6.5.2 The institution reviews its teaching learning process, structures & methodologies of operations and learning outcomes at periodic intervals through IQAC set up as per norms and recorded the incremental improvement in various activities for first cycle incremental improvements made for the preceding one year with regard to quality.

S.No	Two examples of institutional reviews	Page No.
1	Course file	1-111
2	Evaluation	112-123

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M)
R.R Dist. Telangana -501 510

Sri Indu Institute of Engineering and Technology

Department of Computer Science and Engineering

COURSE FILE

PREPARED BY	:S.PRUDHVIRAJ
DEPARTMENT	: COMPUTER SCIENCE AND ENGINEERING
ACADEMIC YEAR	: 2020-2021
SUBJECT	:CS721PE- PYTHON PROGRAMMING(C413)
CLASS	: IV YEAR / I SEM – CSE – A– SECTION

PRINCIPAL
Sri Indu Institute of Engineering & Tc
Sheriguda(V), Ibrahimpatnam(M),
R.R Dist. Telangana 501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
Sheriguda (V), Ibrahimpatnam (M), Hyderabad,
R.R. Dist., Telangana State – 501 510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT VISION

To become prominent knowledge hub for learners, strive for educational excellence with innovative and industrial techniques so as to meet the software industry needs.

DEPARTMENT MISSION

DM1 To Provide educational ambience that enhances innovations, problem solving skills, leadership qualities, decision making, team-spirit and ethical responsibilities.

DM2 Imparting quality education with professional and personal ethics, so as to attain with challenging technological needs of industry and society.

DM3 To provide academic infrastructure and develop linkage with the world class organizations to strengthen industry-academia relationships for learners.

DM4 Provide platform to strengthen novel research in thrust area of Computer Science and Engineering to serve the needs of Government and Society.

Head of the Department
Computer Science & Engg. Dept.
SRI INDU INSTITUTE OF ENGG & TECH.
Sheriguda(V), Ibrahimpatnam(M), R.R.Dist-501 510.

PRINCIPAL
Sri Indu Institute of Engineering & Tec.
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
Sheriguda (V), Ibrahimpatnam (M), Hyderabad,
R.R. Dist., Telangana State – 501 510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES(PEO's)

PEO1 Graduates with strong academic and technical skills of modern computer science and engineering.

PEO2 Graduate with leadership qualities and ability to solve real time problems using current techniques & tools in interdisciplinary environment.

PEO3 Graduates with attitude towards lifelong learning through continuing education and professional development.

PROGRAM SPECIFIC OUTCOMES(PSO's)

PSO1 **Professional Skills:** The ability to implement computer programs of varying complexity in the areas related to web design, cloud computing and networking.

PSO2 **Problem-Solving Skills:** The ability to develop quality products using open ended programming environment

Head of the Department
Computer Science & Engg. Dept.
SRI INDU INSTITUTE OF ENGG & TECH.
Sheriguda(V), Ibrahimpatnam(M), R.R.Dist-501 510.

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Sheriguda (V), Ibrahimpatnam (M), Hyderabad,

R.R. Dist., Telangana State – 501510.

POs: KEYWORDS PROGRAM OUTCOMES

POs	KEYWORDS	PROGRAM OUTCOMES
PO1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions	Design solutions for complex engineering 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 engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO6	The engineer 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 engineering practice.
PO7	Environment and sustainability	Understand the impact of the professional engineering solutions in societal and environmental contexts, demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics	Apply ethical principles and commit to professional ethics, responsibilities, and norms of the engineering 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 engineering community and with society, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

PO11	Project management and finance	Demonstrate knowledge and understanding of the engineering 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(PSO's)

PSO1 **Professional Skills:** The ability to implement computer programs of varying complexity in the areas related to web design, cloud computing and networking.

PSO2 **Problem-Solving Skills**: The ability to develop quality products using open ended programming environment

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

IV Year B.Tech. CSE – I-Sem

PYTHON PROGRAMMING

(PROFESSIONAL ELECTIVE –II)

B.Tech. IV Year I Sem.	**L**	**T**	**P**	**C**
Course Code: CS721PE	**3**	**0**	**0**	**3**

UNIT - I

Python Basics, Objects- Python Objects, Standard Types, Other Built-in Types, InternalTypes, Standard Type Operators, Standard Type Built-in Functions, Categorizing theStandard Types, Unsupported Types.

Numbers - Introduction to Numbers, Integers, Floating Point Real Numbers, ComplexNumbers, Operators, Built-in Functions, Related Modules.

Sequences - Strings, Lists, and Tuples, Mapping and Set Types

UNIT - II

FILES: File Objects, File Built-in Function [open()], File Built-in Methods, File Built-inAttributes, Standard Files, Command-line Arguments, File System, File Execution, PersistentStorage Modules, Related Modules

Exceptions: Exceptions in Python, Detecting and Handling Exceptions, ContextManagement, *Exceptions as Strings, Raising Exceptions, Assertions, Standard Exceptions,*Creating Exceptions, Why Exceptions (Now)?, Why Exceptions at All?, Exceptions and thesys Module, Related Modules

Modules: Modules and Files, Namespaces, Importing Modules, Importing Module Attributes,Module Built-in Functions, Packages, Other Features of Modules

UNIT - III

Regular Expressions: Introduction, Special Symbols and Characters, Res and PythonMultithreaded Programming: Introduction, Threads and Processes, Python, Threads, and theGlobal Interpreter Lock, Thread Module, Threading Module, Related Modules

UNIT - IV

GUI Programming: Introduction, Tkinter and Python Programming, Brief Tour of OtherGUIs, Related Modules and Other GUIs

WEB Programming: Introduction, Wed Surfing with Python, Creating Simple Web Clients,Advanced Web Clients, CGI-Helping Servers Process Client Data, Building CGI ApplicationAdvanced CGI, Web (HTTP) Servers

UNIT – V

Database Programming: Introduction, Python Database Application Programmer's Interface(DB-API), Object Relational Managers (ORMs), Related Modules

Textbook

1. Core Python Programming, Wesley J. Chun, Second Edition, Pearson.

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

Faculty Signature
(N.SHILPA)
(Asst.Prof/CSE,SIIET)

Course Outcomes

Course: Python Programming (C413) **Class: IV–CSE – A - Section**

After completing this course the student will be able to:

C413.1 Examine Python syntax and semantics and be fluent in the use of Python flow controland functions. (Analysis)

C413.2 Demonstrate proficiency in handling Strings and File Systems. (Comprehension)

C413.3 Create, run and manipulate Python Programs using core data structures like Lists, Dictionaries and use Regular Expressions. (Synthesis)

C413.4 Interpret the concepts of Object-Oriented Programming as used in Python. (Application)

C413.5 Recognize exemplary applications related to Network Programming and Web Services. (Knowledge)

C413.6 Summarize the applications related to Databases in Python. (Evaluation)

Mapping of course outcomes with program outcomes:

High -3 Medium -2 Low-1

PO/PSO/ CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO
C413.1	2	-	3	-	-	-	-	-	1	-	-	-	-	2
C413.2	-	-	3	-	2	-	-	-	1	-	-	-	2	
C413.3	1	2	3	-	-	-	-	-	-	-	-	-	-	2
C413.4	-	2	-	-	3	-	-	-	-	-	1	-	-	-
C413.5	-	-	3	-	1	-	-	-	-	-	-	2		2
C413.6	-	2	3	-	-	-	-	-	-	-	-	1	2	-
C413	1.5	2	3	-	2	-	-	-	1	-	1	1.5	2	2

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M).
R.R. Dist. Telangana -501 510

CO – PO / PSO Mapping Justification

Course: Python Programming(C413) **Class: IV – CSE – A - Section**

PROGRAMME OUTCOMES (POs):

PO1 **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 **Design/development of solutions:**Design solutions for complex engineering 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.

PO5 **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO11 Project management and finance: Demonstrate knowledge and understanding of the engineering 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.

PROGRAM SPECIFIC OUTCOMES(PSOs):

PSO1 **Professional Skills:** The ability to implement computer programs of varying complexity in the areas related to web design, cloud computing and networking.

PSO2 **Problem-Solving Skills:** The ability to develop quality products using open ended programming environment.

C413.1 Examine Python syntax and semantics and be fluent in the use of Python flow control and functions. (Analysis)

	Justification
PO3	Learn Syntax and Semantics and create functions in Python.(level 3)

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

PO1	Gain the knowledge about objects, standard type operators in python. (level 2)
PO9	Students can develop python programs using operators. (level 1)
PSO2	Able to handle Lists, Tuples, Set Types(level 2)

C413.2 Demonstrate proficiency in handling Strings and File Systems. (Comprehension)

	Justification
PO3	Students handle Strings and Files in Python(level 3)
PO5	Understanding Exceptions, Related Modules in Python(level 2)
PO9	Students can develop user-defined modules, using modules in programs. (level 1)
PSO1	Students gain the knowledge on Built-in Functions in module and Packages. (level 2)

C413.3 Create, run and manipulate Python Programs using core data structures like Lists, Dictionaries and use Regular Expressions. (Synthesis)

	Justification
PO3	Understand Lists, Dictionaries and Regular Expressions in Python(level 3)
PO2	Analyze Special Symbols and Characters in RE module. (level 2)
PO1	Student get idea about Thread Module and Threading module(level 1)
PSO2	Solve the problems by using Regular Expressions(level 2)

C413.4 Interpret the concepts of Object-Oriented Programming as used in Python. (Application)

	Justification
PO5	Analyze the GUI Programming modules. (level 3)
PO2	Analyze the Web Services and Web Programming using python(level 2)
PO11	Creating simple Web Clients, CGI Applications(level 1)

C413.5 Recognize exemplary applications related to Network Programming and Web Services. (Knowledge)

	Justification
PO3	Learning Web Surfing with Python(level 3)
PO12	Implement Object Oriented Programming concepts in Python.(level 2)

PO5	Developing Web process Client Data (level 1)
PSO2	Ability to develop a Web (HTTP) Servers(level 2)

C413.6 Summarize the applications related to Databases in Python. (Evaluation)

	Justification
PO3	Write a Database Programming in Python(level 3)
PO2	Analyze the Python database Application Programmer's Interface (DB-API) (level 2)
PO12	Learning about Object Relational Managers (ORMs) (level 1)
PSO1	Gain the knowledge on Database Programming in Python(level 2)

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
ACADEMIC CALENDAR 2020-21
For All Constituent & Affiliated Colleges of JNTUH
B. Tech./B.Pharm. II, III & IV Years I & II Semesters

I SEM

S. No	Description	Duration	
		From	To
1.	Commencement of 1st Semester classwork	24.08.2020	
2.	1st Spell of Instructions	24.08.2020	17.10.2020 (8 Weeks)
3.	Dussehra Recess	19.10.2020	24.10.2020 (1 Week)
4.	First Mid Term Examinations	26.10.2020	31.10.2020 (1 Week)
5.	Submission of First Mid Term Exam Marks to the University on or before	07.11.2020	
6.	Parent-Teacher Meeting	13.11.2020	
7.	2nd Spell of Instructions	02.11.2020	26.12.2020 (8 Weeks)
8.	Second Mid Term Examinations	28.12.2020	02.01.2021 (1 Week)
9.	Preparation Holidays and Practical Examinations	04.01.2021	09.01.2021 (1 Week)
10.	Submission of Second Mid Term Exam Marks to the University on or before	09.01.2021	
11.	End Semester Examinations	11.01.2021	23.01.2021 (2 Weeks)

II SEM

S. No	Description	Duration	
		From	To
1.	Commencement of 2nd Semester classwork	25.01.2021	
2.	1st Spell of Instructions	25.01.2021	20.03.2021 (8 Weeks)
3.	First Mid Term Examinations	22.03.2021	27.03.2021 (1 Week)
4.	Submission of First Mid Term Exam Marks to the University on or before	06.04.2021	
5.	Parent-Teacher Meeting	09.04.2021	
6.	2nd Spell of Instructions	29.03.2021	22.05.2021 (8 Weeks)
7.	Second Mid Term Examinations	24.05.2021	29.05.2021 (1 Week)
8.	Preparation Holidays and Practical Examinations	31.05.2021	05.06.2021 (1 Week)
9.	Submission of Second Mid Term Exam Marks to the University on or before	05.06.2021	
10.	End Semester Examinations	07.06.2021	19.06.2021 (2 Weeks)
11.	Summer Vacation	21.06.2021	10.07.2021 (3 Weeks)

Note: All the laboratory courses shall be conducted once normalcy is restored.

REGISTRAR
24/08/20

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

TIME TABLE FOR AY-2020-21

Class: IV-B.Tech CSE-A | Semester: I | LH. NO: A-301 | W.E.F. 01-09-2020

Period/ Day	1	2	3	4	1:00-1:30	5	6	7
	9:40-10:30	10:30-11:20	11:20-12:10	12:10-1:00		1:30-2:20	2:20-3:10	3:10-4:00
Monday	PP	DM	PPL	DS		CC	PPL	PP
Tuesday	CC	PP LAB(BATCH-I) / DM LAB(BATCH-II)			LUNCH	PPL / DM (T)	DS	PP
Wednesday	DM	PPL	PP / PPL (T)	CC		PROJECT WORK		
Thursday	DM	CC	PP	DS / CC (T)		PPL	CO - C/ SS/ DAA	
Friday	PP	CC / DS (T)	SEMINARS			DM LAB(BATCH-I) / PP LAB(BATCH-II)		
Saturday	PPL	DS	DM / PP (T)	CC		DM	DS	PPL

(T) – Tutorial (concern faculty)

Subject Code	Subject Name	Name of the Faculty	Subject Code	Subject Name	Name of the Faculty
DM	Data Mining	Ms GS.Sravanthi	PP LAB	Python Programming Lab	Mr S.Pradvi Raj / Mr.A.Shiva Shankar
PPL	Principles of Programming Languages	Mr. R. Chandrashekar	CO - C/ SS/ DAA	Co-Curriculum/Soft Skills/ Department Association Activities	Ms. K. Rajyalakshmi/ Ms. K. Hephzibah
PP	Python Programming	Mr S.Pradvi Raj	LIB	Library	Ms. S. Anitha
DS	Distributed Systems	Mr. K.RajniKanth	INT	Internet	Ms. E. Rupa
CC	Cloud Computing	Dr.B.G.Obula Reddy	COUN	Counseling	Ms. M. Swarnaletha
DM LAB	Data Mining Lab	Ms GS.Sravanthi / Mr. B. Hari kumar	SPORTS	Sports	Mr. R. Chandrashekar
PROJECT WORK			SEMINARS		
Class In-charge: Mr S.Pradvi Raj		Mentor 1 : Mr S.Pradvi Raj		Mentor 2 : Mr R.Chandra Shekar	

Class In-Charge | Head of the Department | Principal

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering
2019-2020; 1st Semester

Course Title	PYTHON PROGRAMMING
Course Code	CS721PE
Programme	B.Tech
Year & Semester	IV-Year I-Semester
Regulation	R16
Course Faculty	Mrs. N.SHILPA, Assistant Professor , CSE-A

LESSON PLAN

S.NO	Unit	TOPIC	Number of Sessions Planned	Teaching method/Aids	REFERENCE
1.	1	Python Basics	1	ICT Tools	T1
2.		Python Objects	1	ICT Tools	T1
3.		Standard Types	1	ICT Tools	T1
4.		Other Built-in Types	1	ICT Tools	T1
5.		**Tutorial1:**(Standard Types, Other Built-in Types)	1	ICT Tools	
6.		Internal Types	1	ICT Tools	T1
7.		Standard Type Operators	1	ICT Tools	T1
8.		Standard Type Built-in Functions	1	ICT Tools	T1
9.		Categorizing the Standard Types,Unsupported Types.	1	ICT Tools	T1
10.		**Tutorial 2:(** Standard Type Built-in Functions).	1	ICT Tools	T1
11.		Introduction to Numbers, Integers.	1	ICT Tools	T1
12.		Floating Point Real Numbers	1	ICT Tools	T1
13.		Complex Numbers	1	ICT Tools	T1
14.		Operators	1	ICT Tools	T1

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana 501 510

15.		**Tutorial3:**(Floating Point Real Numbers)	1	ICT Tools	T1
16.		Built-in Functions	1	ICT Tools	T1
17.		Related Modules	1	ICT Tools	T1
18.		Sequences - Strings	1	ICT Tools	T1
19.		Sequences - Lists	1	ICT Tools	T1
20.		**Tutorial 4:**(Sequences – Strings).	1	ICT Tools	T1
21.		Sequences - Tuples	1	ICT Tools	T1
22.		Mapping and Set Types	1	ICT Tools	T1
23.		File Objects, File Built-in Function [open()]	1	ICT Tools	T1
24.		File Built-in Methods	1	ICT Tools	T1
25.		**Tutorial 5:**(Mapping and Set Types).	1	ICT Tools	T1
26.		File Built-in Attributes, Standard Files	1	ICT Tools	T1
27.		Command-line Arguments	1	ICT Tools	T1
28.		File System, File Execution	1	ICT Tools	T1
29.	**2**	Persistent Storage Modules, Related Modules	1	ICT Tools	T1
30.		**Tutorial 6:**(File System, File Execution)	1	ICT Tools	T1
31.		Exceptions in Python	1	ICT Tools	T1
32.		Detecting and Handling Exceptions	1	ICT Tools	T1
33.		Context Management	1	ICT Tools	T1
34.		Exceptions as Strings, Raising Exceptions	1	ICT Tools	T1
35.		**Tutorial 7:**(Detecting and Handling Exceptions)	1	ICT Tools	T1
36.		Assertions,Standard Exceptions	1	ICT Tools	T1
37.		Creating Exceptions	1	ICT Tools	T1
38.		Exceptions and the sys Module, Related Modules	1	ICT Tools	T1
39.		Modules and Files, Namespaces	1	ICT Tools	T1
40.		**Tutorial 8:**(Exceptions and the sys Module, Related Modules)	1	ICT Tools	T1

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

41.		Importing Modules, Importing Module Attributes	1	ICT Tools	T1
42.		Module Built-in Functions, Packages	1	ICT Tools	T1
43.		Regular Expressions: Introduction	1	ICT Tools	T1
44.		Special Symbols and Characters	1	ICT Tools	T1
45.		**Tutorial 9:**(Regular Expressions: Introduction)	1	ICT Tools	T1
46.		Res and Python Multithreaded Programming	1	ICT Tools	T1
47.	3	Threads and Processes	1	ICT Tools	T1
48.		Python Threads	1	ICT Tools	T1
49.		Global Interpreter Lock	1	ICT Tools	T1
50.		**Tutorial 10:**(Threads and Processes)	1	ICT Tools	T1
51.		Thread Module, Threading Module	1	ICT Tools	T1
52.		GUI Programming: Introduction	1	ICT Tools	T1
53.		Tkinter and Python Programming	1	ICT Tools	T1
54.		Brief Tour of Other GUIs	1	ICT Tools	T1
55.		**Tutorial 11:**(GUI Programming: Introduction)	1	ICT Tools	T1
56.		Related Modules and Other GUIs	1	ICT Tools	T1
57.	4	WEB Programming: Introduction	1	ICT Tools	T1
58.		Web Surfing with Python	1	ICT Tools	T1
59.		Creating Simple Web Clients	1	ICT Tools	T1
60.		**Tutorial 12:**(Web Surfing with Python).	1	ICT Tools	T1
61.		Advanced Web Clients	1	ICT Tools	T1
62.		CGI-Helping Servers Process Client Data	1	ICT Tools	T1
63.		Building CGI Application	1	ICT Tools	T1
64.		Advanced CGI	1	ICT Tools	T1
65.		**Tutorial 13:**(CGI-Helping Servers Process Client Data)	1	ICT Tools	T1

66.		Web (HTTP) Servers	1	ICT Tools	T1
67.	5	Database Programming: Introduction	2	ICT Tools	T2
68.		Python Database Application Programmer's Interface (DB-API)	3	ICT Tools	T2
69.		Object Relational Managers (ORMs)	2	ICT Tools	T1
70.		**Tutorial 14:(** Python Database Application Programmer's Interface (DB-API)	1	ICT Tools	T1
71.		Related Modules	1	ICT Tools	T2

TEXT BOOKS:

1. Python Programming, Uttam K Roy, Oxford University Press

WEBREFERENCES

WR1 www.w3schools.com
WR2 http://en.m.wikipedia/wiki/Computer network

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

Code No: 137GD

R16

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech IV Year I Semester Examinations, December - 2019
PYTHON PROGRAMMING
(Common to CSE, IT)

Time: 3 Hours **Max. Marks: 75**

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A

(25 Marks)

1.a)	State any four applications where python is more popular.	[2]
b)	List out the main differences between lists and tuples.	[3]
c)	What are the uses of File object?	[2]
d)	Give a brief description of several Built-in attributes related to File objects.	[3]
e)	Summarize the purpose of pipe and dot symbols used for pattern matching.	[2]
f)	Explain the basic functionality of match( ) function.	[3]
g)	What is the need of Tkinter module in python?	[2]
h)	How to create Label widget in Python?	[3]
i)	State the need of persistent storage.	[2]
j)	Discuss the SQL commands/statements used for creating, using and dropping a database.	[3]

PART – B

(50 Marks)

2.a) How to declare and call functions in Python programs? Illustrate with an example script.
b) List and explain few most commonly used built-in types in python. [5+5]

OR

3. Summarize various operators, built-in functions and standard library modules that deals with Python's numeric type. [10]

4. Explain the following file built-in functions and method with clear syntax, description and illustration:
a) open( ) b) file( ) c) seek( ) d) tell( ) e)read( ) [10]

OR

5.a) How does try-except statement work? Demonstrate with an example python code.
b) Illustrate the concept of importing module attributes in python scripts. [5+5]

6. Examine how python supports regular expressions through the're' module with brief introduction and various built-in methods related to it. [10]

OR

7.a) What is the motivation behind parallelism and state how python achieves parallelism?
b) Explain briefly about thread and threading module objects in Python. [3+7]

8. Consider a Python GUI program that produces a window with the following widgets using python code:
a) A button to retrieve the next value in that list(if there is one).This button is displayed if there is no next value in the list
b) A label to display the number of the items being displayed and the total number of items [10]

OR

9. Give an overview and demonstration of building web applications using python's cgi module. [10]

10.a) What is a cursor object? Explain various methods and attributes of cursor object.
b) What do you mean by a constructor? List and describe various constructors used for converting to different data types. [5+5]

OR

11. Describe in detail about Python SQLAlchemy ORM with a case study of Employee role database. [10]

---oo0oo---

Code No: R1621054 **R16** SET - 1

II B. Tech I Semester Supplementary Examinations, October/November - 2020
PYTHON PROGRAMMING
(Com to CSE & IT)

Time: 3 hours Max. Marks: 70

Note: 1. Question Paper consists of two parts (**Part-A** and **Part-B**)
2. Answer **ALL** the question in **Part-A**
3. Answer any **FOUR** Questions from **Part-B**

PART -A

1.	a)	What Python uses, static typing or dynamic typing? Justify your answer with an example.	(2M)
	b)	Write a for loop that prints numbers from 0 to 57, using range function.	(2M)
	c)	Write a Python statement that swaps values of two variables. (only one line statement)	(2M)
	d)	What is PIP.	(2M)
	e)	How to create a destructor in Python? Give an example.	(3M)
	f)	Write python script to print current date and time.	(3M)

PART -B

2.	a)	Explain the history of Python evolution.	(7M)
	b)	Write a Python program that reads four integers from user, prints them with a single print statement, without any space or newline between/after the values.	(7M)
3.	a)	What is the purpose of else clause for a loop? Explain how else works with while and for loops, with examples.	(8M)
	b)	Write a Python program that prints multiplication table of a given number.	(6M)
4.	a)	What is a list in Python? How to create nested lists? Demonstrate how to create and print a 3-dimensional matrix with lists.	(7M)
	b)	Write a Python program that counts the number of occurrences of a letter in a string, using dictionaries.	(7M)
5.	a)	Explain about different types of arguments in Python.	(8M)
	b)	Write a Python function that computes the harmonic sum of n. Harmonic Sum = $(1/2) + (1/4) + (1/8) + (1/16) + \ldots + (1/2^n)$	(6M)
6.	a)	Demonstrate implementation of hierarchical inheritance in Python, with a program.	(7M)
	b)	What happens if except clause is written without any Exception type? Explain with an example.	(7M)
7.	a)	Explain how to write test cases in Python.	(7M)
	b)	Write a Python turtle program to draw a blue square of size 200 and the draw a green circle which touches the square on all sides from inside.	(7M)

Code No: R1621054 **R16** SET - 1

II B. Tech I Semester Regular/Supplementary Examinations, October/November - 2018
PYTHON PROGRAMMING
(Com to CSE & IT)

Time: 3 hours Max. Marks: 70

Note: 1. Question Paper consists of two parts (**Part-A** and **Part-B**)
2. Answer **ALL** the question in **Part-A**
3. Answer any **FOUR** Questions from **Part-B**

PART –A

1.	a)	Define the scope and lifetime of a variable in Python.	(2M)
	b)	Why is * called string repetition operator?	(2M)
	c)	What are the features of tuple data structure?	(3M)
	d)	Compare fruitful and void functions.	(3M)
	e)	Is it possible to convert a class object into a floating type value?	(2M)
	f)	Give the advantages of multi-threading.	(2M)

PART -B

2.	a)	Python has developed as an open source project. Justify this statement.	(7M)
	b)	What are identifiers? Discuss the rules to name an identifier.	(7M)
3.	a)	What are the different loop control statements available in Python? Explain with suitable examples.	(7M)
	b)	Write a Python program that calculates number of seconds in a day.	(7M)
4.	a)	Explain the List Accessing Methods and List Comprehension.	(7M)
	b)	Write a Python program to read a word and print the number of letters, vowels and percentage of vowels in the word using a dictionary.	(7M)
5.	a)	Describe about variable length arguments with suitable program.	(7M)
	b)	What are the two ways of importing a module? Which one is more beneficial? Explain.	(7M)
6.	a)	How to implement method overriding in Python? Explain.	(7M)
	b)	Discuss with an example exceptions with arguments in Python.	(7M)
7.	a)	Write a program for basic web browser using Tkinter which should have a Text widget where the user can enter a URL and a Canvas to display the contents of the page.	(7M)
	b)	Explain data compression using LZMA algorithm.	(7M)

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

Set – I

I- Mid Examinations, DEC -2020

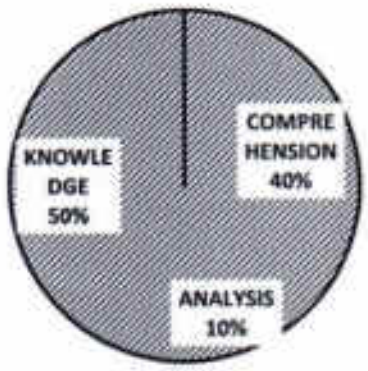
Year & Branch: IV- CSE-A,B&C Date:-28-12-2020(FN)

Subject: **Python Programming** Max. Marks:10 Time: 60mins

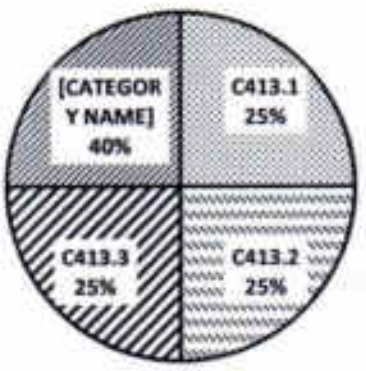
Answer any TWO questions each question carry equal marks2*5=10 marks

1	a) Explain what is Python, why python, who uses python, and the features of python.	COMPREHENSION-3M (C413.1)
	b) Differentiate Python2 and Python3	ANALYSIS-2M (C413.1)
2	Write the Python Program for following Different File operations. i. Create a file and write data into file. ii. Append data. iii. Copy from one file to another file. iv. Check whether the file is existing or not. v. Rename the file and Remove the File.	KNOWLEDGE-5M (C413.2)
3	a) Explain various String Operations in Python with example program.	COMPREHENSION-2M (C413.3)
	b) Illustrate the various standard data types in Python along with its specific functions.	COMPREHENSION-3M (C413.3)
4	a) Define Exception? Explain Exception handling with an example program.	KNOWLEDGE-2M (C413.4)
	b) Define Module and Explain Importing modules and module attributes	KNOWLEDGE-3M (C413.4)

QUESTION PAPER MAPPING WITH BT



QUESTION PAPER MAPPING WITH CO



PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M).
R.R. Dist. Telangana -501 510

Sri Indu Institute of Engineering & Technology
Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510
I- Mid Examinations, DEC -2020

Set – II

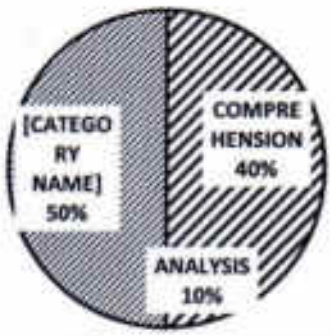
Year & Branch: IV- CSE-A,B&C Date:-28-12-2020(FN)

Subject: **Python Programming** Max. Marks:10 Time: 60mins

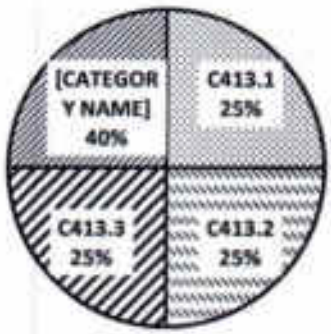
Answer any TWO questions each question carry equal marks2*5=10 marks

1	Explain Python Standard Data Types with its specific functions.	COMPREHENSION-5M (C413.1)
2	Write about Python variables and various operators.	KNOWLEDGE-5M (C413.2)
3	a) Explain various String Operations in Python with example program.	COMPREHENSION-3M (C413.3)
	b) Differentiate List and Dictionary in Python.	ANALYSIS-2M (C413.3)
4	c) What is Exception? Explain Exception handling with an example program.	KNOWLEDGE-2M (C413.4)
	d) Define Module and Explain Importing modules and module attributes	KNOWLEDGE-3M (C413.4)

QUESTION PAPER MAPPING WITH BT



QUESTION PAPER MAPPING WITH CO



PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department Computer Science and Engineering
2020-2021;1st Semester

Mid-I

Answer key-Set-I

1.a) Explain what is Python, why python, who uses python, and the features of python.
COMPREHENSION(C413.1)-3M

A. Python is a general-purpose interpreted, objected-oriented and high level programming language.

It was created by Guido van Rossum and released in 1991.

Python is designed to be highly Readable.

Python works on different platforms (windows, Mac, Linux, Raspberry pi etc)

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This Means that prototyping can be very quick.

It is generally used in a server to create web applications and used to handle big data and perform complex mathematics.

Python is used by google, yahoo!, NASA among many other organizations to create workflow.

-Python is a open source language.

-It has GUI programming support.

-Object oriented Language.

-Python is Integrated Language.

-Python is portable language.

b) Differentiate Python 2 and python 3.ANALYSIS-2M (C413.1)-2M

S.no	Python 2	Python 3
1	Its syntax is completely difficult to understand.	It is simpler and easily understandable.
2	For storing it need to define Unicode string value with "U".	Its default storing of strings is Unicode.
3	Its value of the global variable will be changed while using it inside for loop.	Its values of variables never changes.
4	Exceptions should be enclosed in notations.	Exceptions should be enclosed in parenthesis.
5	Rule of ordering camparison are complex.	Rule of ordering comparisons are simplified.

6	The Xrange() is used for the iterations.	Offers Range() function to perform iterations.
7	In this function print is print "hello".	In this function print is print ("hello").
8	Many older libraries created for python 2 is not forward – compatible.	Many recent developers are creating libraries which you can only use with Python 3.

2. Write the Python Program for following Different File operations.KNOWLEDGE(C413.2) -5M

i. Create a file and write data into file.

ii. Append data.

iii. Copy from one file to another file.

iv. Check whether the file is existing or not.

v. Rename the file and Remove the File.

i)How to Create a Text File

With Python you can create a .text files (guru99.txt) by using the code, we have demonstrated here

f= open("guru99.txt","w+")

We declared the variable f to open a file named guru99.txt. Open takes 2 arguments, the file that we want to open and a string that represents the kinds of permission or operation we want to do on the file

Here, we used "w" letter in our argument, which indicates write and will create a file if it does not exist in library

Plus sign indicates both read and write.

```
for i in range(10):
f.write("This is line %d\r\n" % (i+1))
```

- We have a for loop that runs over a range of 10 numbers.
- Using the **write** function to enter data into the file.
- The output we want to iterate in the file is "this is line number", which we declare with write function and then percent d (displays integer)
- So basically we are putting in the line number that we are writing, then putting it in a carria ge return and a new line character

ii)Append data

You can also append/add a new text to the already existing file or a new file.

```
f=open("guru99.txt", "a+")
```

Once again if you could see a plus sign in the code, it indicates that it will create a new fil e if it does not exist. But in our case we already have the file, so we are not required to create a ne w file.

```
for i in range(2):
f.write("Appended line %d\r\n" % (i+1))
```

iii. Copy from one file to another file.

1. Open one file called test.txt in read mode.
2. Open another file out.txt in write mode.
3. Read each line from the input file and write it into the output file.
4. Exit.

Program/Source Code

Here is source code of the Python Program to copy the contents of one file into another. The program output is also shown below.

```
withopen("test.txt")as f:
withopen("out.txt","w")as f1:
for line in f:
    f1.write(line)
```

iv. Check whether the file is existing or not.

Python exists()

Python exists() method is used to check whether specific file or directory exists or not. It is also used to check if a path refers to any open file descriptor or not. It returns boolean value true if file exists and returns false otherwise. It is used with os module and os.path sub module as os.path.exists(path).

In this tutorial, we will learn how to determine whether a file (or directory) exists using Python. To check this, we use Built-in library functions.

There are different ways to verify a file or directory exists, using functions as listed below.

- os.path.exists()
- os.path.isfile()
- os.path.isdir()
- pathlibPath.exists()

How to check If File Exists

- os.path.exists() – Returns True if path or directory does exists.
- os.path.isfile() – Returns True if path is File.

- os.path.isdir() - Returns True if path is Directory.
- pathlib.Path.exists() - Returns True if path or directory does exists. (In Python 3.4 and above versions)

v. Rename the file and Remove the File.

The rename() Method

The rename() method takes two arguments, the current filename and the new filename.

Syntax

```
os.rename(current_file_name, new_file_name)
```

Example

Following is the example to rename an existing file test1.txt –

```
#!/usr/bin/python

importos

# Rename a file from test1.txt to test2.txt

os.rename("test1.txt","test2.txt")
```

The remove() Method

You can use the remove() method to delete files by supplying the name of the file to be deleted as the argument.

Syntax

```
os.remove(file_name)
```

Example

Following is the example to delete an existing file test2.txt –

```
#!/usr/bin/python
import os
# Delete file test2.txt
os.remove("text2.txt")
```

3.a) Explain various String Operations in Python with example program.

A String is a sequence of characters.

In Python, a string is a sequence of Unicode characters. We can perform some operations on string like.

Creating a string:

String can be created by enclosing characters inside a single quote or double-quotes. Even triple quotes can be used in python.

Example:

```
My_string = 'Hello'
print(my_string)
```

Output: Hello

String length:

To get the length of a string, use the len() function.

Example:

```
a = "Hello, world!"
print(len(a))
```

Output: 13

String Concatenation:

To concatenate, or combine, two strings you can use the + Operator.

Example:

```
a = "Hello"
b = "World"
c = a+b
print(c)
```

Output: HelloWorld

String Lower():

The lower() method returns the string in Lower cases

Example:

```
a = "Hello, World"
print(a.lower())
```

Output: hello, world!

String Upper():

The Upper() method returns the string in upper case

Example:

```
a = "Hello, World!"
print(a.Upper())
```

Output: HELLO, WORLD!

3. b) Illustrate the various standard data types in python along with its specific functions.

COMPREHENSION(C413.4)-3M

A. Variables can hold values, and every value has a data – type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

Standard data types

Python provides various standard data types these are of:

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary

1. Numbers:

Number stores numeric values. The integer, float, and complex values belong to python number data – type.Python supports three types of numeric data.

Int: Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. python has no restriction on the length of an integer. Its value belong to int.

Float: Float is used to store floating-point numbers like 1.9, 9.903, 15.2 etc. It is accurate upto 15 decimal points.

Complex: A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j etc.

2. Sequence Type:

i. String

The string can be defined as the Sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

In the case of string handling, the operator + is used to concatenate two strings as the operation "hello"+"python" returns "hello python".

ii. List

Python Lists are similar to arrays in c. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets []. We can use slice [:] operators to access the data of the list.

iii. Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with comma(,) and enclosed in parentheses ().

3. Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. The items in the dictionary are separated with the comma(,) and enclosed in the curly braces{}.

4. Boolean

Boolean type provides two built-in values, true and false. These values are used to determine the given statement true or false. It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'.

5. Set

Python Set is the unordered collection of the data type. It is iterable, mutable (can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element.

4. a) Define Exception? Explain Exception handling with a program.KNOWLEDGE-2M (C413.4)

A. An exception is an event, which occurs during the execution of a normal flow of the program's instructions. An exception is a Python object that represents an error.

Handling an exception :

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block. After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible.

Example: try....except blocks

```
Try:
    a = 5
    b = '0'
    print(a/b)
except:
print('some error occurred.')
print("out of try except blocks.")
```

Output:

```
Some error occurred.
Out of try except blocks.
```

Else and finally:

In python, keywords else and finally can also be used along with the try and except clauses. While the except block is executed if the exception occurs inside the try block, the else block gets processed if the try block is found to be exception free.

Example:

```
try:
print("try block")
```

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

```
            x = int(input('Enter a number: '))
            y = int(input('Enter another number: '))
            z = x/y
        except ZeroDivisionError:
print("except ZeroDivisionError block")
print("Division by 0 not accepted")
        else:
print("else block")
print("Division = ", z)
        finally:
print("finally block")
            x = 0
            y = 0
print("out of try, except, else and finally blocks.")
```

Output:

```
        try block
        Enter a number:10
        Enter another number: 2
        else block
        Division = 5.0
finally block
        out of try, except, else and finally blocks.
```

b) Define Module and Explain Importing and module attributes.

A. A module is a file consisting of python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

```
    defprint_func( par):
print "Hello : ", par                                                    return
```

The import statement:

You can use any Python source file as a module by executing an import statement in some other python source file. The import has the following syntax-

```
import module1[, module2[,...moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of dictionaries that the interpreter searches before importing a module.

The from…import statement:

Python's from statement lets you import specific attributes from a module into the current namespace. The from…import has the following syntax-

From modname import name1[, name2[, … nameN]]

The form…import* statement:

It is also possible to import all names from a module into the current namespace by using the following import statement-

From modname import *

Module Attributes:

Python module has its attributes that describes it. Attributes perform some tasks or contain some information about the module. Some of the important attributes are explained below:

_name_Attribute:

The _name_ attribute returns the name of the module. By default, the name of the file (excluding the extension .py) is the value of _name_attribute.

```
>>> import math
>>>math._name_
'math'
```

In the same way, it gives the name of your custom module.

```
>>>hello._name_
'hello'
```

_doc_Attribute:

The _doc_attribute denotes the documentation string (docstring) line written in a module code.

```
>>> import math
>>>math._doc_
```

'This module is always available. It provides access to the mathematical functions defined by the C standard.'

_file_Attribute:

file is an optional attribute which holds the name and path of the module file from which it is loaded.

```
>>> import io
>>>io._file_
'C:\\python37\\lib\\io.py'
```

4. b) Define Module and Explain Importing modules and module attributes

KNOWLEDGE-3M (C413.4)

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
defprint_func( par):
print"Hello : ", par
return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax −

```
import module1[, module2[,...moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

```
#!/usr/bin/python

# Import module support
import support

# Now you can call defined function that module as follows
support.print_func("Zara")
```

When the above code is executed, it produces the following result −

```
Hello : Zara
```

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

The *from...import* Statement

Python's *from* statement lets you import specific attributes from a module into the current namespace. The *from...import* has the following syntax –

from modname import name1[, name2[, ... nameN]]

For example, to import the function fibonacci from the module fib, use the following statement –

from fib import fibonacci

This statement does not import the entire module fib into the current namespace; it just introduces the item fibonacci from the module fib into the global symbol table of the importing module.

The *from...import* * Statement

It is also possible to import all names from a module into the current namespace by using the following import statement –

from modname import *

This provides an easy way to import all the items from a module into the current namespace; however, this statement should be used sparingly.

Locating Modules

When you import a module, the Python interpreter searches for the module in the following sequences –

- The current directory.
- If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
- If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.

The module search path is stored in the system module sys as the sys.path variable. The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

The *PYTHONPATH* Variable

The PYTHONPATH is an environment variable, consisting of a list of directories. The syntax of PYTHONPATH is the same as that of the shell variable PATH.

Here is a typical PYTHONPATH from a Windows system –

set PYTHONPATH = c:\python20\lib;

And here is a typical PYTHONPATH from a UNIX system –

set PYTHONPATH = /usr/local/lib/python

Namespaces and Scoping

Variables are names (identifiers) that map to objects. A *namespace* is a dictionary of variable names (keys) and their corresponding objects (values).

A Python statement can access variables in a *local namespace* and in the *global namespace*. If a local and a global variable have the same name, the local variable shadows the global variable.

Each function has its own local namespace. Class methods follow the same scoping rule as ordinary functions.

Python makes educated guesses on whether variables are local or global. It assumes that any variable assigned a value in a function is local.

Therefore, in order to assign a value to a global variable within a function, you must first use the global statement.

The statement *global VarName* tells Python that VarName is a global variable. Python stops searching the local namespace for the variable.

For example, we define a variable *Money* in the global namespace. Within the function *Money*, we assign *Money* a value, therefore Python assumes *Money* as a local variable. However, we accessed the value of the local variable *Money* before setting it, so an UnboundLocalError is the result. Uncommenting the global statement fixes the problem.

```
#!/usr/bin/python

Money=2000
defAddMoney():
# Uncomment the following line to fix the code:
# global Money
Money=Money+1

printMoney
AddMoney()
printMoney
```

The dir( ) Function

The dir() built-in function returns a sorted list of strings containing the names defined by a module.

The list contains the names of all the modules, variables and functions that are defined in a module. Following is a simple example −

```
#!/usr/bin/python

# Import built-in module math
import math

content =dir(math)
print content
```

When the above code is executed, it produces the following result −

['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan',
'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp',
'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log',
'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh']

Here, the special string variable *__name__* is the module's name, and *__file__* is the filename from which the module was loaded.

The *globals()* and *locals()* Functions

The *globals()* and *locals()* functions can be used to return the names in the global and local namespaces depending on the location from where they are called.

If locals() is called from within a function, it will return all the names that can be accessed locally from that function.

If globals() is called from within a function, it will return all the names that can be accessed globally from that function.

The return type of both these functions is dictionary. Therefore, names can be extracted using the keys() function.

The *reload()* Function

When the module is imported into a script, the code in the top-level portion of a module is executed only once.

Therefore, if you want to reexecute the top-level code in a module, you can use the *reload()* function. The reload() function imports a previously imported module again. The syntax of the reload() function is this −

reload(module_name)

Here, *module_name* is the name of the module you want to reload and not the string containing the module name. For example, to reload *hello* module, do the following −

reload(hello)

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
#!/usr/bin/python

defPots():
print"I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above −

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

```
#!/usr/bin/python

# Now import your Phone Package.
importPhone

Phone.Pots()
Phone.Isdn()
Phone.G3()
```

When the above code is executed, it produces the following result –

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department Computer Science and Engineering

2019-2020;1st Semester

Mid-I

Answer key-Set-II

1. **Explain Python Standard Data Types with its specific functions.**

   COMPREHENSION-3M (C413.1)

Python Data Types

Variables can hold values, and every value has a data-type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

a = 5

The variable **a** holds integer value five and we did not define its type. Python interpreter will automatically interpret variables **a** as an integer type.

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda (V), Ibrahimpatnam(M),
R.R Dist. Telangana-501 510

Python enables us to check the type of the variable used in the program. Python provides us the **type()** function, which returns the type of the variable passed.

Consider the following example to define the values of different data types and checking its type.

```
a=10
b="Hi Python"
c = 10.5
print(type(a))
print(type(b))
print(type(c))
```

Output:

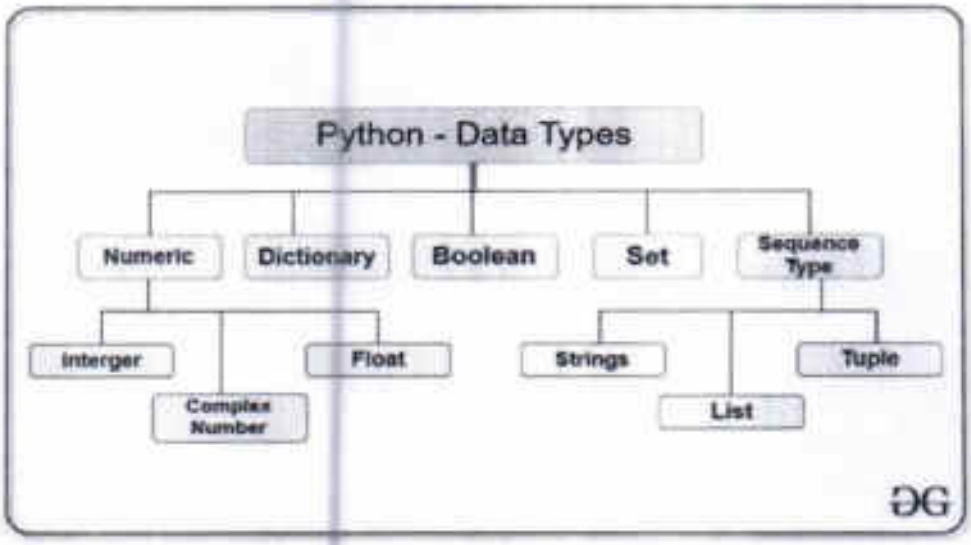
```
<type 'int'>
<type 'str'>
<type 'float'>
```

Standard data types

A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary



In this section of the tutorial, we will give a brief introduction of the above data-types. We will discuss each one of them in detail later in this tutorial.

Numbers

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type. Python provides the **type()** function to know the data-type of the variable. Similarly, the **isinstance()** function is used to check an object belongs to a particular class.

```
a = 5
print("The type of a", type(a))
b = 40.5
print("The type of b", type(b))
c = 1+3j
print("The type of c", type(c))
print(" c is a complex number", isinstance(1+3j,complex))
```

Output:

```
The type of a <class 'int'>
The type of b <class 'float'>
The type of c <class 'complex'>
c is complex number: True
```

Python supports three types of numeric data.

1. **Int** - Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to **int**
2. **Float** - Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate upto 15 decimal points.

3. **complex** - A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

Sequence Type

String

The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.

String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.

In the case of string handling, the operator + is used to concatenate two strings as the operation *"hello"+" python"* returns *"hello python"*.

The operator * is known as a repetition operator as the operation "Python" *2 returns 'Python Python'.

The following example illustrates the string in Python.

Example - 1

```
str = "string using double quotes"
print(str)
s = ""A multiline
string'''
print(s)
```

Output:

```
string using double quotes
A multiline
string
```

Consider the following example of string handling.

Example - 2

```
str1 = 'hello javatpoint' #string str1
str2 = ' how are you' #string str2
print (str1[0:2]) #printing first two character using slice operator
print (str1[4]) #printing 4th character of the string
```

```
print (str1*2) #printing the string twice
print (str1 + str2) #printing the concatenation of str1 and str2
```

Output:

```
hc
o
hello javatpointhellojavatpoint
hello javatpoint how are you
```

List

Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

Consider the following example.

```
list1  = [1, "hi", "Python", 2]
#Checking type of given list
print(type(list1))
#Printing the list1
print (list1)
# List slicing
print (list1[3:])
# List slicing
print (list1[0:2])
. # List Concatenation using + operator
. print (list1 + list1)
. # List repetation using * operator
. print (list1 * 3)
```

Output:


```
[1, 'hi', 'Python', 2]
[2]
[1, 'hi']
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
```

```
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
```

Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Let's see a simple example of the tuple.

```
tup  = ("hi", "Python", 2)
# Checking type of tup
print (type(tup))
#Printing the tuple
print (tup)
# Tuple slicing
print (tup[1:])
print (tup[0:1])
# Tuple concatenation using + operator
. print (tup + tup)
. # Tuple repatation using * operator
. print (tup * 3)
. # Adding value to tup. It will throw an error.
. t[2] = "hi"
```

Output:

```
<class 'tuple'>
('hi', 'Python', 2)
('Python', 2)
('hi',)
('hi', 'Python', 2, 'hi', 'Python', 2)
('hi', 'Python', 2, 'hi', 'Python', 2, 'hi', 'Python', 2)

Traceback (most recent call last):
  File "main.py", line 14, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```


Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.

Consider the following example.

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}
# Printing dictionary
print (d)
# Accesing value using keys
print("1st name is "+d[1])
print("2nd name is "+ d[4])
print (d.keys())
print (d.values())
```

Output:

```
1st name is Jimmy
2nd name is mike
{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}
dict_keys([1, 2, 3, 4])
dict_values(['Jimmy', 'Alex', 'john', 'mike'])
```

Boolean

Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'. Consider the following example.

```
# Python program to check the boolean type
print(type(True))
print(type(False))
print(false)
```


Output:

```
<class 'bool'>
<class 'bool'>
NameError: name 'false' is not defined
```

Set

Python Set is the unordered collection of the data type. It is iterable, mutable(can modify after

creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function **set(),** or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values. Consider the following example.

```
# Creating Empty set
set1 = set()
set2 = {'James', 2, 3,'Python'}
#Printing Set value
print(set2)
# Adding element to the set
set2.add(10)
print(set2)
#Removing element from the set
. set2.remove(2)
. print(set2)
```

Output:

```
{3, 'Python', 'James', 2}
{'Python', 'James', 3, 2, 10}
{'Python', 'James', 3, 10}
```

2. Write about Python variables and various operators.KNOWLEDGE(C413.2)-5M

Python is not "statically typed". We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.

Rules for creating variables in Python:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case-sensitive (name, Name and NAME are three different variables).
- The reserved words(keywords) cannot be used naming the variable.

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x= 5
y= "John"
print(x)
print(y)
```

output:

5

John

Python Operators

Operators in general are used to perform operations on values and variables in Python. These are standard symbols used for the purpose of logical and arithmetic operations. In this article, we will look into different types of Python operators.

In this tutorial, we going to learn various operators

- Arithmetic Operators
- Comparison Operators
- Python Assignment Operators
- Logical Operators or Bitwise Operators
- Membership Operators
- Identity Operators
- Operator precedence

1. Arithmetic Operators perform various arithmetic calculations like addition, subtraction, multiplication, division, %modulus, exponent, etc. There are various methods for arithmetic calculation in Python like you can use the eval function, declare variable & calculate, or call functions.

Arithmetic Operators

Arithmetic Operators perform various arithmetic calculations like addition, subtraction, multiplication, division, %modulus, exponent, etc. There are various methods for arithmetic calculation in Python like you can use the eval function, declare variable & calculate, or call functions.

Example: For arithmetic operators we will take simple example of addition where we will add two-digit 4+5=9

```
x= 4
y= 5
print(x + y)
```

Similarly, you can use other arithmetic operators like for multiplication(*), division (/), substraction (-), etc.

Comparison Operators

Comparison Operators In Python compares the values on either side of the operand and determines the relation between them. It is also referred to as relational operators. Various comparison operators in python are (==, != , <>, >,<=, etc.)

Example: For comparison operators we will compare the value of x to the value of y and print the result in true or false. Here in example, our value of x = 4 which is smaller than y = 5, so when we print the value as x>y, it actually compares the value of x to y and since it is not correct, it returns false.

```
x = 4
y = 5
print(('x > y  is',x>y))
```

Likewise, you can try other comparison operators (x < y, x==y, x!=y, etc.)

Python Assignment Operators

Assignment Operators in **Python** are used for assigning the value of the right operand to the left operand. Various assignment operators used in Python are (+=, - = , *=, /= , etc.).

Example: Python assignment operators is simply to assign the value, for example

```
num1 = 4
num2 = 5
print(("Line 1 - Value of num1 : ", num1))
print(("Line 2 - Value of num2 : ", num2))
```

Example of compound assignment operator

We can also use a compound assignment operator, where you can add, subtract, multiply right operand to left and assign addition (or any other arithmetic function) to the left operand.

- Step 1: Assign value to num1 and num2
- Step 2: Add value of num1 and num2 (4+5=9)

- Step 3: To this result add num1 to the output of Step 2 (9+4)
- Step 4: It will print the final result as 13

```
num1 = 4
num2 = 5
res = num1 + num2
res += num1
print(("Line 1 - Result of + is ", res))
```

Logical Operators

Logical operators in Python are used for conditional statements are true or false. Logical operators in Python are AND, OR and NOT. For logical operators following condition are applied.

- For AND operator – It returns TRUE if both the operands (right side and left side) are true
- For OR operator- It returns TRUE if either of the operand (right side or left side) is true
- For NOT operator- returns TRUE if operand is false

Example: Here in example we get true or false based on the value of a and b

```
a = True
b = False
print('a and b is',a and b))
print('a or b is',a or b))
print('not a is',not a))
```

Membership Operators

These operators test for membership in a sequence such as lists, strings or tuples. There are two membership operators that are used in Python. (in, not in). It gives the result based on the variable present in specified sequence or string

Example: For example here we check whether the value of x=4 and value of y=8 is available in list or not, by using **in** and **not in** operators.

```
x = 4
y = 8
list = [1, 2, 3, 4, 5 ];
if ( x in list ):
print("Line 1 - x is available in the given list")
else:
```

```
print("Line 1 - x is not available in the given list")
if ( y not in list ):
print("Line 2 - y is not available in the given list")
else:
print("Line 2 - y is available in the given list")
```

- Declare the value for x and y
- Declare the value of list
- Use the "in" operator in code with if statement to check the value of x existing in the list and print the result accordingly
- Use the "not in" operator in code with if statement to check the value of y exist in the list and print the result accordingly
- Run the code- When the code run it gives the desired output

Identity Operators

Identity Operators in Python are used to compare the memory location of two objects. The two identity operators used in Python are (is, is not).

- Operator is: It returns true if two variables point the same object and false otherwise
- Operator is not: It returns false if two variables point the same object and true otherwise

Following operands are in decreasing order of precedence.

Operators in the same box evaluate left to right

Operators (Decreasing order of precedence)	Meaning
**	Exponent
*, /, //, %	Multiplication, Division, Floor divisio
+, -	Addition, Subtraction
<= <> >=	Comparison operators

= %= /= //= -= += *= **=	Assignment Operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Example:

```
x = 20
y = 20
if ( x is y ):
        print("x & y  SAME identity")
y=30
if ( x is not y ):
        print("x & y have DIFFERENT identity")
```

- Declare the value for variable x and y
- Use the operator "is" in code to check if value of x is same as y
- Next we use the operator "is not" in code if value of x is not same as y
- Run the code- The output of the result is as expected

Operator precedence

The operator precedence determines which operators need to be evaluated first. To avoid ambiguity in values, precedence operators are necessary. Just like in normal multiplication method, multiplication has a higher precedence than addition. For example in 3+ 4*5, the answer is 23, to change the order of precedence we use a parentheses (3+4)*5, now the answer is 35. Precedence operator used in Python are (unary + - ~, **, * / %, + - ,&) etc.

```
v = 4
w = 5
x = 8
y = 2
z = 0
z = (v+w) * x / y;
print("Value of (v+w) * x/ y is ",  z)
```

- Declare the value of variable v,w…z
- Now apply the formula and run the code
- The code will execute and calculate the variable with higher precedence and will give the output

3.(A) various String Operations in Python with example program

COMPREHENSION-3M (C413.3)

A String is a sequence of characters.

In Python, a string is a sequence of Unicode characters. We can perform some operations on string like.

Creating a string:

String can be created by enclosing characters inside a single quote or double-quotes. Even triple quotes can be used in python.

Example:

```
My_string = 'Hello'
print(my_string)
```

Output: Hello

String length:

To get the length of a string, use the len() function.

Example:

```
a = "Hello, world!"
print(len(a))
```

Output: 13

String Concatenation:

To concatenate, or combine, two strings you can use the + Operator.

Example:

```
a = "Hello"
b = "World"
c = a+b
print(c)
```

Output: HelloWorld

String Lower():

The lower() method returns the string in Lower cases

Example:

```
a = "Hello, World"
print(a.lower())
```

Output: hello, world!

String Upper():

The Upper() method returns the string in upper case

Example:

```
a = "Hello, World!"
print(a.Upper())
```

Output: HELLO, WORLD!

3.(b)Differentiate List and Dictionary in Python. ANALYSIS(C413.3)-2M

Difference between list and dictionary

List and dictionary are fundamentally different **data structures** . A list can store a sequence of objects in a certain order such that you can index into the list, or iterate over the list. Moreover, List is a **mutable type** meaning that lists can be modified after they have been created. Python dictionary is an implementation of a hash table and is a **key-value** store. It is not ordered and it requires that the keys are hashtable. Also, it is fast for lookups by key.

Elements in a list have the following characteristics:

- They maintain their ordering unless explicitly re-ordered (for example, by sorting the list).
- They can be of any type, and types can be mixed.
- They are accessed via numeric (zero based) indices.

Elements in a Dictionary have the following characteristics:

- Every entry has a key and a value
- Ordering is not guaranteed
- Elements are accessed using key values
- Key values can be of any hashtable type (i.e. not a dict) and types can be mixed
- Values can be of any type (including other dict's), and types can be mixed

Usage:

Use a dictionary when you have a set of **unique keys** that map to values and to use a list if you have an **ordered collection** of items.

4. a) Define Exception? Explain Exception handling with a program.KNOWLEDGE(C413.4)-2M

An exception is an event, which occurs during the execution of a normal flow of the program's instructions. An exception is a Python object that represents an error.

Handling an exception :

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block. After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible.

Example: try....except blocks

```
    Try:
        a = 5
        b = '0'
        print(a/b)
    except:
print('some error occurred.')
print("out of try except blocks.")
```

Output:

```
    Some error occurred.
    Out of try except blocks.
```

Else and finally:

In python, keywords else and finally can also be used along with the try and except clauses. While the except block is executed if the exception occurs inside the try block, the else block gets processed if the try block is found to be exception free.

Example:

```
    try:
print("try block")
        x = int(input('Enter a number: '))
        y = int(input('Enter another number: '))
        z = x/y
    except ZeroDivisionError:
print("except ZeroDivisionError block")
print("Division by 0 not accepted")
    else:
print("else block")
print("Division = ", z)
    finally:
print("finally block")
        x = 0
        y = 0
print("out of try, except, else and finally blocks.")
```

Output:

```
    try block
    Enter a number:10
    Enter another number: 2
```

else block

Division = 5.0

finally block

out of try, except, else and finally blocks.

b) Define Module and Explain Importing and module attributes.

A. A module is a file consisting of python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

defprint_func(par):

print "Hello : ", par

return

The import statement:

You can use any Python source file as a module by executing an import statement in some other python source file. The import has the following syntax-

import module1[, module2[,...moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of dictionaries that the interpreter searches before importing a module.

The from...import statement:

Python's from statement lets you import specific attributes from a module into the current namespace. The from...import has the following syntax-

From modname import name1[, name2[, ... nameN]]

The form...import* statement:

It is also possible to import all names from a module into the current namespace by using the following import statement-

From modname import *

Module Attributes:

Python module has its attributes that describes it. Attributes perform some tasks or contain some information about the module. Some of the important attributes are explained below:

_name_Attribute:

The _name_ attribute returns the name of the module. By default, the name of the file (excluding the extension .py) is the value of _name_attribute.

```
>>> import math
>>>math._name_
'math'
```

In the same way, it gives the name of your custom module.

```
>>>hello._name_
'hello'
```

_doc_Attribute:

The _doc_attribute denotes the documentation string (docstring) line written in a module code.

```
>>> import math
>>>math._doc_
```

'This module is always available. It provides access to the mathematical functions defined by the C standard.'

_file_Attribute:

file is an optional attribute which holds the name and path of the module file from which it is loaded.

```
>>> import io
>>>io._file_
'C:\\python37\\lib\\io.py'
```

4. b) Define Module and Explain Importing modules and module attributes.

KNOWLEDGE(C413.4)-3M

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
defprint_func( par):
print"Hello : ", par
return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax −

import module1[, module2[,...moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

```
#!/usr/bin/python

# Import module support
import support

# Now you can call defined function that module as follows
support.print_func("Zara")
```

When the above code is executed, it produces the following result −

Hello : Zara

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

The *from...import* Statement

Python's *from* statement lets you import specific attributes from a module into the current namespace. The *from...import* has the following syntax −

from modname import name1[, name2[, ... nameN]]

For example, to import the function fibonacci from the module fib, use the following statement −

from fib import fibonacci

This statement does not import the entire module fib into the current namespace; it just introduces the item fibonacci from the module fib into the global symbol table of the importing module.

The *from...import* * Statement

It is also possible to import all names from a module into the current namespace by using the following import statement −

from modname import *

This provides an easy way to import all the items from a module into the current namespace; however, this statement should be used sparingly.

Locating Modules

When you import a module, the Python interpreter searches for the module in the following sequences –

- The current directory.
- If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
- If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.

The module search path is stored in the system module sys as the sys.path variable. The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

The *PYTHONPATH* Variable

The PYTHONPATH is an environment variable, consisting of a list of directories. The syntax of PYTHONPATH is the same as that of the shell variable PATH.

Here is a typical PYTHONPATH from a Windows system –

```
set PYTHONPATH = c:\python20\lib;
```

And here is a typical PYTHONPATH from a UNIX system –

```
set PYTHONPATH = /usr/local/lib/python
```

Namespaces and Scoping

Variables are names (identifiers) that map to objects. A *namespace* is a dictionary of variable names (keys) and their corresponding objects (values).

A Python statement can access variables in a *local namespace* and in the *global namespace*. If a local and a global variable have the same name, the local variable shadows the global variable.

Each function has its own local namespace. Class methods follow the same scoping rule as ordinary functions.

Python makes educated guesses on whether variables are local or global. It assumes that any variable assigned a value in a function is local.

Therefore, in order to assign a value to a global variable within a function, you must first use the global statement.

The statement *global VarName* tells Python that VarName is a global variable. Python stops searching the local namespace for the variable.

For example, we define a variable *Money* in the global namespace. Within the function *Money*, we assign *Money* a value, therefore Python assumes *Money* as a local variable. However, we accessed the value of the local variable *Money* before setting it, so an UnboundLocalError is the result. Uncommenting the global statement fixes the problem.

```
#!/usr/bin/python

Money=2000
defAddMoney():
# Uncomment the following line to fix the code:
# global Money
Money=Money+1

printMoney
AddMoney()
printMoney
```

The dir( ) Function

The dir() built-in function returns a sorted list of strings containing the names defined by a module.

The list contains the names of all the modules, variables and functions that are defined in a module. Following is a simple example −

```
#!/usr/bin/python

# Import built-in module math
import math

content =dir(math)
print content
```

When the above code is executed, it produces the following result −

```
['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan',
'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp',
'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log',
'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
```

'sqrt', 'tan', 'tanh']

Here, the special string variable *__name__* is the module's name, and *__file__* is the filename from which the module was loaded.

The *globals()* and *locals()* Functions

The *globals()* and *locals()* functions can be used to return the names in the global and local namespaces depending on the location from where they are called.

If locals() is called from within a function, it will return all the names that can be accessed locally from that function.

If globals() is called from within a function, it will return all the names that can be accessed globally from that function.

The return type of both these functions is dictionary. Therefore, names can be extracted using the keys() function.

The *reload()* Function

When the module is imported into a script, the code in the top-level portion of a module is executed only once.

Therefore, if you want to reexecute the top-level code in a module, you can use the *reload()* function. The reload() function imports a previously imported module again. The syntax of the reload() function is this –

```
reload(module_name)
```

Here, *module_name* is the name of the module you want to reload and not the string containing the module name. For example, to reload *hello* module, do the following –

```
reload(hello)
```

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
#!/usr/bin/python

defPots():
print"I'm Pots Phone"
```


Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

```
#!/usr/bin/python

# Now import your Phone Package.
importPhone

Phone.Pots()
Phone.Isdn()
Phone.G3()
```

When the above code is executed, it produces the following result −

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

II- Mid Examinations, FEB-2021

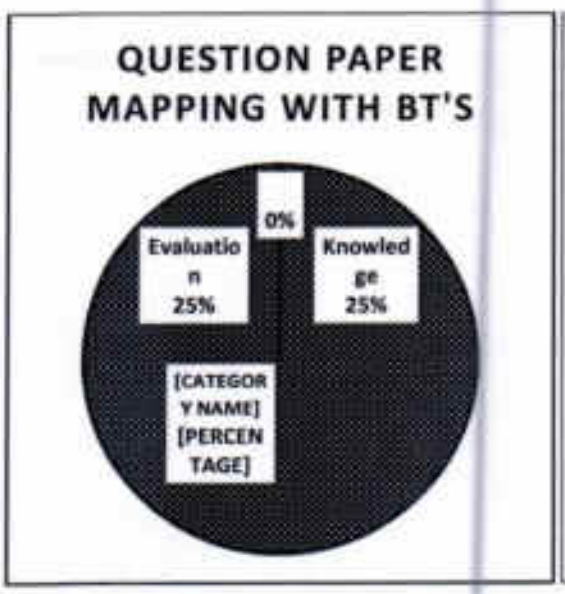
Set – I

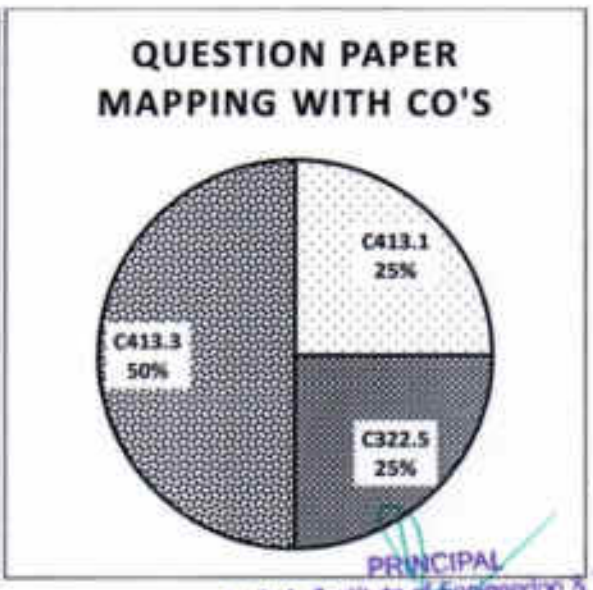
Year & Branch: IV- CSE-A,B&C Date:26-02-21(FN)

Subject: **Python Programming** Max. Marks:10 Time: 60mins

Answer any TWO questions each question carry equal marks2*5=10 marks

1. 1.A) Write in detail about various special symbols and characters in Regular Expressions.&Write a short notes on global interpreter lock.[KNOWLEDGE](C413.4)[5M]
2. Explain about thread module and threading module with an example program. [COMPREHENSION](C413.5)[5M]
3. Illustrate an introduction about GUI programming and Write how Tk is added to the applications. [COMPREHENSION](C413.6)[5M]
4. Summarize basic Database Operations in Python.& Write some SQL commands in detail. [EVALUATION](C413.6) [5M]





PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

II- Mid Examinations, FEB-2021

Set – II

Year & Branch: II- CSE-A,B&C Date:26-02-21(FN)

Subject: **Python Programming** Max. Marks:10 Time: 60mins

Answer any TWO questions each question carry equal marks2*5=10 marks

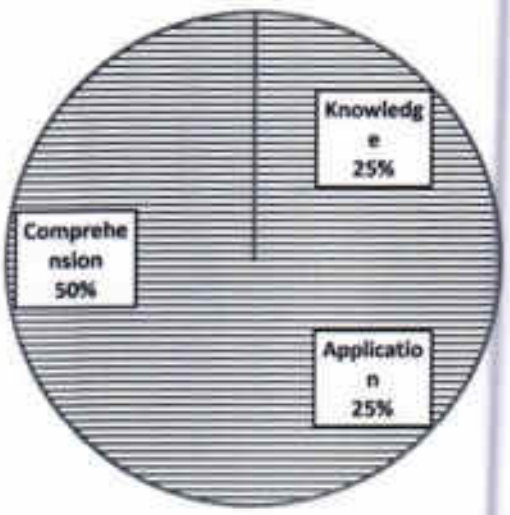
1. A) Describe in detail about Regular Expressions.&Write in detail about various special symbols and characters in Regular Expressions (C413.4) [KNOWLEDGE][5M]

2. Show how threads are accessed from python? Give an example program. (C413.5) [APPLICATION][5M]

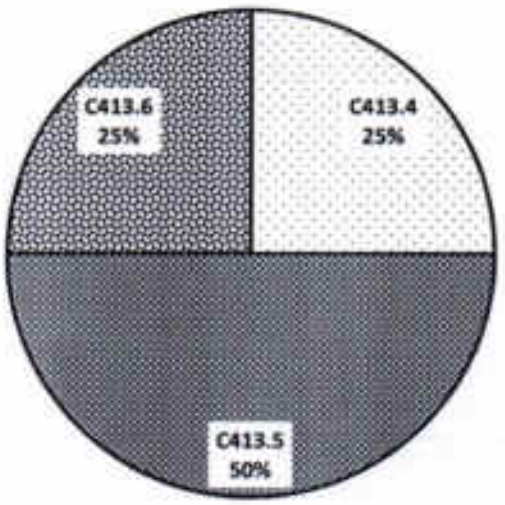
3. Explain about
   (A) TCL, Tk and Tkinter. (C413.5)[COMPREHENSION][2M]
   (B) Installing Tkinter and Working on it. (C413.5) [COMPREHENSION][3M]

4. Explain in detail about database application programmers interface. (C413.6)[COMPREHENSION][5M]

QUESTION PAPER MAPPING WITH BT'S



QUESTION PAPER MAPPING WITH CO'S



PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department Computer Science and Engineering
2020-2021;1st Semester

Mid-II

Answer key-Set-I

1. **A) Write in detail about various special symbols and characters in Regular Expressions.**

[KNOWLEDGE](C413.4)-5M

Regex Tutorial

The term **Regex** stands for **Regular expression**. The **regex** or **regexp** or **regular expression** is a sequence of different characters which describe the particular search pattern. It is also referred/called as a **Rational expression**. It is mainly used for searching and manipulating text strings. In simple words, you can easily search the pattern and replace them with the matching pattern with the help of regular expression.

This concept or tool is used in almost all the programming or scripting languages such as PHP, C, C++, Java, Perl, JavaScript, Python, Ruby, and many others. It is also used in word processors such as word which helps users for searching the text in a document, and also used in various IDEs.

The pattern defined by the regular expression is applied to the given string or a text from left to right.

You can use the regular expression (Regex) in the code of Python by importing the re module in your script. This module defines the various function or methods which are used for handling the regular expression.

The following table defines the various functions:

Methods	Description
re.match()	The **re.match()** method is used to return a string which is matched with the regular expres
re.search()	The **re.search()** method returns an object of the match when the pattern is found in a strin
re.findall()	The **re.findall()** method is used to return a string list containing all the matches

re.split()	The **re.split()** method is used to divide the string on the basis of matching with the regular
re.sub()	The **re.sub()** method is used to replace the matched string with another string.

Examples of use Regular Expression in Python

Example 1: This example helps in understanding how to use the **findall()** method in python script.

```
1. import re
2.
3. string = 'Fruits 32, Animals 80, Cars 34'
4. pattern = '\D+'
5.
6. match = re.findall(pattern, string)      #This statement is used to store the matching values on the b
   asis of a given pattern from the string.
7. print(match)  #This statement displays the values which are stored in match variable
```

1(b). Write a short notes on global interpreter lock.

Python Global Interpreter Lock (GIL) is a type of process lock which is used by python whenever it deals with processes. Generally, Python only uses only one thread to execute the set of written statements. This means that in python only one thread will be executed at a time. The performance of the single-threaded process and the multi-threaded process will be the same in python and this is because of GIL in python. We can not achieve multithreading in python because we have global interpreter lock which restricts the threads and works as a single thread.

What problem did the GIL solve for Python :

Python has something that no other language has that is a reference counter. With the help of the reference counter, we can count the total number of references that are made internally in python to assign a value to a data object. Due to this counter, we can count the references and when this count reaches to zero the variable or data object will be released automatically. For Example

```
# Python program showing

# use of reference counter

importsys
```

```
geek_var ="Geek"

print(sys.getrefcount(geek_var))

string_gfg =geek_var

print(sys.getrefcount(string_gfg))
```

2. Explain about thread module and threading module with an example program. [COMPREHENSION]C413.5) -[5M]

What is a Thread?

A **thread** of execution is the smallest sequence of programmed instructions that can be managed independently by a **scheduler** , which is typically a part of the operating system. Thread in a computer program is an execution path. Threading is creating additional **independent execution** paths in your program. Every program starts with at least one execution path/thread. You can create more threads to execute parallel tasks depending on your requirements. It is a concept of efficient **resource utilization** . Having multiple threads in an application provides two very important potential advantages:

- Improve an application's perceived responsiveness.
- Improve an application's real-time performance on multicore systems.

There are libraries to achieve it. It is important to note that Threading requires careful synchronization to avoid race conditions.

How to create a Thread in Python

The **"thread"** module provides simple functionalities and higher level interface is provided within the threading module that should be used instead. First thing you need to do is to import Thread using the following code:

```
from threading import Thread
```

A simple Thread program

The following example shows how to run a function as Threads in Python. The simplest way is via the thread module and its **start_new_thread()** method.

```
defyourFunc():
```

```
    print "Funcion called!!"

thread.start_new_thread(yourFunc, ())
```

The **thread.start_new_thread** start a new thread and return its identifier. When the function returns, the thread silently exits.

example

```
import time

from threading import Thread

defmyfunc(i):

    print ("Before Sleep :", i)

time.sleep(5)

    print ("After Sleep :", i)

for i in range(10):

    t = Thread(target=myfunc, args=(i,))

t.start()
```

Creating a Thread Class

Python provides the threading module which implements a layer on top of the **thread module** . The threading module provides, among other things, a Thread class which contains a **run() method** . Typical usage is to subclass the Thread class and override the run() method in the subclass to implement the desired functionality. In order to create a thread in Python you'll want to make your class work as a thread. For this, you should subclass your class from the **Thread class** .

```
class MyThreadClass(threading.Thread):

def run(self):
```

Here, MyThreadClass is a child class of the **Thread class** . Next step is to define a run method in this class. The run() method in the MyThreadClass is the entry point for a thread. The **run() method** will be executed when we call the start method of any object in our MyThreadClass class. You can write code inside the run() method for running thread activity. It is possible to pass a

function or other callable object to the Thread class constructor to specify the target that the run() method will call. To do this, we can use the function **thread.start_new_thread** :

```
t = MyThreadClass()

t.start()
```

What is Multithreading ?

Multithreading is a **core concept** of software programming that almost all the high-level programming languages support. **Multithreaded programs** are similar to the single-threaded programs that you have been studying. They differ only in the fact that they support more than one concurrent thread of execution-that is, they are able to **simultaneously execute** multiple sequences of instructions. These threads share the process resources but are able to execute independently. That means that a single process can have many different "functions" executing concurrently, allowing the application to better use the available hardware (multiple cores/processors). For example, a **multithreaded** operating system may run several background tasks, such as logging file changes, indexing data, and managing windows at the same time.

Python Multithreaded Programming

The following example demonstrate how a multithreaded program execute in Python.

```
import time

import threading

class myThread(threading.Thread):

def __init__(self, threadID):

threading.Thread.__init__(self)

self.threadID = threadID

def run(self):

    for x in range(1, 5):

print(" Thread ID :",self.threadID)

time.sleep(1)

print(" ")
```

```
t1 = myThread(1)

t1.start()

t1 = myThread(2)

t1.start()

t1 = myThread(3)

t1.start()

t1 = myThread(4)

t1.start()
```

3. Illustrate an introduction about GUI programming and Write how Tk is added to the applications ? (COMPREHENSION)(C413.5)-5M

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

```
import tkinter
```

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. **Tk(screenName=None, baseName=None, className='Tk', useTk=1):** To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:
   m=tkinter.Tk() where m is the name of the main window object

2. **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.
   m.mainloop()

```
importtkinter

m =tkinter.Tk()

"

widgets are added here

"

m.mainloop()
```

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer.

4.A) Summarize basic Database Operations in Python&b) Write some SQL commands in detail.EVALUATION(C413.6)-5M

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as –

- GadFly
- mSQL
- MySQL
- PostgreSQL

- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

Here is the list of available Python database interfaces: Python Database Interfaces and APIs. You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following –

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

We would learn all the concepts using MySQL, so let us talk about MySQLdb module.

What is MySQLdb?

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

How do I Install MySQLdb?

Before proceeding, you make sure you have MySQLdb installed on your machine. Just type the following in your Python script and execute it –

```
#!/usr/bin/python

importMySQLdb
```

If it produces the following result, then it means MySQLdb module is not installed –

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    import MySQLdb
ImportError: No module named MySQLdb
```

To install MySQLdb module, use the following command –

For Ubuntu, use the following command –

```
$ sudo apt-get install python-pip python-dev libmysqlclient-dev
```

For Fedora, use the following command -

$ sudodnf install python python-develmysql-develredhat-rpm-configgcc

For Python command prompt, use the following command -

pip install MySQL-python

Note – Make sure you have root privilege to install above module.

Database Connection

Before connecting to a MySQL database, make sure of the followings –

- You have created a database TESTDB.
- You have created a table EMPLOYEE in TESTDB.
- This table has fields FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.
- User ID "testuser" and password "test123" are set to access TESTDB.
- Python module MySQLdb is installed properly on your machine.
- You have gone through MySQL tutorial to understand MySQL Basics.

Example

Following is the example of connecting with MySQL database "TESTDB"

```
#!/usr/bin/python

importMySQLdb

# Open database connection
db=MySQLdb.connect("localhost","testuser","test123","TESTDB")

# prepare a cursor object using cursor() method
cursor =db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data =cursor.fetchone()
print"Database version : %s "% data

# disconnect from server
db.close()
```

While running this script, it is producing the following result in my Linux machine.

Database version : 5.0.45

If a connection is established with the datasource, then a Connection Object is returned and saved into **db** for further use, otherwise **db** is set to None. Next, **db** object is used to create a **cursor** object, which in turn is used to execute SQL queries. Finally, before coming out, it ensures that database connection is closed and resources are released.

Creating Database Table

Once a database connection is established, we are ready to create tables or records into the database tables using **execute** method of the created cursor.

Example

Let us create Database table EMPLOYEE −

```
#!/usr/bin/python

importMySQLdb

# Open database connection
db=MySQLdb.connect("localhost","testuser","test123","TESTDB")

# prepare a cursor object using cursor() method
cursor =db.cursor()

# Drop table if it already exist using execute() method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement
sql="""CREATE TABLE EMPLOYEE (
      FIRST_NAME  CHAR(20) NOT NULL,
      LAST_NAME  CHAR(20),
      AGE INT,
      SEX CHAR(1),
      INCOME FLOAT )"""

cursor.execute(sql)

# disconnect from server
db.close()
```

SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE It is used to create a new table in the database.

Syntax:

1. CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

1. CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE) ;

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

1. DROP TABLE ;

Example

1. DROP TABLE EMPLOYEE;

e. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

1. ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

1. LTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

1. ALTER TABLE MODIFY(COLUMN DEFINITION....);

EXAMPLE

1. ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

1. TRUNCATE TABLE table_name;

Example:

1. TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE

- DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

```
1. INSERT INTO TABLE_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, .... valueN);
```

Or

```
1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value3, .... valueN);
```

For example:

```
1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

```
1. UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CON
   DITION]
```

For example:

```
1. UPDATE students
2. SET User_Name = 'Sonoo'
3. WHERE Student_Id = '3'
```

c. DELETE: It is used to remove one or more row from a table.

Syntax:

```
1. DELETE FROM table_name [WHERE condition];
```

For example:

```
1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";
```

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Example

1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

b. Revoke: It is used to take back permissions from the user.

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department Computer Science and Engineering
2019-2020;1st Semester

Mid-II

Answer key-Set-II

1. **Describe in detail about Regular Expressions. & Write in detail about various special symbols and characters in Regular Expressions. (C413.4) [KNOWLEDGE][2M]**

Regex Tutorial

The term **Regex** stands for **Regular expression**. The **regex** or **regexp** or **regular expression** is a sequence of different characters which describe the particular search pattern. It is also referred/called as a **Rational expression**. It is mainly used for searching and manipulating text strings. In simple words, you can easily search the pattern and replace them with the matching pattern with the help of regular expression.

This concept or tool is used in almost all the programming or scripting languages such as PHP, C, C++, Java, Perl, JavaScript, Python, Ruby, and many others. It is also used in word processors such as word which helps users for searching the text in a document, and also used in various IDEs.

The pattern defined by the regular expression is applied to the given string or a text from left to right.

You can use the regular expression (Regex) in the code of Python by importing the re module in your script. This module defines the various function or methods which are used for handling the regular expression.

The following table defines the various functions:

Methods	Description
re.match()	The **re.match()** method is used to return a string which is matched with the regular expres
re.search()	The **re.search()** method returns an object of the match when the pattern is found in a strin
re.findall()	The **re.findall()** method is used to return a string list containing all the matches.
re.split()	The **re.split()** method is used to divide the string on the basis of matching with the regular

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam
R.R Dist. Telangana

re.sub()	The **re.sub()** method is used to replace the matched string with another string.

Examples of use Regular Expression in Python

Example 1: This example helps in understanding how to use the **findall()** method in python script.

```
8. import re
9.
10. string = 'Fruits 32, Animals 80, Cars 34'
11. pattern = '\D+'
12.
13. match = re.findall(pattern, string)      #This statement is used to store the matching values on the b
    asis of a given pattern from the string.
14. print(match)   #This statement displays the values which are stored in match variable
```

1(b). Write a short notes on global interpreter lock.

Python Global Interpreter Lock (GIL) is a type of process lock which is used by python whenever it deals with processes. Generally, Python only uses only one thread to execute the set of written statements. This means that in python only one thread will be executed at a time. The performance of the single-threaded process and the multi-threaded process will be the same in python and this is because of GIL in python. We can not achieve multithreading in python because we have global interpreter lock which restricts the threads and works as a single thread.

What problem did the GIL solve for Python :

Python has something that no other language has that is a reference counter. With the help of the reference counter, we can count the total number of references that are made internally in python to assign a value to a data object. Due to this counter, we can count the references and when this count reaches to zero the variable or data object will be released automatically. For Example

```
# Python program showing

# use of reference counter

import sys

geek_var = "Geek"

print(sys.getrefcount(geek_var))

string_gfg = geek_var

print(sys.getrefcount(string_gfg))
```

2. Show how threads are accessed from python? Give an example program.

[APPLICATION](C413.5)[5M]

A thread is a separate flow of execution. This means that your program will have two things happening at once. But for most Python 3 implementations the different threads do not actually execute at the same time: they merely appear to.

It's tempting to think of threading as having two (or more) different processors running on your program, each one doing an independent task at the same time. That's almost right. The threads may be running on different processors, but they will only be running one at a time.

Getting multiple tasks running simultaneously requires a non-standard implementation of Python, writing some of your code in a different language, or using multiprocessing which comes with some extra overhead.

Because of the way CPython implementation of Python works, threading may not speed up all tasks. This is due to interactions with the GIL that essentially limit one Python thread to run at a time.

Tasks that spend much of their time waiting for external events are generally good candidates for threading. Problems that require heavy CPU computation and spend little time waiting for external events might not run faster at all.

This is true for code written in Python and running on the standard CPython implementation. If your threads are written in C they have the ability to release the GIL and run concurrently. If you

are running on a different Python implementation, check with the documentation too see how it handles threads.

If you are running a standard Python implementation, writing in only Python, and have a CPU-bound problem, you should check out the multiprocessing module instead.

Architecting your program to use threading can also provide gains in design clarity. Most of the examples you'll learn about in this tutorial are not necessarily going to run faster because they use threads. Using threading in them helps to make the design cleaner and easier to reason about.

So, let's stop talking about threading and start using it!

Starting a Thread

Now that you've got an idea of what a thread is, let's learn how to make one. The Python standard library provides threading, which contains most of the primitives you'll see in this article. Thread, in this module, nicely encapsulates threads, providing a clean interface to work with them.

To start a separate thread, you create a Thread instance and then tell it to .start():

```
1importlogging
2importthreading
3importtime
4
5defthread_function(name):
6logging.info("Thread %s: starting",name)
7time.sleep(2)
8logging.info("Thread %s: finishing",name)
9
10if__name__=="__main__":
11format="%(asctime)s: %(message)s"
12logging.basicConfig(format=format,level=logging.INFO,
13datefmt="%H:%M:%S")
14
15logging.info("Main    : before creating thread")
16x=threading.Thread(target=thread_function,args=(1,))
17logging.info("Main    : before running thread")
18x.start()
19logging.info("Main    : wait for the thread to finish")
20# x.join()
21logging.info("Main    : all done")
```

If you look around the logging statements, you can see that the main section is creating and starting the thread:

```
x=threading.Thread(target=thread_function,args=(1,))
x.start()
```

When you create a Thread, you pass it a function and a list containing the arguments to that function. In this case, you're telling the Thread to run thread_function() and to pass it 1 as an argument.

For this article, you'll use sequential integers as names for your threads. There is threading.get_ident(), which returns a unique name for each thread, but these are usually neither short nor easily readable.

thread_function() itself doesn't do much. It simply logs some messages with a time.sleep() in between them.

When you run this program as it is (with line twenty commented out), the output will look like this:

```
$ ./single_thread.py
Main    : before creating thread
Main    : before running thread
Thread 1: starting
Main    : wait for the thread to finish
Main    : all done
Thread 1: finishing
```

You'll notice that the Thread finished after the Main section of your code did. You'll come back to why that is and talk about the mysterious line twenty in the next section.

3.Explain about (i) TCL, Tk and Tkinter. [COMPREHENSION](C413.5)[[2M]

(ii) Installing Tkinter and Working on it.[COMPREHENSION](C413.5) [3M]

What are Tcl, Tk, and Tkinter?

Let's try to understand more about the Tkinter module by discussing more about it origin.

- As mentioned, Tkinter is **Python's default GUI library**, which is nothing but a wrapper module on top of the **Tk toolkit**.
- Tkinter is based upon the Tk toolkit, and which was originally designed for the **Tool Command Language** (Tcl). As Tk is very popular thus it has been ported to a **variety of other scripting languages**, including **Perl (Perl/Tk)**, **Ruby (Ruby/Tk)**, and **Python (Tkinter)**.

- **GUI development portability** and **flexibility of Tk** makes it the right tool which can be used to design and **implement a wide variety of commercial-quality GUI applications.**
- Python **with Tkinter** provides us a **faster and efficient way** in order to build useful applications that would have taken much time if you had to program directly in C/C++ with the help of native OS system libraries.
- Once we have Tkinter up and running, we'll use basic building blocks known as **widgets** to create a variety of desktop applications.

Install Tkinter

Chances are, that Tkinter may be already installed on your system along with Python. But it is not true always. So let's first check if it is available.

If you do not have Python installed on your system - Install Python 3.8 first, and then check for Tkinter.

You can determine **whether Tkinter is available** for your Python interpreter by attempting to **import the Tkinter module** - If Tkinter is available, then there will be no errors, as demonstrated in the following code:

```
importtkinter
```

Nothing exploded, so we know we have **Tkinter available**. If you see any error like module not found, etc, then your **Python interpreter was not compiled with Tkinter enabled**, the module **import fails** and you might need to recompile your **Python interpreter to gain access to Tkinter**.

Adding Tk to your Applications

Basic steps of setting up a GUI application using Tkinter in Python are as follows:

1. First of all, **import the Tkinter module**.
2. The second step is to **create a top-level windowing object** that contains your entire GUI application.
3. Then in the third step, you need to **set up all your GUI components** and their functionality.
4. Then you need to **connect these GUI components** to the underlying application code.
5. Then just enter the main event loop using mainloop()

The above steps may sound gibberish right now. But just read them all, and we will explain everything as we move on with this tutorial.

First Tkinter Example

As mentioned earlier that in GUI programming all main widgets are only built on the top-level window object.

The top-level window object is created by the Tk class in Tkinter.

Let us create a top-level window:

```
importtkinterastk

win =tk.Tk()

win.mainloop()
```

kinter Methods used above:

The two main methods are used while creating the **Python application** with **GUI**. You must need to remember them and these are given below:

1. **Tk(screenName=None, baseName=None, className='Tk', useTk=1)**

This method is mainly used **to create the main window**. You can also change the **name of the window** if you want, just by changing the **className** to the desired one.

The code used to create the main window of the application is and we have also used it in our above example:


```
win =tkinter.Tk()
```

Copy

2. The mainloop() Function

This method is used to start the application. The mainloop() function is an **infinite loop** which is used to run the application, it will wait for **an event to occur** and **process the event** as long as the window is not closed.

4.Explain in detail about database application programmers interface.

[COMPREHENSION](C413.6)[5M]

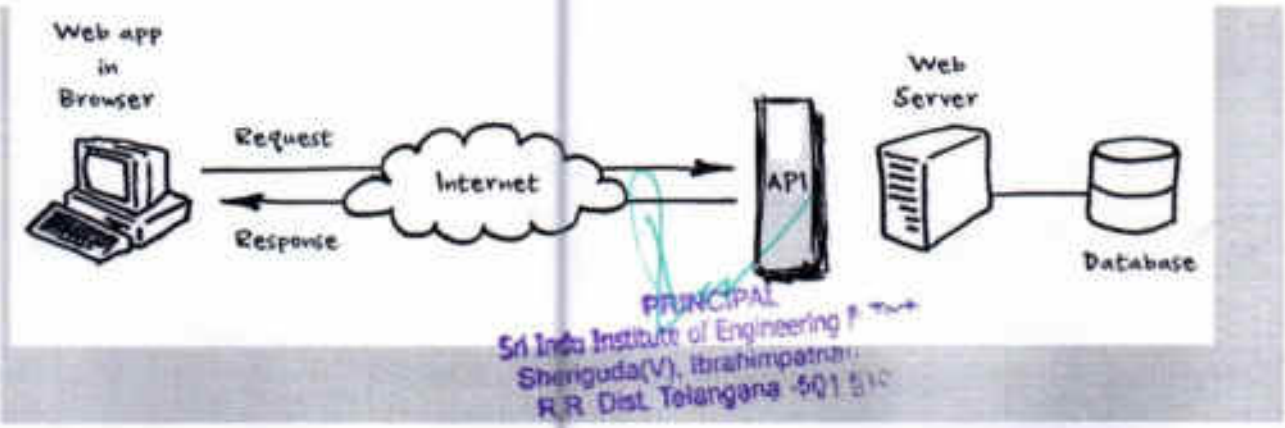
What is Database?

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The **database management system (DBMS)** is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Now Lets understand the meaning of API

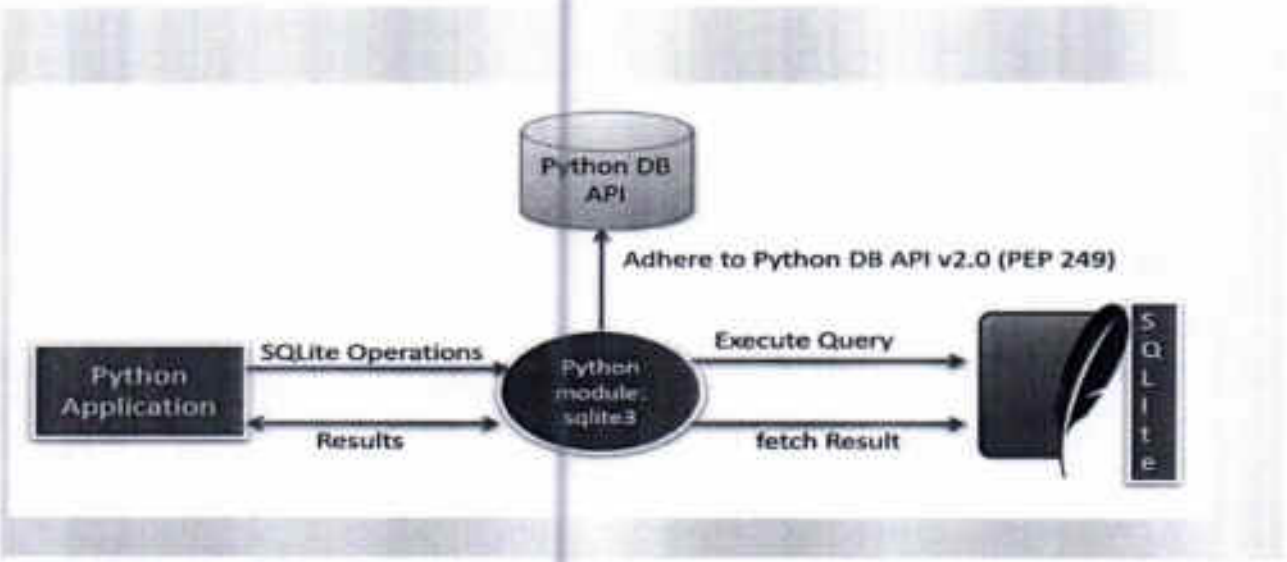
Meaning of API (Application Programming Interface)



(**A**pplication **P**rogramming **I**nterface) API is a language and message format used by an application program to communicate with the operating system or some other control program such as a database management system (DBMS) or communications protocol. APIs are implemented by writing function calls in the program, which provide the linkage to the required subroutine for execution.

Thus, an API implies that a driver or program module is available in the computer to perform the operation or that software must be linked into the existing program to perform the tasks.

DB-API 2.0 interface for SQLite databases



SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

The sqlite3 module was written by Gerhard Häring. It provides a SQL interface compliant with the DB-API 2.0 specification described by **PEP 249**. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform — you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures.

SQLite natively supports the following types: NULL, INTEGER, REAL, TEXT, BLOB. The following Python types can thus be sent to SQLite without any problem:

Python type	SQLite type
None	NULL
int	INTEGER
float	REAL
str	TEXT
bytes	BLOB

Concepts of Python DB-API

1. **Connection Objects** — Database Connections and Manage Transactions

cursor(*factory=Cursor*)-The cursor method accepts a single optional parameter *factory*. If supplied, this must be a callable returning an instance of Cursor or its subclasses.

commit()-This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections. If you wonder why you don't see the data you've written to the database, please check you didn't forget to call this method.

rollback()-This method rolls back any changes to the database since the last call to commit().

close()-This closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost!

2. **Cursor Objects** — Database Queries

execute(*sql*[, *parameters*])-Executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals).

executemany(*sql, seq_of_parameters*)-Executes an SQL command against all parameter sequences or mappings found in the sequence *seq_of_parameters*. The sqlite3 module also allows using an iterator yielding parameters instead of a sequence.

fetchone()-Fetches the next row of a query result set, returning a single sequence, or None when no more data is available.

fetchmany(*size=cursor.arraysize*)-Fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available.

The **number of rows to fetch per call** is specified by the *size* parameter. If it is not given, the cursor's arraysize determines the number of rows to be fetched. The method should try to fetch as many rows as indicated by the size parameter. If this is not possible due to the specified number of rows not being available, fewer rows may be returned.

Note there are **performance considerations** involved with the *size* parameter. For optimal performance, it is usually best to use the arraysize attribute. If the *size* parameter is used, then it is best for it to retain the same value from one fetchmany() call to the next.

fetchall()-Fetches all (remaining) rows of a query result, returning a list. Note that the cursor's arraysize attribute can affect the performance of this operation. An empty list is returned when no rows are available.

close()-Close the cursor now (rather than whenever __del__ is called).

arraysize-Read/write attribute that controls the number of rows returned by fetchmany(). The default value is 1 which means a single row would be fetched per call.

STEPS with different operations used on DB:

1. **Import sqlite3 and connect with a database**

Creating a Connection object that represents the database. Here the data will be stored in the example.db file:

```
In [1]: import sqlite3
        conn = sqlite3.connect('example.db')
```

2. **Create a Table**

Creating a Cursor object and a table name **material** which contains date , trans, symbol, qty and price attributes.

```
In [10]: c = conn.cursor()
         # Create table
         c.execute('''CREATE TABLE material
                      (date text, trans text, symbol text, qty real,

Out[10]: <sqlite3.Cursor at 0x1338822c3b0>
```

3. **Insert command**

call its execute() method to perform SQL command -execute.

```
In [15]: # Insert a row of data
         c.execute("INSERT INTO material VALUES ('2006-01-05','BUY'
```

```
Out[15]: <sqlite3.Cursor at 0x1338822c8f0>
```

4. **Reading the Data**

```
In [21]: records = conn.execute('''SELECT * FROM material''')
```

```
In [22]: for data in records:
             print(data)
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2008-05-10', 'xyz', 'HAT', 400.0, 212.5)
```

5. **Update the Records (Optional)**

Now consider that qty changed to 500 on 2008–05–10

```
In [28]: conn.execute('''UPDATE material SET qty=500 WHERE date = "2(
         # Save (commit) the changes
         conn.commit()
```

```
In [29]: records = conn.execute('''SELECT * FROM material''')
         for data in records:
             print(data)
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2008-05-10', 'xyz', 'HAT', 500.0, 212.5)
```


6. **Delete a record from a table (USE it carefully)**

Lets delete the record of date — 2006-01-05

```
In [30]: conn.execute('''Delete from material where date = "2006-01-0
         conn.commit()
```

```
In [31]: cursor = conn.cursor()
         data = cursor.execute('''SELECT * FROM material''')
         for data in data:
             print(data)
```

```
('2008-05-10', 'xyz', 'HAT', 500.0, 212.5)
```

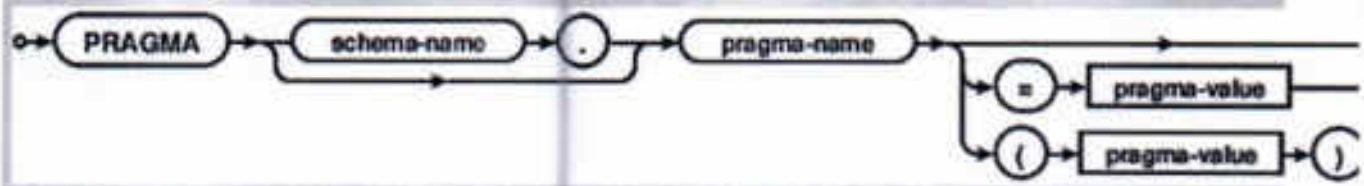
7. **Closing the Connection**

```
# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

How to see the Schema of a Table — Using PRAGMA statement

The PRAGMA statement is an SQL extension specific to SQLite and used to modify the operation of the SQLite library or to query the SQLite library for internal (non-table) data.

PRAGMA command syntax



The PRAGMA statement is issued using the same interface as other SQLite commands (e.g. SELECT, INSERT) but is different in the following important respects:

- The pragma command is specific to SQLite and is not compatible with any other SQL database engine.
- Specific pragma statements may be removed and others added in future releases of SQLite. There is no guarantee of backwards compatibility.

- No error messages are generated if an unknown pragma is issued. Unknown pragmas are simply ignored. This means if there is a typo in a pragma statement the library does not inform the user of the fact.

```
In [32]: Schema = cursor.execute('''PRAGMA table_info("material")''')

         for r in Schema:
             print(r)
```

```
(0, 'date', 'text', 0, None, 0)
(1, 'trans', 'text', 0, None, 0)
(2, 'symbol', 'text', 0, None, 0)
(3, 'qty', 'real', 0, None, 0)
(4, 'price', 'real', 0, None, 0)
```

Benefits of the Python for Database Programming

1. Python is very popular scripting language for connected with databases.
2. Python ecosystem is very rich and it is easy to use tools for datascience.
3. Due to open source nature, Python is portable for many platforms.
4. Python support Relational Database System.
5. Writing Python code to access database API commands which refers to as DB-API.

Assignment Questions-I
(Assignment Questions are mapped with CO's, BT)

1. Describe Numbers in python and its built-in functions.**(Knowledge)(C413.1)**
2. Write about Lists and Tuples with some operations.**(Knowledge) (C413.2)**
3. Explain python files and related modules.**(Comprehension) (C413.2)**
4. Summarize Exception Handling in Python.**(Evaluation)(C413.3)**
5. Explain Python Threads.**(Comprehension) (C413.4)**

Assignment Questions-II
(Assignment Questions are mapped with CO's, BT)

1. What is Regular Expression in Python? Describe special characters and symbols in Regular Expression.**(Synthesis)(C413.4)**
2. Explain about RE methods search(), match(), findall(), split(), sub() with example programs. **(Comprehension)(C413.5)**
3. Describe about Global Interpreter Lock in python? Write a simple program on it.**(Knowledge)(C413.5)**
4. Write about Tkinter in GUI programming.**(Knowledge)(C413.6)**
5. Explain Database Application Programmer's Interface (DB-API) in Python.**(Comprehension)(C413.6)**

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 5

Python Programming

B. Pooja
CSE-A
17X31A0518

1. What is Regular Expression in Python? Describe special characters and symbols in Regular expression?

A. Regular Expression:

A Regular Expression (RegEx) is a sequence of characters that defines a particular pattern. It is also known as RegEx in python is a sequence of metacharacters, numbers and alphabets that define a particular pattern.

* Regular Expression are used to identify whether a pattern exists in the given string or not.
* Regular Expression are used in search algorithms, search and replace dialogs of text editors, and in lexical analysis. It is also used for input validation.
* re module in python supports Regular Expressions Special characters (or) Metacharacters supported by the submodule.

[ ] . ^ $ * ? [} ( ) \ |

① [ ] :- Square brackets

→ Square brackets specifies a set of characters you wish to match.
→ checks if a string contains any one of the characters given within it.

Expression	String	Matched?
[a b c]	a	1 match
	ac	2 matches
	Hey jude	No match
	abc de ca	5 matches

You can also specify a range of characters using inside square brackets

- [a-e] is the same as [abcde]
- [1-4] is the same as [1234]
- [0-39] is the same as [01239]

② . - Period

A period matches any single character (except newline '\n')

Expression	string	Matched?
. .	a	No match
	ac	1 match
	acd	1 match
	acde	2 matches (contain 4 characters)

③ ^ - caret

The caret symbol ^ is used to check if a string starts with a certain character.

Expression	string	Matched?
^a	a	1 match
	abc	1 match
	bac	No match
^ab	abc	1 match
	acb	No match (starts with a but not followed by b)

④ $ - Dollar

The dollar symbol $ is used to check if a string ends with a certain character.

Expression	String	Matched?
a$	a	1 match
	formula	1 match
	cab	No match

⑤ * - star

The star symbol * matches zero or more occurrences of the pattern left to it.

Expression	String	Matched?
ma*n	mn	1 match
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

⑥ + : plus

The plus symbol (+) matches one or more occurrences of the pattern left to it

Expression	String	Matched
ma+n	mn	No match (no character)
	man	1 match
	maan	1 match
	main	No match (a is not followed by n)
	woman	1 match

⑦ ? - Question mark

The question mark symbol(?) matches zero or one occurrence of the pattern left to it.

Expression	String	Matched?
	mn	1 match
ma?n	man	1 match
	maaan	No match(more than one a character)
	main	No match(a is not followed by n)
	woman	1 match

⑧ ( ) :- used for grouping different symbols of RE acts as a single block.

⑨ {n} :- specifies the preceding element to be matched exactly n times.

Special symbols in Regular Expression:

Special sequences make commonly used patterns easier to write. A special sequence is a \ followed by one of the characters in the list below, and has it a special meaning.

① \A - Matches if the specified characters are at the start of a string

Expression	String	Matched?
\Athe	the sun	Match
	In the sun	No match

③ \b: Matches if the specified characters are at the beginning or end of a word.

Expression	String	Matched?
\bfoo	football	Match
	a football	Match
	a football	No match

④ \B - opposite of \b. Matches if the specified characters are not at beginning or end of a word.

Expression	String	Matched?
\Bfoo	football	No Match
	a football	No Match
	afootball	Match
foo\B	the a foo test	No match
	the a footest	Match

⑤ \d - Matches any decimal digit. Equivalent to [0-9]

Expression	String	Matched?
\d	12abc3	3 matches (at 12abc3)
	python	No match

⑤ \D - Matches any non decimal digit. Equivalent to [^0-9]

Expression	String	Matched?
\D	1ab34"50	3 matches (at 1ab34"50)
	1345	No match

⑥ \S :- Matches where a string contains any whitespace character. Equivalent to [\t\n\r\f\v]

Expression	String	Matched!
\S	Python RegEx	1 match
	Python RegEx	No match

⑦ \S - Matches where a string contains any non-whitespace character. Equivalent to ['\t\n\r\f\v].

Expression	String	Matched?
\S	ab	2 matches (at a b)
	▭	No match

⑧ \w - matches any single, alphabet, number, underscore

⑨ \W :- Matches any one-alphanumeric character. Equivalent to [^a-zA-z0-9_]

⑩ \Z :- Matches if the specified characters are at the end of a string.

2. Explain about RE methods search(), match(), findall(), split(), sub() with example programs?

A. ReModule Functions In Python:

Python has a module named re to work with Regular expressions. To use it, we need to import the module.

* import re

The Module defines several functions and constant to work with RegEx.

1. re.search()

The search function searches for first occurance of RE pattern within string with optional flags.

Syntax: match = re.search (pattern, str)

The re.search() method takes two arguments: a pattern and a string. The method looks for the first location where the RegEx pattern produces a match with the string. If the search is successful, re.search() returns a match object; if not, it returns None.

Pgm

```
import re
string = "Python is fun"
match = re.search ('\APython', string) # check if python is
                                         at the beginning
```

```
if match:
  print("pattern found inside the string")
else:
  print("pattern not found")
```

O/p: pattern found inside the string.

Match object

You can get methods and attributes of a match object using dir() function.

Some of the commonly used methods and attributes of match objects are:

match.group()

The group() method returns the part of the string where there is a match.

Ex:
```
import re
  String = '39801 356,2102 1111'
# Three digit number followed by space followed by two digit number.
  pattern = '(\d{3}) (\d{2})'
# match variable contains a match object
  match = re.search(pattern, string)
if match:
```

```
    print(match.group())
else:
    print("pattern not found")
```

O/P: 801 35

→ match.start(), match.end() and match.span(). The start() function returns the index of the start of the matched substring. Similarly, end() returns the end index of the matched substring.
The span() function returns a tuple containing start and end index of matched part.

3. findall()

The re.findall() method returns a list of strings containing all matches.

Ex1 : re.findall()

```
#program to extract numbers from a string
import re
string = 'hello 12 hi 89. Howdy 34'
pattern = '\d+'
result = re.findall(pattern, string)
print(result)
```

O/p: ['12', '89', '34']

If the pattern is not found, re.findall() returns an empty list.

4. split():

The re.split() method splits the string where there is a match and returns a list of strings where the splits have occurred.

Ex:

```
import re
string = 'Twelve:12 Eighty nine:89.'
pattern = '\d+'
result = re.split (pattern, string)
print(result)
```

o/p : ['Twelve:', 'Eighty nine:', '.']

If the pattern is not found, re.split() returns a list containing the original string.

5. sub()

The syntax of re.sub() is

re.sub(pattern, replace, string)

→ It searches for a particular pattern in a string and replaces it with a new pattern

re.sub(regex, replacement, target string)

Ex :

```
import re
str = "Analytical Engine was invented in the year 1837".
m = re.sub("Analytical Engine","Electrical telegraph", str)
print(m)
```

o/p : Electrical telegraph was invented in the year 1837.

3. Describe about Global Interpreter lock in python? Write a simple program on it?

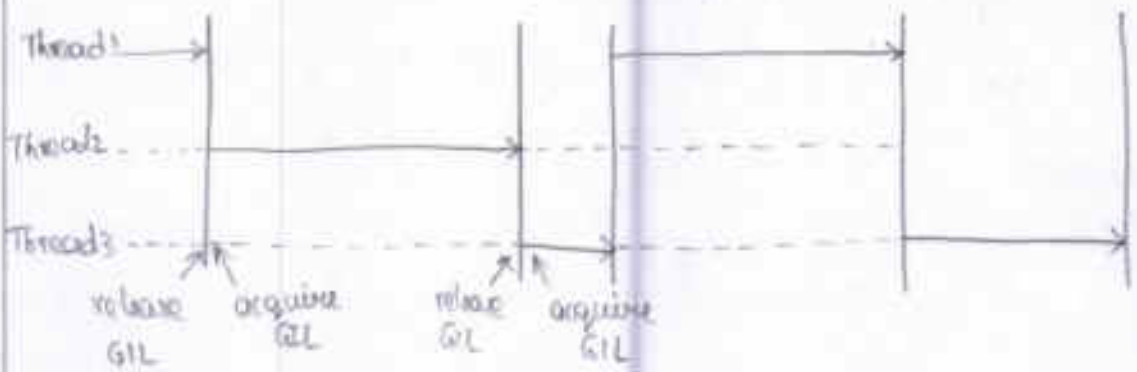
A. The python Global Interpreter lock or GIL in simple words, is a mutex (or a lock) that allows only one thread to hold the control of the python interpreter ie, only one thread can be in a state of execution at any point of time.

→ It is a performance bottleneck in CPU bound and multi-threaded code.

→ Each thread that wants to run must wait for the ~~GIT~~ GIL to be released by the other thread which means multi-threaded python application is actually single threaded.

→ The GIL prevents simultaneous access to python objects by multiple threads

→ Python uses a global lock on each interpreter process which means that every process treats the python interpreter itself as a resource.



→ GIL does not prevent a process form running on a different processor of a machine. It simply allows one thread to run once within the interpreter.

So Multiprocessing will allow to achieve true concurrency.
The Cpython garbage collector uses an efficient memory management technique known as reference counting.
The reference count variable is prove to race condition like any other global variable. To solve this problem, the developers of python decided to use the global Interpreter lock.
python supports 2 modules for multi threading
(i) threading module
(ii) thread module

Syntax:
thread.start_new_thread (function, args(, kwargs))
This method starts a new thread and returns its identifier. It will invoke the function specified as the "function" parameter with the passed list of arguments when the <function> returns, the thread would silently exit.
- args is the tuple of arguments
- The optional <KW args> arguments specify the dictionary of keyword arguments.

Pgm
Ex:
```
from _thread import start_new_thread
from time import sleep
threadId = 1
waiting = 2
def factorial(n):
    global threadId
```

```
    rc=0
    if n<1:
        print("{}:{}".format('In Thread', threadId))
    threadId+=1
        rc=1
    else:
        num=n*factorial(n-1)
        print("{}!={}".format(str(n),str(num)))
        rc=num
    return rc
start_new_thread(factorial,(5,))
start_new_thread(factorial,(4,))
print("waiting for threads to return...")
sleep(waiting)
```

<u>O/P</u>:

```
Thread:1
    1!=1
    2!=2
    3!=6
    4!=24
    5!=120
Thread:2
    1!=1
    2!=2
    3!=6
    4!=24
```

4. Write about Tkinter in GUI programming?

A. Tkinter Programming

Tk inter is the standard GUI library for python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI Toolkit.

→ It is an inbuilt python module used to create simple GUI apps

→ Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps

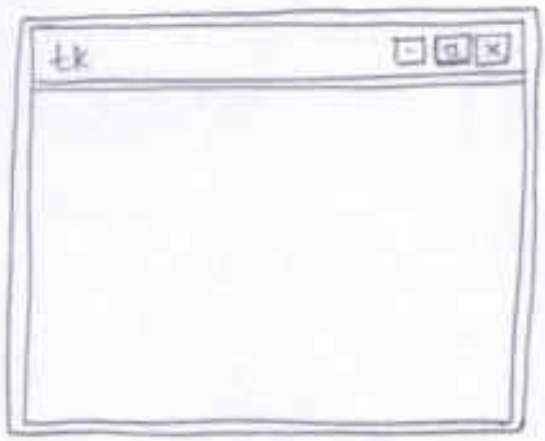
- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above mentioned widgets to the GUI application, widgets like labels etc.
- Enter the main event loop to take action against each event triggered by the user.

→ Python's Tkinter module contains Tk class. Its object forms a top level window. other widgets such as labels, button, Entry etc can be placed on this top level window.

Ex: #! /usr/bin/python

```
import Tkinter
top = Tkinter.Tk() # creating the application main window
top.mainloop()  # entering the event main loop
```

This would create a following window



Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and Text boxes used in a GUI application. These controls are commonly called widgets. There are currently 15 types of widgets in Tkinter. we present these widgets as well as a brief description in the following table.

Sr.No	operator & Description
1	Button The Button widget is used to display buttons in your application.
2	canvas The canvas widget is used to draw shapes such as lines, ovals, polygons and Rectangles in your App^n.
3	check button: The check button widget is used to display a number of options as check boxes. The user can select multiple options at a time.
4	Entry: The Entry widget is used to display a single-line text field for accepting values from a user.

5 Frame:
The frame widget is used as a container widget to organize other widgets.

6 Label:
The label widget is used to provide a single-line caption for other widgets. It can also contain images.

7 List box:
The list box widget is used to provide a list of options to a user

8 Menubutton:
The Menubutton widget is used to display menus in your application

9 Menu:
The menu widget is used to provide various commands to a user. These commands are contained inside menubutton.

10 Message:
The message widget is used to display a multiline text fields for accepting values from a user.

Geometry Management:

All Tkinter widget have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: Pack, grid and place.

- The pack() method: This Geometry manager organizes widgets in blocks before placing them in the parent widget.
- The grid() method: This Geometry manager organizes widgets in a table-like structure in the parent widget.
- The place() method: This Geometry manager organizes widgets by placing them in a specific position in the parent widget.

5. Explain Database Application Programmer's Interface (DB-API) in python?

A. DB-API (SQL-API) for Python:

Python DB-API is independent of any database engine, which enables you to write python scripts to access any database engine.

The python DB API implementation for mySql is mysqldb

Python's DBAPI consists of connection objects, cursor objects, standard exceptions and some other module contents.

Connection objects:

Connection objects create a connection with the database & these are further used for different transactions. These connection objects are used as representatives of the database session. A connection is created as follows:

```
conn = mysqldb.connect('library', user='sys', password='admin')
```

connection object can be used for calling methods like commit(), rollback() and close().

Cursor objects

cursor is one of the powerful features of SQL. These are objects responsible for submitting various SQL statements to a database server. There are several cursor classes in MYSQLdb cursors:

1. Base cursor: Base cursor is the base class for cursor objects
2. cursor is the default cursor class.

CursorWarningMixIn, CursorStoreResultMixIn, CursorTupleRowMixIn, and BaseCursor are some components of the cursor class.

3. CursorStoreResultMixIn uses the mysql_store_result() function to retrieve result sets from the executed query. These result sets are stored at client side.
4. CursorUseResultMixIn uses the mysql_use_result() function to retrieve result sets from the executed Query. These results are stored at server side.

* The following example illustrates the execution of SQL commands using cursor objects. you can use execute to execute SQL commands like SELECT. To commit all SQL operations you need to close the cursor at cursor.close

```
>>> cursor.execute('SELECT * FROM books')
>>> cursor.execute('SELECT * FROM books WHERE bookname = 'python' AND
                    book_author = 'Mark Lutz')
```

>>> cursor.close()

Error and exception handling in DB-API

Exception handling is very easy in the python DB-API module. We can place warnings and error handling messages in the programs. Python DB-API has various options to handle this, like:
Warning, Interface Error, Database Error, Integrity Error, Internal Error, Not Supported Error, Operational Error and Programming Error.

1. Integrity Error: Let's look at integrity error in detail. In the following example, we will try to enter duplicate records in the database. It will shown an integrity error.
_mysql_exceptions.IntegrityError.

2. Operational Error: If there are any operation errors like no databases selected, python DB-API will handle this error as operational Error.

3. Programming Error: If there are any programming errors like duplicate database creations, python DB-API will handle this error as programming Error.

4. Warning: WarningError is used for non-fatal issues must subclass standard Error.

5. Error: Base class for errors. Must subclass standardError.

6. Interface Error: used for errors in the database module, not the database itself. Must subclass Error.

7. Database Error: Subclass of Database Error that refers to errors in the data like division by zero.

8. Internal Error: Subclass of Database Error that refers to errors internal to the database module such as a cursor no longer being active.

9. Not Supported Error: Subclass of Database Error that refers to trying to call unsupported functionality.

2.2.1: Assess Learning Levels, Special programs for adv& Slow learners

Result Analysis:

Course Title	PYTHON PROGRAMMING
Course Code	CS721PE
Programme	B.Tech
Year & Semester	IV Year I-Semester,A,B&C sec
Regulation	R16
Course Faculty	Mrs,N.SHILPA Assistant Professor , CSE

Weak Students:

S No	Roll no	No of backlogs	Internal-I Status	Internal-II Status
1	17X31A0523	2	18	19
2	17X31A0524	3	20	15
3	17X31A0533	5	14	14
4	17X31A0554	3	18	17
5	17X31A0557	2	14	17
	17X31A0559	2	21	20

Advanced learners:

S No	Roll No	(SGPA)	Gate Material
1	17X31A0501	6.29	
2	17X31A0505	6.70	
3	17X31A0506	6.54	
4	17X31A0510	6.37	
5	17X31A0511	6.79	
6	17X31A0520	7.33	

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam,
R.R. Dist. Telangana -501 510

GATE MATERIAL

1. What will be the output of the following code (b)
print type(type(int))

(A) type 'int'
(B) type 'type'
(C) Error
(D) 0

2.What is the output of the following code (C)

print 9//2

a)4.5

b)4

c)4

d)error

3. Which operator is overloaded by the or() function?(B)

(A) ||

(B) |

(C) //

(D) /

4. Given a function that does not return any value, what value is shown when executed at the shell?(d)

(A) int

(B) bool

(C) void

(D) None

5.What is the output of the following program (B)

Print 0.1+0.2==0.3

(A) True

(B) False

(C) Machine dependent

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana-501 510

(D) Error

6. Which of these is not a core data type?(d)

(A) Lists

(B) Dictionary

(C) Tuples

(D) Class

7. What data type is the object below ?(A)

L = [1, 23, 'hello', 1]

(A) List

(B) Dictionary

(C) Tuple

(D) Array

8. Which of the following function convert a string to a float in python?(C)

(A) int(x [,base])

(B) long(x [,base])

(C) float(x)

(D) str(x)

9.Find the output of the following program(D)

```
nameList =['Harsh', 'Pratik', 'Bob', 'Dhruv']
pos =nameList.index("GeeksforGeeks")

print(pos *3)
```

(A) GeeksforGeeksGeeksforGeeksGeeksforGeeks
(B) Harsh
(C) Harsh HarshHarsh
(D) ValueError: 'GeeksforGeeks' is not in list

10.Find the output of the following program(D)

```
a ={i: i *i fori inrange(6)}

print(a)
```

(A) Dictionary comprehension doesn't exist

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam
R.R Dist. Telangana -501 5

(B) {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6:36}
(C) {0: 0, 1: 1, 4: 4, 9: 9, 16: 16, 25: 25}
(D) {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science And Engineering

Course Outcome Attainment (Internal Examination-1)

Name of the faculty : Mr.s.Prudhviraj
Academic Year: 2020-21
Branch & Section: CSE-A
Examination: I Internal
Course Name: Python Programming
Year: IV
Semester: I

S.No	HT No.	Q1a	Q1b	Q2a	Q2b	Q3		Q4a	Q4b	Obj1	A1
Max. Marks ⟹		2.5	2.5	2.5	2.5	5		3	2	10	5
1	17X31A0501	3	2			1	1			7	5
2	17X31A0502	3	2			2	1			7	5
3	17X31A0503	3	2			2	1			9	5
4	17X31A0505	2.5	2			1.5	2			9	5
5	17X31A0506	3	2			2	2			9	5
6	17X31A0508	3	2						0.5	8	5
7	17X31A0509	3	2					2	3	9	5
8	17X31A0510	3	2					2	2	9	5
9	17X31A0511	3	2					2	2	9	5
10	17X31A0513	1	2					2	2	8	5
11	17X31A0515	3	2			1	2			8	5
12	17X31A0516	2	2							8	5
13	17X31A0518	3	2					2	3	8	5
14	17X31A0520	3	2			2	3			8	5
15	17X31A0521	3	2			2	2			9	5
16	17X31A0522	3	2			2	3			8	5
17	17X31A0523	1	1				2			8	5
18	17X31A0524	3	2				2			8	5
19	17X31A0525	2	1			1.5	2.5			8	5
20	17X31A0526	3	2			2	2			8	5
21	17X31A0528	2	1			1	2			8	5
22	17X31A0529	1.5	1.5			2				8	5
23	17X31A0531	1.5	1.5			1	1			8	5
24	17X31A0532	3	2					2	2	8	5
25	17X31A0533									0	5
26	17X31A0534	2.5	1.5					1.5	2.5	7	5
27	17X31A0538	2	2			2	1			7	5
28	17X31A0539	3	2			2	3			8	5
29	17X31A0542									0	5
30	17X31A0545	3	1			1	2			9	5
31	17X31A0546	2	2					1	2	9	5
32	17X31A0547	2	2			1.5	1.5			8	5
33	17X31A0548	3	2			1.5	2			7	5
34	17X31A0549	2	2			0.5	2.5			9	5
35	17X31A0550	2	2			1.5	2.5			9	5
36	17X31A0552	2	2					1		7	5
37	17X31A0553	1.5	1.5			1	2			7	5
38	17X31A0554	1.5	2				1.5			8	5
39	17X31A0555	2	2					1	1	8	5
40	17X31A0556	2	2			2	3			8	5

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam(M)
R.R Dist. Telangana -501 510

41	17X31A0557	2	2			1	2			2	5
42	17X31A0558	1.5	1.5					1.5	1.5	8	5
43	17X31A0559	3	2					1.5	2.5	7	5
Target set by the faculty / HoD		1.50	1.50	1.50	1.50	3.00	0.00	1.80	1.20	6.00	3.00
Number of students performed above the target		39	37	0	0	0	26	6	10	40	43
Number of students attempted		41	41	0	0	24	26	12	12	43	43
Percentage of students scored more than target		95%	90%			0%	100%	50%	83%	93%	100%

CO Mapping with Exam Questions:

CO - 1	Y	Y							Y	Y
CO - 2			Y						Y	Y
CO - 3					Y	Y			Y	Y

PRINCIPAL
Sri Indu Institute of Engineering & Te...
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

CO - 4							Y	Y		
CO - 5										
CO - 6										

CO Attainment based on Exam Questions:

CO - 1	95%	90%							93%	100%
CO - 2									93%	100%
CO - 3					0%	100%			93%	100%
CO - 4							50%	83%		
CO - 5										
CO - 6										

CO	Subj	obj	Asgn	Overall	Level
CO-1	93%	93%	100%	95%	3.00
CO-2		93%	100%	97%	3.00
CO-3	50%	93%	100%	81%	3.00
CO-4	67%			67%	3.00
CO-5					
CO-6					

Attainment (Internal 1 Examination) **3.00**

Attainment Level	
1	40%
2	60%
3	>60%

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

RI INDU INSTITUTE OF ENGINEERING AND TECHNOLOG'

Department of Computer Science And Engineering

Course Outcome Attainment (Internal Examination-2)

Name of the faculty Mr.s.Prudhviraj | Academic Year: 2020-21
Branch & Section: CSE-A | Examination: II
Course Name: Python Programming | Year: IV | Semester: I

S.No	HT No.	Q1a	Q1b	Q2a	Q2b	Q3a	Q3b	Q4a	Q4b	Obj4	A4
Max. Marks ==>		5		5		5		5		10	5
1	17X31A0501					3		5		5	5
2	17X31A0502			2		5				3	5
3	17X31A0503			3		1				4	5
4	17X31A0505					2				5	5
5	17X31A0506					4				4	5
6	17X31A0508					2				5	5
7	17X31A0509			3		4				5	5
8	17X31A0510			4		5				6	5
9	17X31A0511					2				5	5
10	17X31A0513	2		4						5	5
11	17X31A0515										5
12	17X31A0516			4		5				6	5
13	17X31A0518	3				3				4	5
14	17X31A0520			2		3				4	5
15	17X31A0521					3		5		5	5
16	17X31A0522			5		5				9	5
17	17X31A0523			5		2				4	5
18	17X31A0524			0		0				2	5
19	17X31A0525			1		3				4	5
20	17X31A0526			3		4				5	5
21	17X31A0528			3		4				5	5
22	17X31A0529	2								5	5
23	17X31A0531					2				5	5
24	17X31A0532	2				3				4	5
25	17X31A0533			5		3				5	5
26	17X31A0534					3				4	5
27	17X31A0538			3		5				6	5
28	17X31A0539					3				4	5
29	17X31A0542			3		3				6	5
30	17X31A0545			3		5				6	5
31	17X31A0546					4		5		6	5
32	17X31A0547			0		5				5	5
33	17X31A0548	0				2				5	5
34	17X31A0549					2				5	5
35	17X31A0550			3		3				4	5
36	17X31A0552			3		3				5	5
37	17X31A0553					4				5	5
38	17X31A0554										5
39	17X31A0555			4		5				6	5
40	17X31A0556					2		4		5	5
41	17X31A0557			3		5				5	5
42	17X31A0558										5
43	17X31A0559			2		3				3	5
44											

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M)
R.R Dist. Telangana -501 510

45											
46											
47											
48											
49											
50											
51											
52											
53											
54											
55											
56											
57											
58											
59											
60											
61											
62											
63											
64											
65											
66											
67											
68											
69											
70											
71											
72											
73											
Target set by the faculty / HoD		3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00	6.00	3.00
Number of students performed above the target		1	0	17	0	28	0	4	0	8	43
Number of students attempted		5	0	23	0	38	0	4	0	40	43
Percentage of students scored more than target		20%		74%		74%		100%		20%	100%

CO Mapping with Exam Questions:

CO - 1										
CO - 2										
CO - 3										
CO - 4	y		y						y	y
CO - 5					y				y	y
CO - 6							y		y	y

CO Attainment based on Exam Questions:

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

CO - 1										
CO - 2										
CO - 3										
CO - 4	20%		74%						20%	100%
CO - 5					74%				20%	100%
CO - 6							100%		20%	100%

CO	Subj	obj	Asgn	Overall	Level
CO-1					
CO-2					
CO-3					
CO-4	47%	20%	100%	56%	2.00
CO-5	74%	20%	100%	65%	3.00
CO-6	100%	20%	100%	73%	3.00

Attainment Level	
1	40%
2	60%
3	>60%

Attainment (Internal Examination- **2.67**

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M)
R.R Dist. Telangana -501 510

RI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science And Engineering

Course Outcome Attainment (University Examinations)

Name of the faculty : Mr.s.Prudhviraj
Academic Year: 2020-21
Branch & Section: CSE-A
Year / Semester: IV / I
Course Name: Python Programming

S.No	Roll Number	Marks Secured
1	17X31A0501	13
2	17X31A0502	26
3	17X31A0503	45
4	17X31A0505	56
5	17X31A0506	51
6	17X31A0508	26
7	17X31A0509	13
8	17X31A0510	28
9	17X31A0511	28
10	17X31A0513	13
11	17X31A0515	17
12	17X31A0516	51
13	17X31A0518	26
14	17X31A0520	54
15	17X31A0521	60
16	17X31A0522	51
17	17X31A0523	44
18	17X31A0524	34
19	17X31A0525	33
20	17X31A0526	55
21	17X31A0528	26
22	17X31A0529	37
23	17X31A0531	17
24	17X31A0532	38
25	17X31A0533	-1
26	17X31A0534	26
27	17X31A0538	28
28	17X31A0539	64
29	17X31A0542	27
30	17X31A0545	26
31	17X31A0546	48
32	17X31A0547	26
33	17X31A0548	37
34	17X31A0549	50
35	17X31A0550	26
Max Marks		75

S.No	Roll Number	Marks Secured
36	17X31A0552	12
37	17X31A0553	26
38	17X31A0554	58
39	17X31A0555	30
40	17X31A0556	42
41	17X31A0557	9
42	17X31A0558	27
43	17X31A0559	51
44		
45		
46		
47		
48		
49		
50		
51		
52		
53		
54		
55		
56		
57		
58		
59		
60		
61		
62		
63		
64		
65		
66		
67		
68		
69		
70		

Class Average mark	#DIV/0!
Number of students performed above the target	0
Number of successful students	43
Percentage of students scored more than target	0%
Attainment level	**1**

Attainment Level	% students
1	40%
2	60%
3	>60%

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M),
R.R Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science And Engineering

Course Outcome Attainment

Name of the faculty : Mr.s.Prudhviraj
Branch & Section: CSE-A
Course Name: Python Programming

Academic Year 2020-21
Examination: I Internal
Year: IV
Semester: I

Course Outcomes	1st Internal Exam	2nd Internal Exam	Internal Exam	University Exam	Attainment Level
CO1	3.00		3.00	1.00	2.40
CO2	3.00		3.00	1.00	2.40
CO3	3.00		3.00	1.00	2.40
CO4	3.00	2.00	2.50	1.00	2.05
CO5		3.00	3.00	1.00	2.40
CO6		3.00	3.00	1.00	2.40
Internal & University Attainment:			2.92	1.00	
Weightage			70%	30%	
O Attainment for the course (Internal, Universit			2.04	0.30	
CO Attainment for the course (Direct Method)			2.34		

Overall course attainment level **2.34**

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M),
R.R Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department of Computer Science And Engineering

Program Outcome Attainment (from Course)

Name of Faculty:	Mr.s.Prudhviraj	Academic Year:	2019-20
Branch & Section:	CSE-A	Year:	II
Course Name:	Python Programming	Semester:	I

CO-PO mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2		3						1			
CO2	1		3		2				1			
CO3		2	3									
CO4		2			3						1	
CO5			3		1							2
CO6		2	3									1
Course	**1.50**	**2.00**	**3.00**		**2.00**				**1.00**		**1.00**	**1.50**

CO	Course Outcome Attainment
CO1	2.40
CO2	2.40
CO3	2.40
CO4	2.05
CO5	2.40
CO6	2.40
Overall course attainment level	**2.34**

PO-ATTAINMENT

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO Attainment	**1.17**	**1.56**	**2.34**		**1.56**				**0.78**		**0.78**	**1.17**

CO contribution to PO - 33%, 67%, 100% (Level 1/2/3)

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
Dist. Telangana -501 510