28	Types of Inheritance	Practical		
29	Encapsulation	Practical		
30	Data Abstraction	Practical		
31	Polymorphism: Overriding and Overloading,	Tutorial	Ability to apply polymorphism concepts and use NumPy, Pandas, and Matplotlib to develop simple Python applications such as a student management system using OOP principles.	CO4
32	Difference between method overloading and overriding	Tutorial		
33	Basics of NumPy, Panda, Matplotlib	Tutorial		
34	Student Class Management System: Use class, object, constructor.	Tutorial		
35	Library Management System: Use inheritance for books and members.	Practical		
36	Shape Area Calculator: Use method overriding for different shapes	Practical		
37	Basics of NumPy	Practical		
38	Basics of Pandas	Practical		
39	Basics of Matplotlib	Practical		
40	Attendance Visualization: Plot attendance using Matplotlib.	Practical		

12. BIBLIOGRAPHY

Text Book:

1. Reema Thareja, Python Programming Using Problem Solving Approach, Oxford University Press
2. Allen B. Downey, Think Python, O'Reilly Media
3. Sourav Sahay, Object-Oriented Programming with Python, Oxford University Press
4. Jake VanderPlas, Python Data Science Handbook, O'Reilly Media

Reference Books:

1. Mark Lutz, "Learning Python", McGraw-Hill publication, 2nd Edition, 2010
2. Luciano Ramalho, "Fluent Python", O'Reilly Media, 1st Edition, 2015

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

13. COURSE ASSESSMENT

Assessment forms an integral part of curriculum design. A learning-teaching system can only be effective if the student's learning is measured at various stages which means while the student processes learning (Assessment for Learning) a given content and after completely learning a defined content (Assessment of Learning). Assessment for learning is referred to as formative assessment, that is, an assessment designed to inform instruction.

The ability to use and apply the knowledge in different ways may not be the focus of the assessment. With regard to designing assessments, the faculty members must be willing to put in the time required to create a valid, reliable assessment, that ideally would allow students to demonstrate their understanding of the information while remaining. The following are the five main areas that assessment reporting should cover.

1. **Learning Outcomes**: At the completion of a program, students are expected to know their knowledge, skills, and attitude. Depending on whether it is a UG or PG program, the level of sophistication may be different. There should be no strict rule on the number of outcomes to be achieved, but the list should be reasonable, and well-organized.
2. **Assessable Outcomes**: After a given learning activity, the statements should specify what students can do to demonstrate. Criteria for demonstration are usually addressed in rubrics and there should be specific examples of work that doesn't meet expectations, meets expectations, and exceeds expectations. One of the main challenges is faculty communication whether all faculty agreed on explicit criteria for assessing each outcome. This can be a difficult accomplishment when multiple sections of a course are taught or different faculty members. Hence there is a need for common understanding among the faculty on what is assessed and how it is assessed.

3. **Assessment Alignment**: This design of an assessment is sometimes in the form of a curriculum map, which can be created in something as easy as an Excel spreadsheet. Courses should be examined to see which program outcomes they support, and if the outcome is assessed within the course. After completion, program outcomes should be mapped to multiple courses within the program.
4. **Assessment Planning**: Faculty members need to have a specific plan in place for assessing each outcome. Outcomes don't need to be assessed every year, but faculty should plan to review the assessment data over a reasonable period of time and develop a course of action if the outcome is not being met.
5. **Student Experience**: Students in a program should be fully aware of the expectations of the program. The program outcomes are aligned on the syllabus so that students are aware of what course outcomes they are required to meet, and how the program outcomes are supported. Assessment documents should clearly communicate what is being done with the data results and how it is contributing to the improvement of the program and curriculum.

Designing quality assessment tools or tasks involves multiple considerations if it is to be fit for purpose. The set of assessments in a course should be planned to provide students with the opportunity to learn as they engage with formative tasks as well as the opportunity to demonstrate their learning through summative tasks. Encouraging the student through the use of realistic, authentic experiences is an exciting challenge for the course faculty team, who are responsible for the review and quality enhancements to assessment practices.

14. FORMATIVE AND SUMMATIVE ASSESSMENT

Assessment Pattern for Theory Course:

Type of Course (C)	CIE			Total Marks		Grand Total Marks	Final Marks CIE*0.5+SEE*0.5
	LAB (Daily work/ Record)	MTE	Lab Exam	CIE	SEE		
Integrated	25	50	25	100	100	200	100

PROBLEM-BASED LEARNING:

1. Develop a Python program to accept student marks, calculate total, average, and assign grades using conditional statements.
2. Write a program to analyze a list of numbers and find the maximum, minimum, sum, and average using loops.
3. Create a menu-driven Python application to perform basic arithmetic operations using user input.
4. Design a Python program to check password strength based on length, digits, and special characters.
5. Implement a program to manage employee records using lists and dictionaries.
6. Write a Python program to simulate a simple banking system (deposit, withdraw, balance check).
7. Develop a program to count word frequency in a given text file.
8. Create a Python program using functions to generate Fibonacci series and prime numbers.
9. Design a program to validate email and mobile numbers using string operations.
10. Implement a Python program to model a real-world object (Student/Book/BankAccount) using classes and objects.

PRACTICE PROBLEMS

Practical 1: Python Data Types & Variables

1. Print "Hello, World"
2. Program to declare and print different types of variables
3. Program to take user input and display its data type
4. Program to swap two variables
5. Program to assign multiple variables in a single line.

Practical 2: Operators in Python

6. Program using arithmetic operators
7. Program using relational operators
8. Program using logical operators
9. Program using assignment operators

Practical 3: Conditional Statements

10. Program using simple if statement
11. Program using if–else statement
12. Program using nested if–else
13. Program using elif ladder

Practical 4: Looping Statements

14. Program using for loop
15. Program using while loop
16. Program using nested for loop
17. Program using break statement
18. Program using continue statement

Practical 5: Pattern Printing

19. Program to print star pattern
20. Program to print number pattern
21. Program to print pyramid pattern
22. Print alphabet triangle pattern
23. Print Pascal's triangle pattern

Practical 6: Strings

24. Program to create and index a string
25. Program demonstrating string slicing

26. Program using built-in string methods
27. Program to check palindrome string
28. Program to find length of a string
29. Program to reverse a string
30. Program to check palindrome string

Practical 7: Lists

31. Program to create and access list elements
32. Program demonstrating list slicing
33. Program using built-in list methods
34. Program to find sum of list elements
35. Program to find largest and smallest element in a list
36. Program to remove duplicate elements from a list
37. Program to sort a list

Practical 8: Tuples

38. Program to create and access tuple
39. Program demonstrating tuple slicing
40. Program to convert tuple to list
41. Program to find length of a tuple
42. Program to concatenate two tuples

Practical 9: Sets and Dictionary

43. Program to perform set operations
44. Program using set methods
45. Program to create and print a dictionary
46. Program to add new key-value pairs to a dictionary
47. Program to update and modify dictionary values
48. Program to delete elements from a dictionary

Practical 10: Functions

49. Program to check prime number using function
50. Program to find greatest of three numbers using function
51. Program to find sum of digits using function

52. Program to perform calculator operations using functions

Practical 11: Lambda Function, Importing Modules, Math Module

53. Program to find square of a number using lambda function
54. Program to add two numbers using lambda function
55. Program to import a module and use its functions
56. Program to use math module constants (pi, e)
57. Program to perform square root and power operations using math module
58. Program to use trigonometric functions (sin, cos, tan)

Practical 12: Creating Class and Object

59. Program to create a simple class and object
60. Program to create a student class and display details
61. Program to create an employee class and calculate salary

Practical 13: Constructors

62. Program using default constructor
63. Program using parameterized constructor

Practical 13: Inheritance

64. Program demonstrating single inheritance
65. Program demonstrating multilevel inheritance
66. Program demonstrating multiple inheritance

Practical 14: Inheritance

67. Program demonstrating encapsulation
68. Program demonstrating abstraction using abstract class
69. Program demonstrating method overriding

Practical 15: Inheritance

70. Program to create NumPy array
71. Program to create Pandas Series
72. Program to plot line graph, bar chart, pie chart

Practical 16:

73. Student Class Management System: Use class, object, constructor.
74. Library Management System: Use inheritance for books and members.
75. Shape Area Calculator: Use method overriding for different shapes
76. Attendance Visualization: Plot attendance using Matplotlib.

SOME SAMPLE PROJECT(Psychomotor skills)

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

- Create a code generator.
- Build an interactive quiz.
- Tic-Tac-Toe by Text.
- Make a temperature/measurement converter.

Link for more project ideas:

1. https://data-flair.training/blogs/python-project-ideas/
2. https://realpython.com/tutorials/projects/

SELF-LEARNING THROUGH MOOCs (Cognitive Skills):

Certification

1. https://www.codechef.com/practice/python
2. https://skillsforall.com/course/python-essentials-1