GUJARAT TECHNOLOGICAL UNIVERSITY (GTU)

Competency-focused Outcome-based Green Curriculum-2021 (COGC-2021)
Semester-III

Course Title: Scripting Language - Python
(Course Code: 4330701)

Diploma programme in which this course is offered	Semester in which offered
Computer Engineering	Third

1.   RATIONALE

Python is an open-source, high-level, general-purpose programming language used for software development. It is one of the most popular programming languages in the world today and known for its simplicity as well as rich library. It is widely used programming language in various domains, such as Automation, Server-side Web Development, Tools Development, Game Programming, Blockchain, Data Science, Artificial Intelligence, Machine Learning, Big Data etc. It's relatively easy to learn to use and incredibly versatile.

This course aims to teach the basics of Python programming. The course focuses on how to use the building blocks of Python programming to solve different problems. At the end of the course, students will be able to develop simple applications using Python programming. This knowledge will provide a solid foundation for exploring advanced applications of Python programming in the different domains mentioned above.

2.   COMPETENCY

The course content should be taught and implemented with the aim of developing different types of skills so that students are able to acquire the following competencies:

● **Develop simple applications using scripting language Python.**

3.   COURSE OUTCOMES (COs)

The practical exercises, the underpinning knowledge and the relevant soft skills associated with this competency are to be developed in the student to display the following Course Outcomes (COs):

a)   Develop programs to solve the given simple computational problems.
b)   Apply control flow structures to solve the given problems.
c)   Implement data structures lists, tuples, sets and dictionaries to solve the given problems.
d)   Apply modular programming approach to solve given problems using user-defined functions.
e)   Perform string manipulation and file operations to solve a given problem.

4. TEACHING AND EXAMINATION SCHEME

Teaching Scheme (In Hours)			Total Credits (L+T/2+P/2)	Examination Scheme				
				Theory Marks		Practical Marks		Total Marks
L	T	P	C	CA*	ESE	CA	ESE	
3	-	4	5	30	70	25	25	150

(*): Out of 30 marks under the theory CA, 10 marks are for assessment of the micro-project to facilitate integration of COs and the remaining 20 marks is the average of 2 tests to be taken during the semester for the assessing the attainment of the cognitive domain UOs required for the attainment of the COs.

Legends: L-Lecture; **T** – Tutorial/Teacher Guided Theory Practice; **P** - Practical; **C** – Credit, **CA** - Continuous Assessment; **ESE** - End Semester Examination.

5. SUGGESTED PRACTICAL EXERCISES

The following practical outcomes (PrOs) are the subcomponents of the COs. *Some of the **PrOs** marked '*' are compulsory, as they are crucial for that particular CO at the 'Precision Level' of Dave's Taxonomy related to 'Psychomotor Domain'.*

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
1	**Environment Setup** Install and configure the Python environment. Run basic Python commands to verify the Python environment.	1	02
2	**Input-Output** Write a program to read your name, contact number, email, and birthdate and print those details on the screen.	1	02
3	**Variables, operators, Expressions** i. Write a program to convert temperature from Celsius to Fahrenheit. Equation to convert Celsius to Fahrenheit: $$F = (9/5) * C + 32.$$ ii. Write a program to compute the slope of a line between two points (x1, y1) and (x2, y2). $$Slope = \frac{y_2 - y_1}{x_2 - x_1}$$ iii. Write a program to calculate simple and compound interest. $$Simple\ Interest = \frac{P * R * T}{100}$$ $$Compound\ Interest = P * \left(1 + \frac{R}{100 * n}\right)^{n * T}$$	1	08

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	iv. Write a program to get change values in Quarter, Dime, Nickels and Pennies, and calculate the value of change in Dollars. Consider Quarter = 0.25 \$, Dime = 0.10 \$, Nickels = 0.05 \$ and Penny = 0.01 \$. v. Write a program to find a maximum of given three numbers (Use ternary operator). vi. Write a program to calculate area and volume of Sphere. $$Area\ of\ Sphere\ =\ 4\,\pi\,r^2$$ $$Volume\ of\ Sphere\ =\ \frac{4}{3}\,\pi\,r^3$$ vii. Write a program that computes the real roots of a given quadratic equation (Use math library). $$Discriminant\ \Delta\ =\ b^2 - 4\,a\,c$$ $$Real\ Roots\ =\ \frac{-b\ \pm\ \sqrt{\Delta}}{2\,a}$$ viii. Write a program to determine the length of ladder required to reach a given height when leaned against the house. The height and the angle of the ladder are given as inputs (Use math Library).		
4	**Decision-Making Structures** i. A year is a Leap year if it is divisible by 4, unless it is a century year that is not divisible by 400 (1800 and 1900 are not leap years, 1600 and 2000 are leap years). Write a program that calculates whether a given year is a leap year or not. ii. Many companies pay time-and-a-half for any hours worked above 40 hours in a given week. Write a program to input the number of hours worked and hourly rate and calculate the total wages for the week. iii. The Body Mass Index (BMI) is calculated as a person's weight (in kg), divided by the square of the person's height (in meters). If the BMI is between 19 and 25, the person is healthy. If the BMI is below 19, then the person is underweight. If the BMI is above 25, then the person is overweight. Write a program to get a person's weight (in kgs) and height (in cms) and display a message whether the person is healthy, underweight or overweight. $$BMI\ =\ \frac{Weight\ in\ kg}{(Height\ in\ m)^2}$$	2	04

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	iv. Write a program to read the marks and assign a grade to a student. Grading system: A (>=90), B (80-89), C (70-79), D (60-69), E (50-59), F (<50). (Use the Switch case)		

5	**Loops**
	i. Write a program to read *n* numbers from users and calculate the average of those *n* numbers.
	ii. Write a program that prompts the user to enter 10 integers and displays all the combinations of picking two numbers from the 10.
	iii. Write programs to print below patterns:
	
	``
	    * 1
	   * * 1 2
	  * * * 1 2 3
	 * * * * 1 2 3 4
	* * * * * 1 2 3 4 5
	``
	
	iv. Write a program that displays an ASCII character table from ! to ~. Display the ASCII value of a character in decimal and hexadecimal. Display five characters per line.
	v. Write a program to sum the following series:
	$$\frac{1}{3} + \frac{3}{5} + \frac{5}{7} + \frac{7}{9} + \frac{9}{11} + \frac{11}{13} + \cdots + \frac{95}{97} + \frac{97}{99}$$
	vi. A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because 6 = 3 + 2 + 1, the next is 28 = 14 + 7 + 4 + 2 + 1. There are four perfect numbers that are less than 10,000. Write a program to find these four numbers.
		2	06

6	**Lists** i. Write a program to perform the below operations on the list: • Create a list. • Add/Remove an item to/from a list. • Get the number of elements in the list. • Access elements of the list using the index. • Sort the list. • Reverse the list. ii. Write a program to read n numbers from a user and print: ● Number of positive numbers. ● Number of negative numbers. ● Number of zeros.	3	05

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	● Number of odd numbers. ● Number of even numbers. ● Average of all numbers. iii. Write a program that counts the occurrences of each digit in a string. The program counts how many times a digit appears in the string. For example, if the input is "12203AB3", then the output should output 0 (1 time), 1 (1 time), 2 (2 times), 3 (2 times). iv. Write a program to eliminate duplicate values in the list. v. Write a program to randomly fill in 0s and 1s into a 4x4 2-dimension list, print the list and find the rows and columns with the most number of 1s.		

7	**Tuples, Sets and Dictionaries**
	i. Write a program to perform below operations on tuple:
	  • Create a tuple with different data types.
	  • Print tuple items.
	  • Convert tuple into a list.
	  • Remove data items from a list.
	  • Convert list into a tuple.
	  • Print tuple items.
	ii. Write a program to perform below operations on set:
	  • Create two different sets with the data.
	  • Print set items.
	  • Add/remove items in/from a set.
	  • Perform operations on sets: union, intersection, difference, symmetric difference, check subset of another set.
	iii. Write a program to perform below operations on dictionary:
	  • Create a dictionary.
	  • Print dictionary items.
	  • Add/remove key-value pair in/from a dictionary.
	  • Check whether a key exists in a dictionary.
	  • Iterate through a dictionary.
	  • Concatenate multiple dictionaries.
	iv. Write a program that is given a dictionary containing the average daily temperature for each day of the week, and prints all the days on which the average temperature was between 40 and 50 degrees.
	v. Write a program to repeatedly prompt the user to enter the capital of a state. Upon receiving the user's input, the program reports whether the answer is correct. Assume the
		3	05

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	states and their capitals are stored in dictionaries as key-value pairs.		
8	**Function** i. Write a program that defines a function (shuffle) to scramble a list into a random order, like shuffling a deck of cards. ii. Write a program that defines a function to return a new list by eliminating the duplicate values in the list. iii. Write a program to print Fibonacci sequence up to n numbers using recursion. Fibonacci sequence is defined as below: $Fibonacci\ Sequence =\ 1\ 1\ 2\ 3\ 5\ 8\ 13\ 21\ ...$ $where\ n^{th}term\ x_n =\ x_{n-1} +\ x_{n-2}$ iv. Write a program that defines a function to determine whether input number *n* is prime or not. A positive whole number *n* > 2 is prime, if no number between 2 and $\sqrt{n}$ (inclusive) evenly divides *n*. If *n* is not prime, the program should quit as soon as it finds a value that evenly divides *n*. v. Write a program that defines a function to find the GCD of two numbers using the algorithm below. The greatest common divisor (GCD) of two values can be computed using Euclid's algorithm. Starting with the values m and n, we repeatedly apply the formula: n, m = m, n%m until m is 0. At that point, n is the GCD of the original m and n (Use Recursion). vi. Write a program that lets the user enter the loan amount, number of years, and interest rate, and defines a function to calculate monthly EMI, total payment and display the amortization schedule for the loan.	4	06
9	**Modules** i. Write a program that defines functions (mean and deviation), that computes mean and standard deviation of given numbers. The formula for the mean and standard deviation of n numbers is given as: $$mean = \sum_{i=1}^{n} x_i = \frac{x_1 +\ x_2 + \cdots +\ x_n}{n}$$	4	06

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	$$deviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean)^2}{n-1}}$$ ii. Write a program that plays the popular scissor-rock-paper game. (A scissor can cut a paper, a rock can knock a scissor, and a paper can wrap a rock.) The program randomly generates a number 0, 1, or 2 representing scissor, rock, and paper. The program prompts the user to enter a number 0, 1, or 2 and displays a message indicating whether the user or the computer wins, loses, or draws. iii. Write a program to print the dates of all the Sundays in a given year. iv. Write a program to display a graph for ReLU (Rectified Linear Unit) function. ReLU function is defined as below: $$y = max\,(0, x)$$ Consider the range of x from -5 to 5. v. Write a program to create a list representing the results of 100 students in a test, where each element represents a student's marks (between 0 to 10), and display a histogram for the result. vi. Create a user defined module with simple functions for: addition, subtraction, multiplication, division, modulo, square, factorial. Write a program to import the module and access functions defined in the module.		
10	**String Processing** i. Write a program to check whether a given string is palindrome or not. ii. Write a program to read a string containing letters, each of which may be in either uppercase or lowercase, and return a tuple containing the number of vowels and consonants in the string. iii. Write a program to read a date in the format DD/MM/YYYY and print the same date in MM-DD-YYYY format. iv. Write a program that checks whether two words are anagrams. Two words are anagrams if they contain the same letters. For example, *silent* and *listen* are anagrams. v. Write a program that allows users to enter six-digit RGB color codes and converts them into base 10. In this format, the first two hexadecimal digits represent the amount of red, the second two the amount of green, and the last two the	5	06

S. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
	amount of blue. For example: If a user enters FF6347, then the output should be Red (255), Green (99) and Blue (71). vi. Numerologists claim to be able to determine a person's character traits based on the "numeric value" of a name. The value of a name is determined by summing up the values of the letters of the name, where "a" is 1 "b" is 2 "c" is 3 and so on up to "z" being 26. For example, the name "Python" would have the value 16 + 25 + 20 + 8 + 15 + 14 = 98. Write a program that calculates the numeric value of a name provided as input.		
11	**File Handling** i. Write a program to perform the below operations on files: • Create a text file and write a string to it. • Read an entire text file. • Read a text file line by line. • Write a string to a file. • Write a list of strings to a file. • Count the number of lines, words in a file. ii. Write a program that reads a text file and counts the occurrences of each alphabet in the file. The program should prompt the user to enter the filename. iii. Write a program that reads a text file and displays all the numbers found in the file. iv. Write an automated censor program that reads the text from a file and creates a new file where all of the four-letter words have been replaced by "****". You can ignore punctuation, and you may assume that no words in the file are split across multiple lines. v. Write a program that reads a text file and calculates the average word length and sentence length in that file. vi. Write a program that reads two strings stored in two different text files and prints a string containing the characters of each string interleaved. Remove white spaces from both strings before string interleaving. For example, Two strings "Hello World" and "Sky is the Limit" should generate output "HSeklyliosWtohrelLdimit"	5	06
		Total	**56**

Note

i. More **Practical Exercises** can be designed and offered by the respective course teacher to develop the industry relevant skills/outcomes to match the COs. The above table is only a suggestive list.

*ii. The following are some **sample** 'Process' and 'Product' related skills (more may be added/deleted depending on the course) that occur in the above listed **Practical Exercises** of this course required, which are embedded in the COs and ultimately the competency.*

S. No.	Sample Performance Indicators for the PrOs	Weightage in %
1	Correctness of the program.	30
2	Readability and documentation of the program/Quality of input and output displayed (messaging and formatting).	10
3	Code efficiency.	20
4	Debugging ability.	20
5	Execution of the program/answer to the sample questions.	20
	Total	**100**

6.   MAJOR EQUIPMENT/ INSTRUMENTS REQUIRED

The major equipment/instruments and Software required to develop PrOs are given below with broad specifications to facilitate procurement of them by the administrators/management of the institutes. This will ensure the proper conduct of practicals in all institutions across the state in a proper way so that the desired skills are developed in students.

S. No.	Equipment Name with Broad Specifications	PrOs. No.
1	Computer with latest configuration with Windows/Linux/Unix Operating System.	All
2	Text Editor (VS Code, Sublime Text, Atom, Vim or any other editor) or Python IDE (IDLE, PyCharm, PyDev, Spyder or any other IDE)	All
3	Python Interpreter (Versions: 3.6.x or higher)	All
4	Jupyter Notebook (Optional)	All

7.   AFFECTIVE DOMAIN OUTCOMES

The following *sample* Affective Domain Outcomes (ADOs) are embedded in many of the above mentioned COs and PrOs. More could be added to fulfil the development of this competency.
   a) Practice good housekeeping.
   b) Demonstrate working as a team leader/member.
   c) Maintain tools and equipments.
   d) Adhere to ethical practices.
   e) Follow safety practices.

The ADOs are best developed through the laboratory/field based exercises. Moreover, the level of achievement of the ADOs according to Krathwohl's 'Affective Domain Taxonomy' should gradually increase as planned below:

i.   'Valuing Level' in 1st year
ii. 'Organization Level' in 2nd year.
iii. 'Characterization Level' in 3rd year.

8. UNDERPINNING THEORY

The major underpinning Theory is formulated as given below and only higher level UOs of Revised Bloom's taxonomy are mentioned for the development of the COs and competency in the students by the teachers. (Higher level UOs automatically include lower level UOs in them). If required, more such higher level UOs could be included by the course teacher to focus on attainment of COs and competency.

Unit	Unit Outcomes (UOs)	Topics and Sub-topics
Unit-1: **Fundamentals of Python**	1.a Install and configure Python 1.b Explain general structure of python program 1.c Develop programs using variables, operators and input-output functions	1.1 Introduction to Python, History of Python, Python Features, Python Applications 1.2 Installing Python 1.3 Basic Structure of Python program 1.4 Keywords and Identifiers 1.5 Data types and Variables 1.6 Type Casting 1.7 Input-Output functions: input, print 1.8 Operators
Unit-2: **Control Flow Structures**	2.a Explain different types of Control Structures 2.b Develop programs using Decision making Structures 2.c Develop programs using Loops	2.1 Introduction to Control Structures **Decision Making Structures** 2.2 if, if-else statements 2.3 Nested if-else and if-elif-else statements 2.4 switch statement **Loops** 2.5 for loop 2.6 while loop 2.7 Nested loops 2.8 break, continue and pass statements
Unit-3: **Lists, Tuples, Sets and Dictionaries**	3.a Develop program using Structured types – Tuples, Lists, Sets and Dictionaries	3.1 Lists and operations on Lists 3.2 Tuples and operations on Tuples 3.3 Sets and operations on Sets 3.4 Dictionaries and operations on Dictionaries

Unit	Unit Outcomes (UOs)	Topics and Sub-topics

Unit-4:
 	
Functions and Modules
	4.a Develop programs using function
	4.b Develop programs using recursion
	4.c Use Built-in functions in programs
	4.d Develop programs using rand, math, datetime modules
		4.1 Introduction to Python User defined Function
		4.2 Passing parameters to a function and returning values from a function
		4.3 Recursion
		4.4 Standard Library: Built-in Functions
		4.5 Modules and Packages
		  ● rand module - Random numbers generators
		  ● math module – Mathematical functions
		  ● datetime module - Date and time functions
		  ● matplotlib module – Plotting functions
		4.6 Create and import custom user defined module

Unit-5:
 	
String Processing and File Handling
	5.a Develop programs to create and access strings
	5.b Use built-in functions to manipulate strings
	5.c Develop programs using text files
		5.1 Introduction to String
		5.2 Access String elements using index operator
		5.3 String functions
		  ● Basic functions: len, max, min
		  ● Testing functions: isalnum, isalpha, isdigit, isidentifier, islower, isupper, and isspace
		  ● Searching functions: endswith, startswith, find, rfind, count
		  ● Manipulation functions: capitalize, lower, upper, title, swapcase, replace, lstrip, rstrip, strip
		  ● Formatting functions: format, center, ljust, rjust
		5.4 Introduction to Text files
		5.5 File Handling functions:
		  ● Basic functions: open, close
		  ● Reading file: read, readline, readlines
		  ● Writing file: write, append, writelines

9. SUGGESTED SPECIFICATION TABLE FOR QUESTION PAPER DESIGN

Unit No.	Unit Title	Teaching Hours	Distribution of Theory Marks			
			R Level	U Level	A Level	Total Marks
I	Fundamentals of Python	7	2	4	4	10
II	Control Flow Statements	9	2	4	8	14
III	Lists, Tuples, Sets and Dictionaries	8	4	4	6	14
IV	Functions and Modules	9	4	4	8	16
V	String Processing and File Handling	9	4	4	8	16
Total		42	**16**	**20**	**34**	**70**

Legends: R=Remember, U=Understand, A=Apply and above (Revised Bloom's taxonomy)

Note: *This specification table provides general guidelines to assist student for their learning and to teachers to teach and question paper designers/setters to formulate test items/questions assess the attainment of the UOs. The actual distribution of marks at different taxonomy levels (of R, U and A) in the question paper may vary slightly from above table.*

10. SUGGESTED STUDENT ACTIVITIES

Other than the classroom and laboratory learning, the following are the suggested student-related co-curricular activities which can be undertaken to accelerate the attainment of the various outcomes in this course: Students should conduct the following activities in groups and prepare small reports (of 1 to 5 pages for each activity). For micro project reports should be as per the suggested format. For other activities, students and teachers together can decide the format of the report. Students should also collect/record physical evidence such as photographs/videos of the activities for their (student's) portfolio, which will be useful for their placement interviews:

a) Students are encouraged to use Jupyter Notebook for lab activities and projects.
b) Undertake micro-projects in teams.
c) Make a list of the Python-based applications or software.
d) Students are encouraged to register themselves in various MOOCs such as: Swayam, edx, Coursera, Udemy etc. to further enhance their learning.
e) Encourage students to participate in different coding competitions like hackathons, online competitions on codechef etc.
f) Encourage students to form a coding club at institute level and can help the slow learners.
g) https://www.codechef.com, in this website, very elementary programs are available, students are expected to solve those programs using Python programming.
h) https://code.org/, an hour of code may be organized and students are encouraged to participate.

11.   SUGGESTED SPECIAL INSTRUCTIONAL STRATEGIES (if any)

These are sample strategies that the teacher can use to accelerate the attainment of the various outcomes in this course:

a) Massive open online courses (***MOOCs***) may be used to teach various topics/subtopics.

b) Guide student(s) in undertaking micro-projects.

c) ***'L' in section No. 4*** means different types of teaching methods that are to be employed by teachers to develop the outcomes.

d) Guide students to do personalized learning so that students can understand the course material at their own pace.

e) Encourage students to do group learning by sharing so that teaching can easily be enhanced.

f) About ***20% of the topics/sub-topics*** which are relatively simpler or descriptive in nature is to be given to the students for ***self-learning***, but to be assessed using different assessment methods.

g) With respect to ***section No.10***, teachers need to ensure to create opportunities and provisions for ***co-curricular activities***.

h) Diagnose Essential missed learning concepts that will help the students.

i) Using the knowledge gained from this course, instruct students on how to address environmental and sustainability issues.

12.   SUGGESTED MICRO-PROJECTS

Only one micro-project is planned to be undertaken by a student that needs to be assigned to him/her in the beginning of the semester. In the first four semesters, the micro-project are group-based. However, in the fifth and sixth semesters, it should be preferably be individually undertaken to build up the skill and confidence in every student to become problem solver so that s/he contributes to the projects of the industry. In special situations where groups have to be formed for micro-projects, the number of students in the group should *not exceed three.* The micro-project could be industry application based, internet-based, workshop-based, laboratory-based or field-based. Each micro-project should encompass two or more COs which are in fact, an integration of PrOs, UOs and ADOs. Each student will have to maintain dated work diary consisting of individual contribution in the project work and give a seminar presentation of it before submission. The total duration of the micro-project should not be less than *14-16 (sixteen) student engagement hours* during the course. The student ought to submit micro-project by the end of the semester to develop the industry oriented COs.

A suggestive list of micro-projects is given here. This has to match the competency and the COs. Similar micro-projects could be added by the concerned course teacher:

a) Automating Emails: Develop a Python project to send emails automatically.

b) Alarm Clock: Develop a Python project for an alarm clock.

c) Unit Conversion Calculator: Develop a Python project to convert between commonly used units.

d) Scraping Google Results: Develop a Python project that can scrape results from google based on some query.

e) Weather app: Develop a Python project that takes the city name and returns the weather information of that city using web scraping.

f) Article Reader: Develop a Python project that automatically reads the article from the provided link.

g) Audio Book: Develop a Python project that can be used to convert a pdf into an audiobook.

h) URL shortener: Develop a Python script that shortens a given URL.

i) Tic-Tac-Toe: Develop a Python project for tic-tac-toe game.

j) Desktop Notifier in Python: Develop a Python project to generate pop-up notification messages on desktop.

k) Morse Code Encryption/Decryption program: Develop a Python project that should be able to both convert an English message file into Morse code, and a Morse code fi le into English.

13. SUGGESTED LEARNING RESOURCES

S. No.	Title of Book	Author	Publication with place, year and ISBN
1	Introduction to Computer Science Using Python: A Computational Problem–Solving Focus	Charles Dierbach	John Wiley & Sons (25 January 2013) ISBN-10: 0470555157 ISBN-13: 978-0470555156
2	Python Programming: An Introduction to Computer Science	John Zelle	Franklin, Beedle & Associates Inc; Pap/Cdr edition (1 December 2003) ISBN-10 : 1590280288 ISBN-13 : 978-1887902991
3	Introduction to Programming Using Python	Liang Y. Daniel	Pearson Education; First edition (26 February 2017) ISBN-10: 9332551847 ISBN-13: 978-9332551848
4	Core Python Programming	R. Nageswara Rao	Dreamtech Press (1 September 2021), Delhi; ISBN-10: 9390457157 ISBN-13: 978-9390457151
5	Head First Python	Paul Barry	Shroff/O'Reilly; Second edition (1 December 2016). ISBN-10: 9789352134823 ISBN-13: 978-9352134823
6	Learning Python	Lutz M	Shroff; Fifth edition (1 January 2013) ISBN-10:9351102017 ISBN-13:9351102014-978

14. SOFTWARE/LEARNING WEBSITES

a) https://www.python.org

b) https://www.learnpython.org

c) https://www.python-course.eu

d) https://nptel.ac.in

e) https://www.youtube.com

f) https://www.edx.org

g) https://www.coursera.org/in

h) https://www.udemy.com

i) https://www.codecademy.com

15. PO-COMPETENCY-CO MAPPING

Semester III	Scripting Language - Python (Course Code: 4330701)									
	POs and PSOs									
Competency & Course Outcomes	PO 1 Basic & Discipline specific knowledge	PO 2 Problem Analysis	PO 3 Design/ development of solutions	PO 4 Engineering Tools, Experimentation and Testing	PO 5 Engineering practices for society, sustainability and environment	PO 6 Project Management	PO 7 Life-long learning	PSO 1	PSO 2	PSO 3 (If needed)
Competency	**Develop simple applications using Python to solve the given problem.**									
CO a) Develop programs to solve the given simple computational problems.	3	2	1	2	-	-	1			
CO b) Apply control flow structures to solve the given problems.	3	2	2	2	-	2	2			
CO c) Implement data structures lists, tuples, sets and dictionaries to solve the given problems.	2	1	2	2	-	2	2			
CO d) Apply modular programming approach to solve the given problems using user-defined functions.	2	2	3	2	-	3	3			
CO e) Perform string manipulation and file operations to solve the given problems.	2	2	3	2	-	2	2			

Legend: '**3**' for high, '**2**' for medium, '**1**' for low or '**-**' for the relevant correlation of each competency, CO, with PO/ PSO

16.   COURSE CURRICULUM DEVELOPMENT COMMITTEE

GTU Resource Persons

S. No.	Name and Designation	Institute	Contact No.	Email
1	**Shri K N Raval, Head of the Computer Department**	**DTE**	**9428011250**	**raval.kamlesh@gmail.com**
2	**Smt. Manisha Mehta Head of Computer Department**	**Government Polytechnic - Gandhinagar**	**9879578273**	**manishamehtain@gmail.com**
3	**Smt. Jasmine Kargathala, Lecturer in Computer Engineering**	**Government Girls Polytechnic - Ahmedabad**	**9824799620**	**jdaftary@gmail.com**
4	**Shri Kartik Detroja Lecturer in Computer Engineering**	**Government Polytechnic - Porbandar**	**9972419091**	**detroja.kartik@gmail.com**