Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

THEORY COURSE FILE CONTENTS

Check list Course Outcomes Attainment

S. No.	Contents	Available (Y/N/NA)	Date of Submission	Signature of HOD
1.	Authenticated Syllabus Copy	Y	25-11-21	
2.	Individual Time Table	Y		
3.	Students' Name List (Approved Copy)	Y		
4.	Course Plan, PO, PSO, COs, CO-PO Mapping, COA Plan, Session Plan and Periodic Monitoring	Y		
5.	Previous Year End Semester Question Papers	Y		
6.	Question Bank (All Units - Part A, Part B & C)	Y		
7.	Dissemination of Syllabus and Course Plan to Students	Y		
8.	Lecture Notes - Unit I, II & III	Y		
9.	**Sample Documents and Evaluation Sheet for Internal Assessment** – Tutorials / Assignments / Class Test / Open Book Test / Quiz / Project / Seminar / Role Play if any (Before Mid Term)	Y		
10.	**Mid Term Examination** A. Question Paper / Any Other Assessment Tools Used B. Sample Answer Scripts (Best, Average, Poor) if required C. Evaluation Sheet D. Slow Learners List and Remedial Measures			
11.	Lecture Notes – Unit IV & V			
12.	**Sample Documents and Evaluation Sheet for Internal Assessment** – Tutorials / Assignments / Class Test /Open Book Test / Quiz / Project / Seminar / Role Play if any (After Mid Term)			
13.	Course End Survey (Indirect Assessment) & Consolidation			
14.	**End Term Examination** A. Question Paper & Answer Key B. Sample Answer Scripts (Best, Average, Poor) if required C. Evaluation Sheet			

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

	D. Slow Learners List and Remedial Measures.			
15.	Content Beyond the Syllabus (Proof)			
16.	Innovative Teaching Tools Used for TLP			
17.	Details of Visiting Faculty Session / Industry Expert / Guest Lecture / Seminar / Field Visit / Webinars / Flipped Class Room / Blended Learning / Online Resources etc.			
18.	Consolidated Mark Statement			
19.	CO Attainment (Mid Term + Internal Assessment + End Term)			
20.	Gap Analysis & Remedial Measures			
21.	CO - PO Attainment			
22.	Class Record (Faculty Logbook)			

Signature of HOD/ Dean **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

Syllabus Copy

Course Code	**CSE11410**	L	T	P	C
Version 1.0		3	0	0	3
Pre-requisites/Exposure	**Browser compatibility knowledge /HTML**				
Co-requisites					

Course Objectives

1. To help the pupils to develop an understanding of Python programming
2. To enable students a precise understanding of list ,dictionary in Python.
3. To give the students a perspective of Python language
4. To enable students, to give an over view of File operation in Python.
5. To assess student's knowledge in Python library Import /Python IDE.

Course Content

Course Content

Unit-I ***12 Lecture Hours***

Introduction to Python: Introduction to Python, Python variables, expressions, statements, Variables, Keywords, Operators & operands, Expressions, Statements, Order of operations, String operations, Comments, Keyboard input, Example programs, Functions- Type conversion function, Math functions, Composition of functions, Defining own function, parameters, arguments, Importing functions, Example programs

Unit II: ***8 Lecture Hours***

Conditions & Iterations: Conditions- Modulus operator, Boolean expression, Logical operators, if, ifelse, if-elif-else, Nested conditions, Example programs,

Iteration- while, for, break, continue, Nested loop, Example programs

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

Unit III: ***13 Lecture Hours***

Recursion, Strings, List, Dictionaries, Tuples: Recursion- Python recursion, Examples of recursive functions, Recursion error, Advantages & disadvantages of recursion

Strings- Accessing values in string, Updating strings, Slicing strings, String methods – upper(), find(), lower(), capitalize(), count(), join(), len(), isalnum(), isalpha(), isdigit(), islower(), isnumeric(), isspace(), isupper() max(), min(), replace(), split(), Example programs

List- Introduction, Traversal, Operations, Slice, Methods, Delete element, Difference between lists and strings, Example program

Dictionaries- Introduction, Brief idea of dictionaries & lists

Tuples- Introduction, Brief idea of lists & tuples, Brief idea of dictionaries & tuples

Unit IV: ***10 Lecture Hours***

I/O & File: Data Streams, Creating Your Own Data Streams, Access Modes, Writing Data to a File, Reading Data from a File, Additional File Methods, Using Pipes as Data Streams

Classes & Objects: Creating class, Instance objects, Accessing attributes, Built in class attributes, destroying objects, Inheritance, Method overriding, Overloading methods, Overloading operators, Data hiding, Example program

Unit V: ***2 Lecture Hours***

Python Exceptions Exception handling: assert statement, Except clause - with no exceptions and multiple exceptions, Try - finally, raising exceptions, user-defined exceptions.

Text Books:

1. Introducing Python- Modern Computing in Simple Packages – Bill Lubanovic, O,,Reilly Publication
2. Beginning Python: From Novice to Professional, Magnus Lie Hetland, Apress
3. Programming In Python, Dr. Pooja Sharma, BPB

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Reference Books:

1. Beginning Programming with Python for Dummies Paperback – 2015 by John Paul Mueller
2. Python Programming - Using Problem Solving Approach, Reema Thareja, OXFORD UNIVERSITY PRESS

Faculty Individual Time Table

ADAMAS UNIVERSITY, KOLKATA

SCHOOL OF ENGINEERING TECHNOLOGY

DEPARTMENT OF CSE

Programme:BCA

Course Code & Course: CSE11410,PYTHON
Faculty Coordinator: SUBHASISH MOHAPATRA

Day & Time	10.30 - 11.20	11.20 - 12.10	12.10 - 01.00	01.00 - 01.50	01.50 - 02.40	02.40 - 03.30	03.30 - 04.20	04.20 - 05.10
Monday	-			LUNCH				
Tuesday	PYTHON		-					
Wednesday	-				-			
Thursday	-				-	PYTHON		-
Friday	-	-	-		PYTHON			

Signature of HOD **Signature of Class Coordinator**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Students Name List

Roll Number	Registration Number	Name of the Student
UG/02/BCA/2020/001	AU/2020/0004253	DEBOJYOTI SAHA
UG/02/BCA/2020/037	AU/2020/0005526	Oliva Dutta
UG/02/BCA/2020/005	AU/2020/0004453	SAYANTAN JANA
UG/02/BCA/2020/006	AU/2020/0004457	SANJUKTA JANA
UG/02/BCA/2020/007	AU/2020/0004458	AYAN RAHAMAN
UG/02/BCA/2020/008	AU/2020/0004461	SUSOVON NANDY
UG/02/BCA/2020/009	AU/2020/0004478	Hritankar Das
UG/02/BCA/2020/021	AU/2020/0004513	Arpan Mondal
UG/02/BCA/2020/024	AU/2020/0004517	Aparesh Muhuri
UG/02/BCA/2020/025	AU/2020/0004520	Kosturi Mondal
UG/02/BCA/2020/026	AU/2020/0004522	Aritra Das
UG/02/BCA/2020/028	AU/2020/0004526	Neelash Saha
UG/02/BCA/2020/029	AU/2020/0004533	Bittaswer Ghosh
UG/02/BCA/2020/031	AU/2020/0004543	Abhishek Tarafdar
UG/02/BCA/2020/032	AU/2020/0004547	Ayon Chakraborty
UG/02/BCA/2020/034	AU/2020/0004564	Asmat Sk
UG/02/BCA/2020/035	AU/2020/0004575	Nikhil Kumar Sah
UG/02/BCA/2020/036	AU/2020/0004582	Suprita Nandy
UG/02/BCA/2020/022	AU/2020/0004514	Parichoy nandi
UG/02/BCA/2020/018	AU/2020/0004507	Pritam Hore
UG/02/BCA/2020/019	AU/2020/0004509	Aratrika Bose
UG/02/BCA/2020/010	AU/2020/0004482	SWARNAMOY GHOSH
UG/02/BCA/2020/013	AU/2020/0004496	Suman Ghosh
UG/02/BCA/2020/033	AU/2020/0004552	JYOTISHKA DE
UG/02/BCA/2020/020	AU/2020/0004510	Tithi Paul
UG/02/BCA/2020/017	AU/2020/0004504	Dhrubajyoti Dey
UG/02/BCA/2020/011	AU/2020/0004483	ANWESHA PRAMANIK
UG/02/BCA/2020/002	AU/2020/0004290	AZMAT ALI
UG/02/BCA/2020/027	AU/2020/0004525	RISHI BARUA
UG/02/BCA/2020/012	AU/2020/0004492	Swapnil No Mitra
UG/02/BCA/2020/015	AU/2020/0004498	Anthony Prakash Rozario
UG/02/BCA/2020/016	AU/2020/0004501	MOUSUMI DUTTA
UG/02/BCA/2020/030	AU/2020/0004535	SUNEET CHOUDHARY
UG/02/BCA/2020/004	AU/2020/0004449	DEBDYUTI DAS

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

UG/02/BCA/2020/023	AU/2020/0004515	Aditya Jaman
UG/02/BCA/2020/003	AU/2020/0004448	SATYAJIT GHOSH
UG/02/BCABFSI/2020/001	AU/2020/0004505	Somnath Gayen
UG/02/BCABFSI/2020/002	AU/2020/0004598	BARUN RAJBHAR
UG/02/BCABFSI/2020/003	AU/2020/0004605	RAKIBUL ISLAM
UG/02/BCAGA/2020/006	AU/2020/0004497	Abhishek Mondal
UG/02/BCAGA/2020/005	AU/2020/0004568	SUBHAJIT SIRCAR
UG/02/BCAGA/2020/001	AU/2020/0004493	Susmit Shaw
UG/02/BCAGA/2020/003	AU/2020/0004524	Sourav Mondal
UG/02/BCAGA/2020/002	AU/2020/0004500	Arka Mitra
UG/02/BCAGA/2020/004	AU/2020/0004539	Ranita Bagchi

Signature of HOD/Dean **Signature of Class Coordinator**

Date: **Date:**

COURSE PLAN

Target	60% (marks)
Level-1	50% (population)
Level-2	60% (population)

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Level-3	70% (population)

1. **Method of Evaluation**

UG	**PG**
Internal Assessment (30%) (Quizzes/Tests, Assignments & Seminars etc.)	Internal Assessment (30%) (Quizzes/Tests, Assignments & Seminars etc.)
Mid Semester Examination (20%)	Mid Semester Examination (20%)
End Semester Examination (50%)	End Semester Examination (50%)

*Keep as per Program (UG/PG)

2. **Passing Criteria**

Scale	**PG**	**UG**
Out of 10 Point Scale	CGPA – "5.00" Min. Individual Course Grade – "C" Passing Minimum – 40	CGPA – "5.00" Min. Individual Course Grade – "C" Passing Minimum – 35

*Keep as per Program (UG/PG)

3. **Pedagogy**
   - **Direct Instruction**
   - Kinesthetic Learning
   - **Flipped Classroom**
   - Differentiated Instruction
   - Expeditionary Learning
   - Inquiry Based Learning
   - Game Based Learning
   - Personalized Learning
4. **Topics introduced for the first time in the program through this course**
   - (New Topics Related to this Course – Syllabus Revision if any/Content Beyond Syllabus)
5. **References:**

Text Books	Web Resources	Journals	Reference Books
2	2	NA	1

Signature of HOD/Dean **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra		**Course Code: CSE11410**
7. Course	**: CSE**	**L: 3**
8. Program	**: BCA**	**T: 0**
9. Target	**: 60%**	**P: 0**
		C: 3

GUIDELINES TO STUDY THE SUBJECT

Instructions to Students:

1. Go through the 'Syllabus' in the LMS in order to find out the Reading List.
2. Get your schedule and try to pace your studies as close to the timeline as possible.
3. Get your on-line lecture notes (Content, videos) at Lecture Notes section. These are our lecture notes. Make sure you use them during this course.
4. check your LMS regularly
5. go through study material
6. check mails and announcements on blackboard
7. keep updated with the posts, assignments and examinations which shall be conducted on the blackboard
8. Be regular, so that you do not suffer in any way
9. C**ell Phones and other Electronic Communication Devices:** Cell phones and other electronic communication devices (such as Blackberries/Laptops) are not permitted in classes during Tests or the Mid/Final Examination. Such devices MUST be turned off in the class room.
10. **E-Mail and online learning tool:** Each student in the class should have an e-mail id and a pass word to access the LMS system regularly. Regularly, important information – Date of conducting class tests, guest lectures, via online learning tool. The best way to arrange meetings with us or ask specific questions is by email and prior appointment. All the assignments preferably should be uploaded on online learning tool. Various research papers/reference material will be mailed/uploaded on online learning platform time to time.
11. **Attendance:** Students are required to have minimum attendance of 75% in each subject. Students with less than said percentage shall NOT be allowed to appear in the end semester examination.

This much should be enough to get you organized and on your way to having a great semester! If you need us for anything, send your feedback through e-mail XXX@adamasuniversity.ac.in Please use an appropriate subject line to indicate your message details.

There will no doubt be many more activities in the coming weeks. So, to keep up to date with all the latest developments, please keep visiting this website regularly.

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

RELATED OUTCOMES

1. **The expected outcomes of the Program are:**

PO1	Engineering Knowledge- Acquire Knowledge of web design foundations, in the design and modelling of web application-based system.
PO2	Problem analysis- Avail appropriately web design notations and apply web design in Real time ProJect in order to design, plan, and implement software systems.
PO3	Design/development of solutions- Design solutions for complex engineering problems and design system components or processes that meet the specified needs
PO4	Conduct investigation of complex problem- Own Skills of observations and drawing logical inferences from the scientific experiments and develop application programs to meet the desired results
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- meet the desired results including attainable constraints such as social, economic, environmental, functional, and technological.
PO7	Environment and sustainability- Appraise regarding the social and environmental issues to fulfil the local and global needs and give relevant solutions for them.
PO8	Ethics-: Understand and adopt emerging technologies, research, strategies for lifelong learning at national and international level
PO9	Individual or Team work- Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
PO10	Communication- Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work
PO11	Project management and finance- Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO12	Life long learning- Understand and adopt emerging technologies, research, strategies for lifelong learning at national and international level.

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

2. **The expected outcomes of the Specific Program are: (up to 3)**

PSO1	To engage in professional development and to pursue post graduate education in the fields of Information Technology and Computer Applications.
PSO2	To provide the students about computing principles and business practices in software solutions, outsourcing services, public and private sectors.
PSO3	Analyze and synthesis computing systems through quantitative and qualitative techniques.

3. **The expected outcomes of the Course are: (minimum 4 and maximum 6)**

CO1	**Classify** the fundamental Python syntax and semantics and show the use of Python control flow statements.
CO2	**Demonstrate** the methods to create and manipulate Python programs by utilizing the data structures like lists, dictionaries, tuples, sets and strings.
CO3	**Develop** proficiency in the handling of functions.
CO4	**Identify** the Object-Oriented Programming concepts such as encapsulation, inheritance and polymorphism as used in Python.
CO5	**Find** the commonly used operations to handle run time error or Exception
etc.	

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

4. **Co-Relationship Matrix**
Indicate the relationships by 1- Slight (Low) 2- Moderate (Medium) 3-Substantial (High)

Program Outcomes / Course Outcomes	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	1					2	2	3	3	3	1		
CO2	3	3	1					2	2	3	3	3	1		
CO3	3	3						2	2	3	3	3			
CO4			1					2	2	3			1		
CO5	3	3									3	3			
etc.															
Average	**3**	**3**	**1**					**2**	**2**	**3**	**3**	**3**	**1**		

5. **Course Outcomes Assessment Plan (COA):**

Course Outcomes	Internal Assessment* (30 Marks)		Mid Term Exam (20 Marks)	End Term Exam (50 Marks)	Total (100 Marks)
	Before Mid Term	After Mid Term			
CO1	5	NA	7	8	20
CO2	5	NA	7	8	20
CO3	3	3	6	8	20
CO4	NA	7	NA	13	20
CO5	NA	7	NA	13	20

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

etc.					
Total	13	17	20	50	100

* Internal Assessment – Tools Used: Tutorial, Assignment, Seminar, Class Test etc.

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

OVERVIEW OF COURSE PLAN OF COURSE COVERAGE

Course Activities:

S. No.	Description	Planned			Actual			Remarks
		From	To	No. of Session	From	TO	No. of Session	
1.	Introduction To Python,LOOP,Array	11.09.2021	25.10.2021	10	11.09.2021	25.10.2021	10	
2.	*Tuple,String,List,Dictionary*	02.11.2021	28.11.2021	8	2.11.2021	28.11.2021	8	4 classes extra due to less students
3.	*File and matrix concept*	29.11.2021	4.12.2020	9	29.11.2021	4.12.2021	9	3 days extra due to Exam
4.	*Matrix operation, numpy*	4.12.2021	25.12.2021	8	4.12.2021	25.12.2021	8	4 days extra due to problem solvimg
5.	*Python and OOP concept*	25.12.2021	5.01.2022	10	25.12.2021	5.01.2022	10	

Total No. of Instructional periods available for the course: _____ Sessions

Signature of HOD/Dean **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

SESSION PLAN

Session Plan				Actual Delivery			
Lect.	Date	Topics to be Covered	CO Mapped	Lect.	Date	Topics Covered	CO Achieved
1	11-09-21	Introduction to Python, Python variables, expressions, statements	CO1	1	11-09-21	Introduction to Python, Python variables, expressions, statements	CO1
2	13-09-21	Variables, Keywords, Operators & operands	CO1	2	13-09-21	Variables, Keywords, Operators & operands	CO1
3	16-09-21	Expressions, Statements, Order of operations, String	CO1	3	16-09-21	Expressions, Statements, Order of operations, String	CO1
4	5-10-21	operations, Comments, Keyboard input, Example	CO1	4	5-10-21	operations, Comments, Keyboard input, Example	CO1
5	7-10-21	Math functions,	CO1	5	7-10-21	Math functions,	CO1
6	11-10-21	Composition of functions	CO1	6	11-10-21	Composition of functions	CO1
7	14-10-21	Defining own function,	CO1	7	14-10-21	Defining own function,	CO1
8	15-10-21	parameters, arguments, Importing functions, Example programs	CO1	8	15-10-21	parameters, arguments, Importing functions, Example programs	CO1
9	16-10-21	programs, Functions- Type conversion function,	CO1	9	16-10-21	programs, Functions- Type conversion function,	CO1

UNIT-I

Remarks: **Signature of Faculty**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Date:

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

SESSION PLAN
UNIT-II

Session Plan				Actual Delivery			
Lect.	Date	Topics to be Covered	CO Mapped	Lect.	Date	Topics Covered	CO Achieved
1	17-10-21	Composite function with example	CO2	1	17-10-21	Composite function with example	CO2
2	18-10-21	**Conditions & Iterations: Conditions-** Modulus operator,	CO2	2	18-10-21	**Conditions & Iterations: Conditions-** Modulus operator,	CO2
3	21-10-21	Boolean expression, Logical operators	CO2	3	21-10-21	Boolean expression, Logical operators	CO2
4	22-10-21	Document Structure Tags, Formatting Tags, Text Level Formatting,	CO2	4	22-10-21	Document Structure Tags, Formatting Tags, Text Level Formatting,	CO2
5	23-10-21	If-Elif,while logic with example	CO2	5	23-10-21	If-Elif,while logic with example	CO2
6	29-10-21	Array concept with example	CO2	6	29-10-21	Array concept with example	CO2
7	2-11-21	while, for, example	CO2	7	2-11-21	while, for, example	CO2
8	4-11-21	break, continue, with example	CO2	8	4-11-21	break, continue, with example	CO2
9							

Remarks:

Signature of Faculty

Date:

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

SESSION PLAN
UNIT-III

Session Plan				Actual Delivery			
Lect.	Date	Topics to be Covered	CO Mapped	Lect.	Date	Topics Covered	CO Achieved
1	6-11-21	**Recursion, Strings, List**	CO3	1	6-11-21	**Recursion, Strings, List**	CO3
2	7-11-21	Python recursion, Examples of recursive functi	CO4	2	7-11-21	Python recursion, Examples of recursive functi	CO4
3	10-11-21	Recursion error, Advantages & disadvantages of recursion	CO3	3	10-11-21	Recursion error, Advantages & disadvantages of recursion	CO3
4	11-11-21	**Strings-** Accessing values in string, Updating strings, Slicing strings	CO4	4	11-11-21	**Strings-** Accessing values in string, Updating strings, Slicing strings	CO4
5	14-11-21	String methods – upper(), find(), lower(), capitalize(), .	CO3	5	14-11-21	String methods – upper(), find(), lower(), capitalize(), .	CO3
6	18-11-21	count(), join(), len(), isalnum(), isalpha(),	CO4	6	18-11-21	count(), join(), len(), isalnum(), isalpha(),	CO4
7	22-11-21	isdigit(), islower(), isnumeric(),	CO3	7	22-11-21	isdigit(), islower(), isnumeric(),	CO3
8	24-11-21	isspace(), isupper() max(), min(),	CO4	8	24-11-21	isspace(), isupper() max(), min(),	CO4
9	25-11-21	Doubt Clearing	CO3	8	25-11-21	Doubt Clearing	CO3

Remarks: **Signature of Faculty**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Date:

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

SESSION PLAN
UNIT-IV

Session Plan				Actual Delivery			
Lect.	Date	Topics to be Covered	CO Mapped	Lect.	Date	Topics Covered	CO Achieved
1	2-06-21	Javascript Assignment operator	CO4	1	2-06-21	Javascript Assignment operator	CO4
2	4-06-21	Javascript security	CO4	2	4-06-21	Javascript security	CO4
3	7-06-21	Javasript event	CO4	3	7-06-21	Javasript event	CO4
4	12-06-21	Different event mouse click event	CO4	4	12-06-21	Different event mouse click event	CO4
5	15-06-21	If-else ,for loop in javascript	CO4	5	15-06-21	If-else ,for loop in javascript	CO4
6	17-06-21	Class assignment	CO4	6	17-06-21	Class assignment	CO4
7	18-06-21	Javascript array	CO4	7	18-06-21	Javascript array	CO4
8	21-06-21	Boolean operator	CO4	8	21-06-21	Boolean operator	CO4
9	22-06-21	String concept in Javascript	CO4	8	22-06-21	String concept in Javascript	CO4

Remarks: **Signature of Faculty**

Date:

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

SESSION PLAN
UNIT-V

Session Plan				Actual Delivery			
Lect.	Date	Topics to be Covered	CO Mapped	Lect.	Date	Topics Covered	CO Achieved
1	2-06-21	Javascript Assignment operator	CO4	1	2-06-21	Javascript Assignment operator	
2	4-06-21	Javascript security	CO4	2	4-06-21	Javascript security	
3	7-06-21	Javasript event	CO4	3	7-06-21	Javasript event	
4	12-06-21	Different event mouse click event	CO4	4	12-06-21	Different event mouse click event	
5	15-06-21	If-else ,for loop in javascript	CO4	5	15-06-21	If-else ,for loop in javascript	
6	17-06-21	Class assignment	CO4	6	17-06-21	Class assignment	
7	18-06-21	Javascript array	CO4	7	18-06-21	Javascript array	
8	21-06-21	Boolean operator	CO4	8	21-06-21	Boolean operator	
9	16-7-21						

Remarks: **Signature of Faculty**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Date:

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

PERIODIC MONITORING

Actual date of completion and remarks, if any

Components		From	To	From	To
Duration (Mention from and to Dates)		11.09.2021	04.10.2021	07.11.2021	26.12.2021
Percentage of Syllabus covered		50%		50%	
Lectures	1	27	28	51	
	1	27	28	51	
Tutorials					
Test/Quizzes/ Mid Semester/ End Semester	1	1(MID)	1	1(END)	
	1	1	1	1	
	CO1 & CO2	CO1, CO2 & CO3	CO4 & CO5	CO1, CO2, CO3, CO4 & CO5	
	CO1 & CO2	CO1, CO2 & CO3	CO4 & CO5	CO1, CO2, CO3, CO4 & CO5	
Assignments	1	1	1	1	
	1	1	1	1	
	CO1	CO2 & CO3	CO4	CO5	
	CO1	CO2 & CO3	CO4	CO5	
Signature of Faculty					
Head of the Department					
OBE Coordinator					

Signature of HOD/ Dean

Date

Signature of Faculty

Date

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

PERIODIC MONITORING

Attainment of the Course (Learning) Outcomes:

Components	Attainment level	Action Plan	Remarks
Assignment	CO1:	12-09-21	Assignment Questions on CO2and CO3
	CO2:		
	CO3:		
	CO4:		
	CO5:		
Quiz/Test etc.	CO1:		
	CO2:	5-11-21	Practice Question on CO3, CO4 and CO5
	CO3:		
	CO4:		
	CO5:		
Mid Semester	CO1:	17-12-21	Mid Semester Question given to understand CO1,CO2 and CO3
	CO2:		
	CO3:		
	CO4:		
	CO5:		
End Semester	CO1:		
	CO2:		
	CO3:		
	CO4:		
	CO5:		
Any Other	CO1:		
	CO2:		
	CO3:		
	CO4:		
	CO5:		

Signature of HOD/ Dean

Date

Signature of Faculty

Date

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

Lecture Notes

UNIT-1

The split() method returns a list where the text between the specified separator becomes the list items.

Example

The split() method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

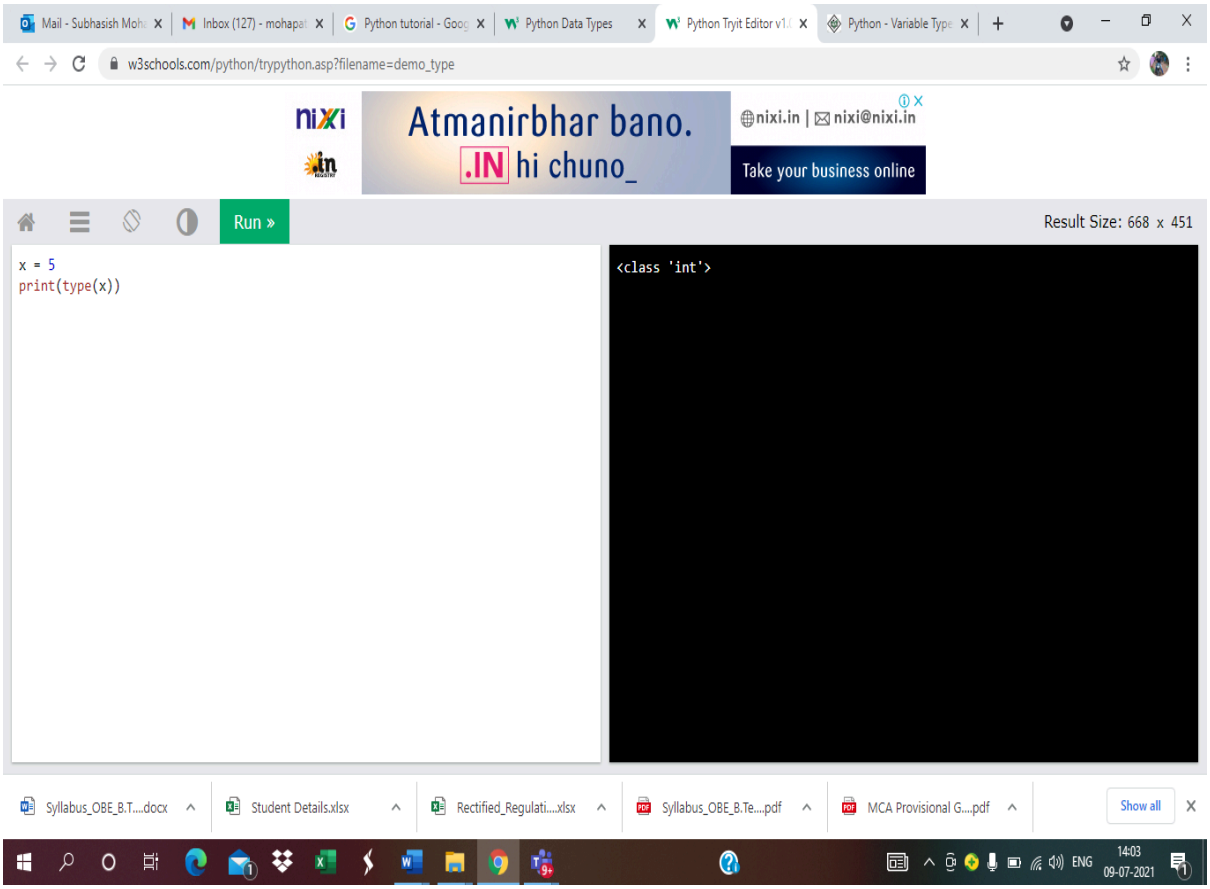
Example

```
if 5 > 2:
  print("Five is greater than two!")
```

.

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3



Python Comments

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

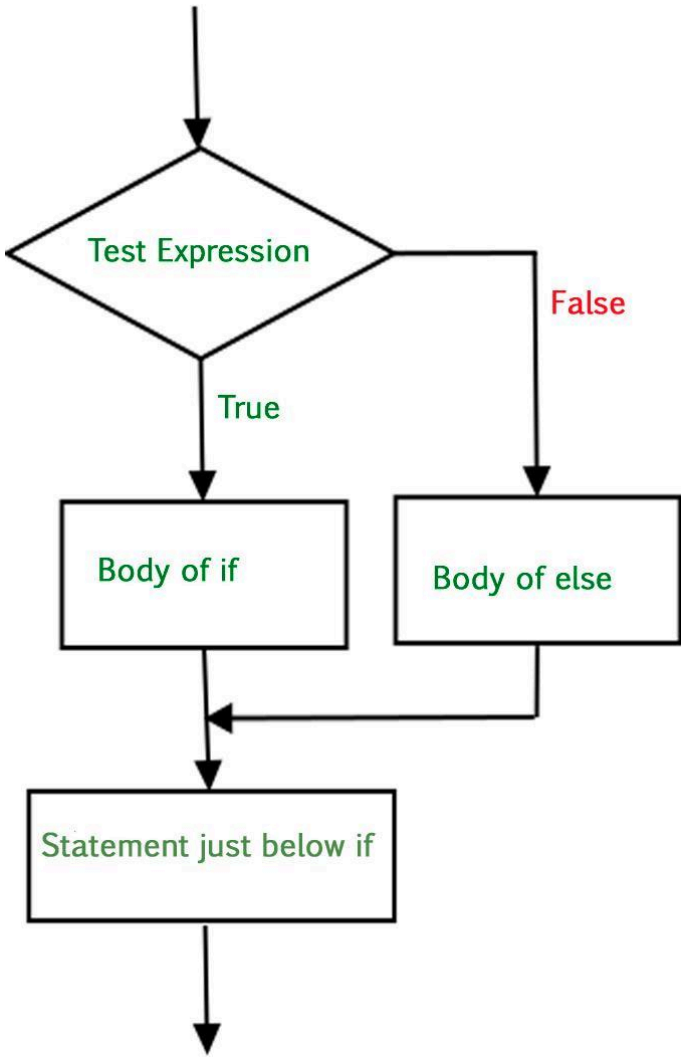
Creating a Comment

Comments starts with a #, and Python will ignore them:

Example

```
#This is a comment
print("Hello, World!")
```

UNIT-2



UNIT-3

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

FUNCTION IN PYTHON

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

Creating a Function

In Python a function is defined using the def keyword:

Example

```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
  print("Hello from a function")

my_function()
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

MODULOUS OPERATOR

When we see a '%' the first thing that comes to our mind is the "Percentage-sign", but when we think of it from the perspective of computer language, this sign has, in fact, another name and meaning. In computing, the **modulo operation**(%) finds the **remainder** or **signed remainder** after the division of one number by another (called the modulus of the operation).

Given two positive numbers, a and n, a modulo n (a % n, abbreviated as a **mod** n) is the remainder of the **Euclidean division** of a by n, where a is the dividend and n is the divisor.

Basically, Python modulo operation is used to get the remainder of a division. The modulo operator(**%**) is considered an arithmetic operation, along with **+**, **–**, **/**, *, **, **//**. In most languages, both operands of this modulo operator have to be an integer. But Python Modulo is versatile in this case. The operands can be either **integer** or **float**.

Syntax:

```
a % b
```

Here, a is divided by b, and the remainder of that division is returned.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

Code:

- Python3

```
# inputs
a = 13
b = 5

# Stores the remainder obtained
# when dividing a by b, in c
c = a % b
print(a, "mod", b, "=",
      c, sep = " ")

# inputs
d = 15.0
e = 7.0

# Stores the remainder obtained
# when dividing d by e, in f
f = d % e
print(d, "mod", e, "=",
      f, sep = " ")
```

Output:

```
13 mod 5 = 3
```

```
15.0 mod 7.0 = 1.0
```

This was a simple example showing the use of the syntax, and a basic operation performed by the modulo operator. Suppose, we want to calculate the remainder of every number from 1 to n when divided by a fixed number k.

- Python3

```
# function is defined for finding out

# the remainder of every number from 1 to n

def findRemainder(n, k):

  for i in range(1, n + 1):

    # rem will store the remainder

    # when i is divided by k.

    rem = i % k

    print(i, "mod", k, "=",

          rem, sep = " ")

# Driver code

if __name__ == "__main__":

  # inputs

  n = 5

  k = 3
```

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

```
# function calling

findRemainder(n, k)
```

Output:

```
1 mod 3 = 1

2 mod 3 = 2

3 mod 3 = 0

4 mod 3 = 1

5 mod 3 = 2
```

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

UNIT-IV

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

iF-Else logic

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

EX

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

O/P

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

Else

The else keyword catches anything which isn't caught by the preceding conditions.

EX

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b"
```

O/P

a is greater than b

EX

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

O/P

Both conditions are True

Recursion

Python also accepts function recursion, which means a defined function can call itself.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, tri_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing

EX

```
def tri_recursion(k):

  if(k > 0):

    result = k + tri_recursion(k - 1)

    print(result)

  else:

    result = 0

  return result



print("\n\nRecursion Example Results")

tri_recursion(6)
```

O/P

```
Recursion Example Results
1
3
6
```

10
15
21

Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the `print()` function:

EX

print("Hello")

print('Hello')

Hello

a = "Hello"

print(a)

Hello

EX a = "Hello, World!"

print(a[1])

e

EX

txt = "The best things in life are free!"

print("expensive" not in txt)

O/P

True

EX

```
txt = "The best things in life are free!"

if "expensive" not in txt:

  print("No, 'expensive' is NOT present.")
```

O/P

No, 'expensive' is NOT present.

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

EX

```
b = "Hello, World!"
print(b[2:5])
```

O/P

Llo

EX

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

O/P
Hello

EX

Slice To the End

By leaving out the *end* index, the range will go to the end:

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

O/P
llo, World!
EX

Negative Indexing

Use negative indexes to start the slice from the end of the string:

Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

O/P

Orl

EX

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

Example

The `upper()` method returns the string in upper case:

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

HELLO, WORLD!

EX

The split() method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

COCATE METHOD

```
a = "Hello"

b = "World"

c = a + " " + b

print(c)
```

Hello World

EX

Use the format() method to insert numbers into strings:

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

My name is John, and I am 36

EX

Escape Characters

EX

```
txt = 'It\'s alright.'
print(txt)
```

O/P

```
It's alright.
```

EX

```
#A backslash followed by three integers will result in a octal value:
txt = "\110\145\154\154\157"
print(txt)
```

O/P

```
Hello
#A backslash followed by an 'x' and a hex number represents a hex value:
txt = "\x48\x65\x6c\x6c\x6f"
print(txt)
```

O/P

```
HELLO
```

Definition and Usage

The `map()` function executes a specified function for each item in an iterable. The item is sent to the function as a parameter.

Syntax

map(*function*, *iterables*)

Parameter	Description
function	Required. The function to execute for each item
iterable	Required. A sequence, collection or an iterator object. You can send as many iteral make sure the function has one parameter for each iterable.

EX

```
def myfunc(a):
  return len(a)

x = map(myfunc, ('apple', 'banana', 'cherry'))

print(x)

#convert the map into a list, for readability:
print(list(x))
```

```
<map object at 0x056D44F0>
[5, 6, 6]
```

Python Lambda: Exercise-6 with Solution

Write a Python program to square and cube every number in a given list of integers using Lambda.

Sample Solution:

Python Code :

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print("Original list of integers:")

print(nums)

print("\nSquare every number of the said list:")

square_nums = list(map(lambda x: x ** 2, nums))

print(square_nums)

print("\nCube every number of the said list:")

cube_nums = list(map(lambda x: x ** 3, nums))

print(cube_nums)
```

Copy

Sample Output:

```
Original list of integers:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Square every number of the said list:
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Cube every number of the said list:
```

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra		**Course Code: CSE11410**
7. Course	**: CSE**	**L: 3**
8. Program	**: BCA**	**T: 0**
9. Target	**: 60%**	**P: 0**
		C: 3

[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

Python Interview Questions for Freshers

1. What is Python?

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management which help in modeling real-world problems and building applications to solve these problems.

2. What are the benefits of using Python?

- Python is a general-purpose programming language that has a simple, easy-to-learn syntax that emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, is completely open-source, and supports third-party packages encouraging modularity and code reuse.
- Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

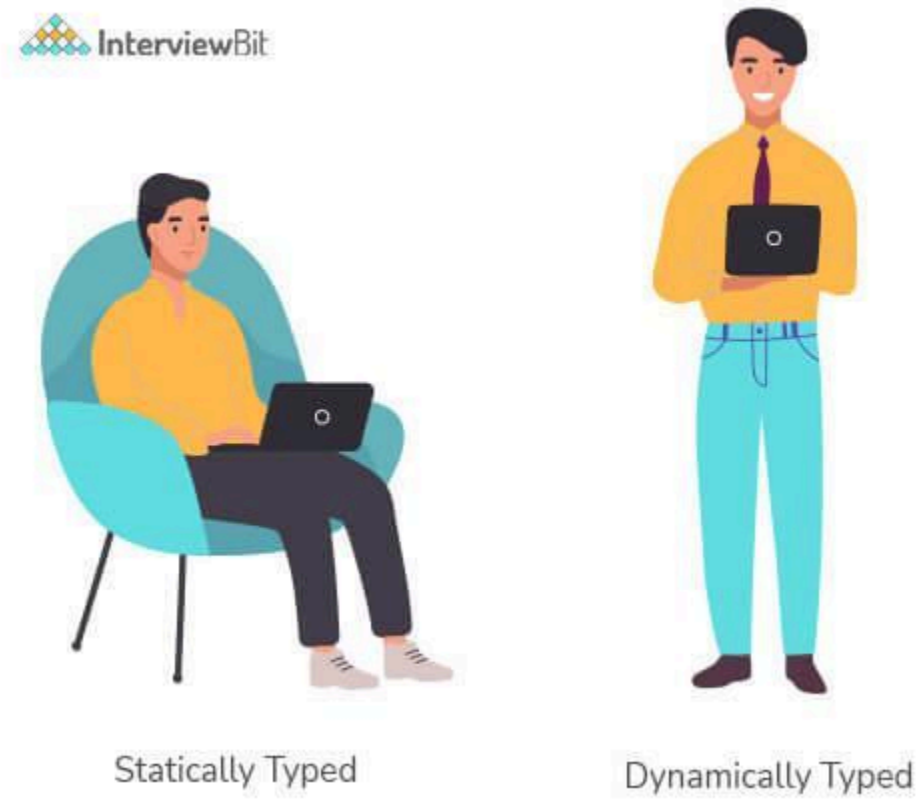
3. What is a dynamically typed language?

Before we understand a dynamically typed language, we should learn about what typing is. **Typing** refers to type-checking in programming languages. In a **strongly-typed** language, such as Python, **"1" + 2** will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a **weakly-typed** language, such as Javascript, will simply output **"12"** as result.

Type-checking can be done at two stages -

- **Static** - Data Types are checked before execution.
- **Dynamic** - Data Types are checked during execution.

Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed Language.



You can download a PDF version of Python Interview Questions.

4. What is an Interpreted language?

An Interpreted language executes its statements line by line. Languages such as Python, Javascript, R, PHP, and Ruby are prime examples of Interpreted languages. Programs written in an interpreted language runs directly from the source code, with no intermediary compilation step.

5. What is PEP 8 and why is it important?

PEP stands for **Python Enhancement Proposal**. A PEP is an official design document providing information to the Python community, or describing a new feature for Python or its

processes. **PEP 8** is especially important since it documents the style guidelines for Python Code. Apparently contributing to the Python open-source community requires you to follow these style guidelines sincerely and strictly.

6. What is Scope in Python?

Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

- A **local scope** refers to the local objects available in the current function.
- A **global scope** refers to the objects available throughout the code execution since their inception.
- A **module-level scope** refers to the global objects of the current module accessible in the program.
- An **outermost scope** refers to all the built-in names callable in the program. The objects in this scope are searched last to find the name referenced.

Note: Local scope objects can be synced with global scope objects using keywords such as **global**.

7. What are lists and tuples? What is the key difference between the two?

Lists and **Tuples** are both **sequence data types** that can store a collection of objects in Python. The objects stored in both sequences can have **different data types**. Lists are represented with **square brackets** `['sara', 6, 0.19]`, while tuples are represented with **parantheses** `('ansh', 5, 0.97)`.

But what is the real difference between the two? The key difference between the two is that while **lists are mutable**, **tuples** on the other hand are **immutable** objects. This means that lists can be modified, appended or sliced on the go but tuples remain constant and cannot be modified in any manner. You can run the following example on Python IDLE to confirm the difference:

```
my_tuple = ('sara', 6, 5, 0.97)
my_list = ['sara', 6, 5, 0.97]
print(my_tuple[0])    # output => 'sara'
print(my_list[0])    # output => 'sara'
my_tuple[0] = 'ansh'    # modifying tuple => throws an error
my_list[0] = 'ansh'    # modifying list => list modified
print(my_tuple[0])    # output => 'sara'
print(my_list[0])    # output => 'ansh'
```

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra		**Course Code: CSE11410**
7. Course	**: CSE**	**L: 3**
8. Program	**: BCA**	**T: 0**
9. Target	**: 60%**	**P: 0**
		C: 3

8. What are the common built-in data types in Python?

There are several built-in data types in Python. Although, Python doesn't require data types to be defined explicitly during variable declarations type errors are likely to occur if the knowledge of data types and their compatibility with each other are neglected. Python provides `type()` and `isinstance()` functions to check the type of these variables. These data types can be grouped into the following categories-

- **None Type:**
  None keyword represents the null values in Python. Boolean equality operation can be performed using these NoneType objects.

Class Name	Description
NoneType	Represents the **NULL** values in Python.

- **Numeric Types:**
  There are three distinct numeric types - **integers, floating-point numbers**, and **complex numbers**. Additionally, **booleans** are a sub-type of integers.

Class Name	Description
int	Stores integer literals including hex, octal and binary numbers as integers
float	Stores literals containing decimal values and/or exponent signs as floating-point numbers
complex	Stores complex numbers in the form (A + Bj) and has attributes: real and imag
bool	Stores boolean value (True or False).

Note: *The standard library also includes* ***fractions*** *to store rational numbers and* ***decimal*** *to store floating-point numbers with user-defined precision.*

- **Sequence Types:**
  According to Python Docs, there are three basic Sequence Types - **lists, tuples,** and **range** objects. Sequence types have the in and not in operators defined for their traversing their elements. These operators share the same priority as the comparison operations.

Class Name	Description
list	Mutable sequence used to store collection of items.

Class Name	Description
tuple	Immutable sequence used to store collection of items.
range	Represents an immutable sequence of numbers generated during execution.
str	Immutable sequence of Unicode code points to store textual data.

Note: The standard library also includes additional types for processing:
1. **Binary data** such as `bytearray` `bytes` `memoryview` , and
2. **Text strings** such as `str`.

- **Mapping Types:**

A mapping object can map hashable values to random objects in Python. Mappings objects are mutable and there is currently only one standard mapping type, the ***dictionary***.

Class Name	Description
dict	Stores comma-separated list of **key: value** pairs

- **Set Types:**
  Currently, Python has two built-in set types - **set** and **frozenset**. **set** type is mutable and supports methods like add() and remove(). **frozenset** type is immutable and can't be modified after creation.

Class Name	Description
set	Mutable unordered collection of distinct hashable objects.
frozenset	Immutable collection of distinct hashable objects.

Note: *`set` is mutable and thus cannot be used as key for a dictionary. On the other hand, `frozenset` is immutable and thus, hashable, and can be used as a dictionary key or as an element of another set.*

- **Modules:**
  Module is an additional built-in type supported by the Python Interpreter. It supports one special operation, i.e., **attribute access**: mymod.myobj, where mymod is a module and **myobj** references a name defined in m's symbol table. The module's symbol table resides in a very special attribute of the module **__dict__**, but direct assignment to this module is neither possible nor recommended.

- **Callable Types:**
  Callable types are the types to which function call can be applied. They can be **user-defined functions, instance methods, generator functions**, and some other **built-in functions, methods** and **classes**.
  Refer to the documentation at docs.python.org for a detailed view of the **callable types**.

9. What is pass in Python?

The `pass` keyword represents a null operation in Python. It is generally used for the purpose of filling up empty blocks of code which may execute during runtime but has yet to be written. Without the **pass** statement in the following code, we may run into some errors during code execution.

```
def myEmptyFunc():
  # do nothing
  pass
myEmptyFunc()    # nothing happens
## Without the pass keyword
# File "<stdin>", line 3
# IndentationError: expected an indented block
```

10. What are modules and packages in Python?

Python packages and Python modules are two mechanisms that allow for **modular programming** in Python. Modularizing has several advantages -

- **Simplicity**: Working on a single module helps you focus on a relatively small portion of the problem at hand. This makes development easier and less error-prone.
- **Maintainability**: Modules are designed to enforce logical boundaries between different problem domains. If they are written in a manner that reduces interdependency, it is less likely that modifications in a module might impact other parts of the program.
- **Reusability**: Functions defined in a module can be easily reused by other parts of the application.
- **Scoping**: Modules typically define a separate namespace, which helps avoid confusion between identifiers from other parts of the program.

Modules, in general, are simply Python files with a .py extension and can have a set of functions, classes, or variables defined and implemented. They can be imported and initialized once using the `import` statement. If partial functionality is needed, import the requisite classes or functions using `from foo import bar`.

Packages allow for hierarchial structuring of the module namespace using **dot notation**. As, **modules** help avoid clashes between global variable names, in a similar

manner, **packages** help avoid clashes between module names.
Creating a package is easy since it makes use of the system's inherent file structure. So just stuff the modules into a folder and there you have it, the folder name as the package name. Importing a module or its contents from this package requires the package name as prefix to the module name joined by a dot.

Note: *You can technically import the package as well, but alas, it doesn't import the modules within the package to the local namespace, thus, it is practically useless.*

11. What are global, protected and private attributes in Python?
    - **Global** variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.
    - **Protected** attributes are attributes defined with an underscore prefixed to their identifier eg. _sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.
    - **Private** attributes are attributes with double underscore prefixed to their identifier eg. __ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.
12. What is self in Python?

Self is a keyword in Python used to define an instance of an object of a class. In Python, it is explicitly used as the first parameter, unlike in Java where it is optional. It helps in distinguishing between the methods and attributes of a class from its local variables.

13. What is __init__?

`__init__` is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a **__init__** method associated with them. It helps in distinguishing methods and attributes of a class from local variables.

```
# class definition
class Student:
  def __init__(self, fname, lname, age, section):
      self.firstname = fname
      self.lastname = lname
      self.age = age
      self.section = section
# creating a new object
stu1 = Student("Sara", "Ansh", 22, "A2")
```

6. Name of the Faculty: Subhasish Mohapatra		**Course Code: CSE11410**
7. Course	**: CSE**	**L: 3**
8. Program	**: BCA**	**T: 0**
9. Target	**: 60%**	**P: 0**
		C: 3

14. What is break, continue and pass in Python?

Break	The break statement terminates the loop immediately and the control flows to the statement after the body of the loop.
Continue	The continue statement terminates the current iteration of the statement, skips the rest of the code in the current iteration and the control flows to the next iteration of the loop.
Pass	As explained above, the pass keyword in Python is generally used to fill up empty blocks and is similar to an empty statement represented by a semi-colon in languages such as Java, C++, Javascript, etc.

```
pat = [1, 3, 2, 1, 2, 3, 1, 0, 1, 3]
for p in pat:
  pass
  if (p == 0):
    current = p
    break
  elif (p % 2 == 0):
    continue
  print(p)    # output => 1 3 1 3 1
print(current)    # output => 0
```

15. What are unit tests in Python?

- Unit test is a unit testing framework of Python.
- Unit testing means testing different components of software separately. Can you think about why unit testing is important? Imagine a scenario, you are building software that uses three components namely A, B, and C. Now, suppose your software breaks at a point time. How will you find which component was responsible for breaking the software? Maybe it was component A that failed, which in turn failed component B, and this actually failed the software. There can be many such combinations.
- This is why it is necessary to test each and every component properly so that we know which component might be highly responsible for the failure of the software.

16. What is docstring in Python?

- Documentation string or docstring is a multiline string used to document a specific code segment.
- The docstring should describe what the function or method does.

17. What is slicing in Python?

- As the name suggests, ‘slicing’ is taking parts of.
- Syntax for slicing is **[start : stop : step]**
- **start** is the starting index from where to slice a list or tuple
- **stop** is the ending index or where to sop.
- **step** is the number of steps to jump.

- Default value for **start** is 0, **stop** is number of items, **step** is 1.
- Slicing can be done on **strings, arrays, lists**, and **tuples**.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(numbers[1 : : 2])  #output : [2, 4, 6, 8, 10]
```

18. Explain how can you make a Python Script executable on Unix?

- Script file must begin with **#!/usr/bin/env python**

19. What is the difference between Python Arrays and lists?

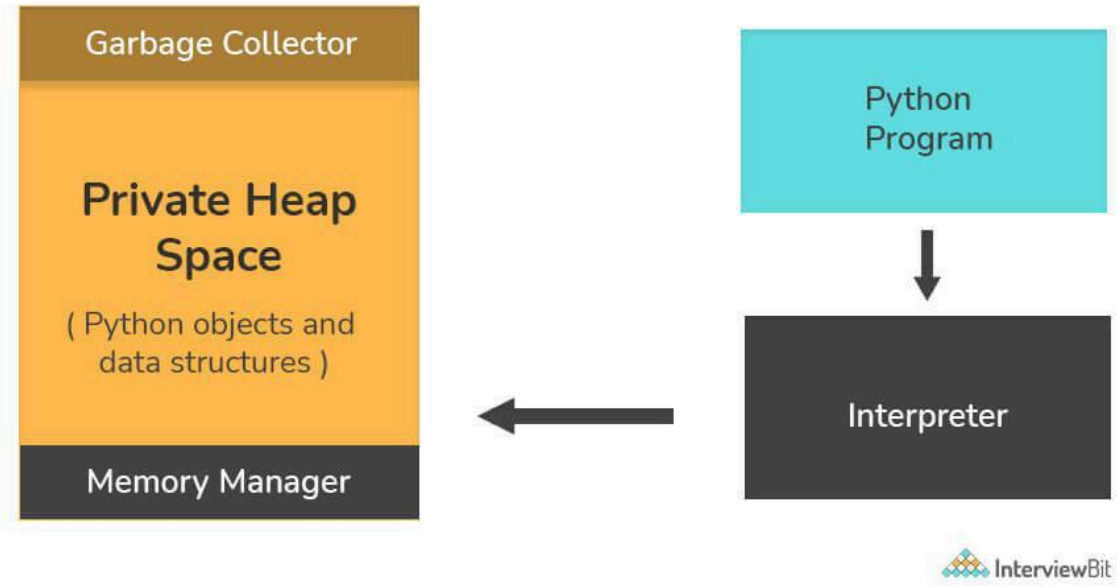
- Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays and consumes far less memory than lists.
- Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

```
import array
a = array.array('i', [1, 2, 3])
for i in a:
   print(i, end=' ')    #OUTPUT: 1 2 3
a = array.array('i', [1, 2, 'string'])    #OUTPUT: TypeError: an integer is required (got type str)
a = [1, 2, 'string']
for i in a:
  print(i, end=' ')    #OUTPUT: 1 2 string
```

Python Interview Questions for Experienced

20. How is memory managed in Python?

- Memory management in Python is handled by the **Python Memory Manager**. The memory allocated by the manager is in form of a **private heap space** dedicated to Python. All Python objects are stored in this heap and being private, it is inaccessible to the programmer. Though, python does provide some core API functions to work upon the private heap space.
- Additionally, Python has an in-built garbage collection to recycle the unused memory for the private heap space.

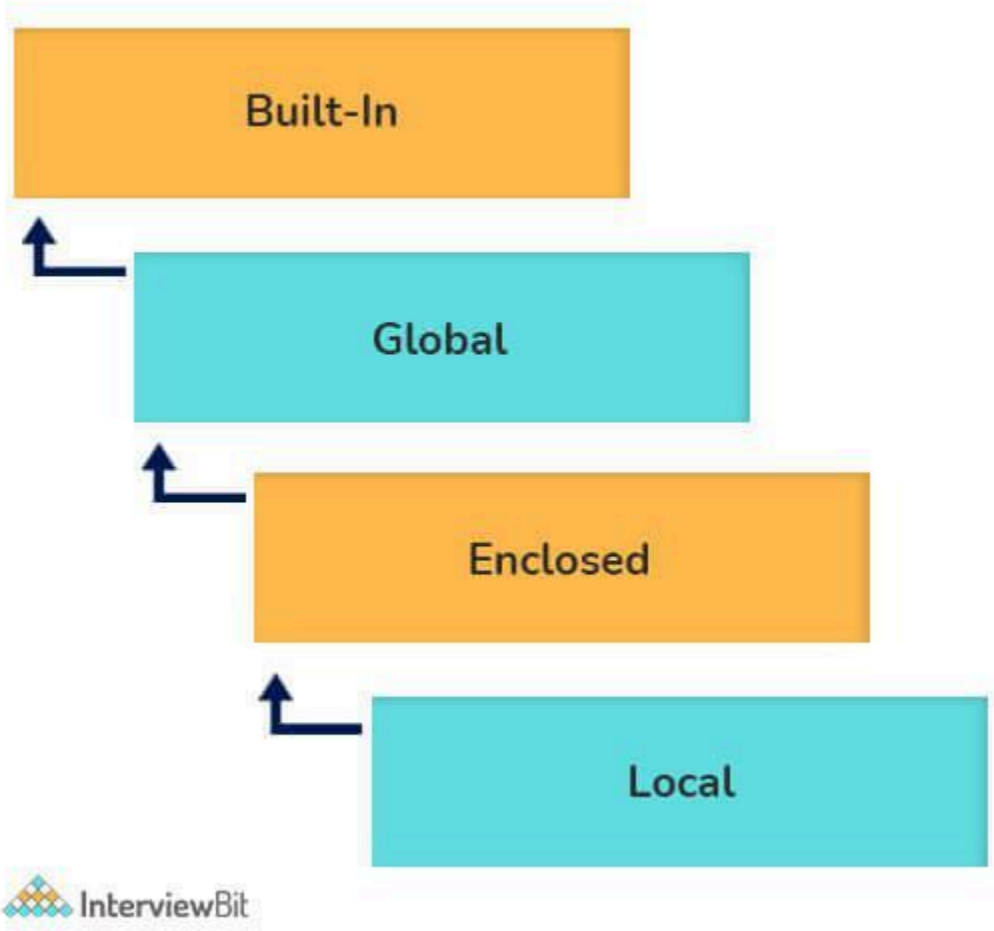


21. What are Python namespaces? Why are they used?

A namespace in Python ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to a corresponding 'object as value'. This allows for multiple namespaces to use the same name and map it to a separate object. A few examples of namespaces are as follows:

- **Local Namespace** includes local names inside a function. the namespace is temporarily created for a function call and gets cleared when the function returns.
- **Global Namespace** includes names from various imported packages/ modules that are being used in the current project. This namespace is created when the package is imported in the script and lasts until the execution of the script.
- **Built-in Namespace** includes built-in functions of core Python and built-in names for various types of exceptions.

The **lifecycle of a namespace** depends upon the scope of objects they are mapped to. If the scope of an object ends, the lifecycle of that namespace comes to an end. Hence, it isn't possible to access inner namespace objects from an outer namespace.



22. What is Scope Resolution in Python?

Sometimes objects within the same scope have the same name but function differently. In such cases, scope resolution comes into play in Python automatically. A few examples of such behavior are:

- Python modules namely 'math' and 'cmath' have a lot of functions that are common to both of them - log10(), acos(), exp() etc. To resolve this ambiguity, it is necessary to prefix them with their respective module, like math.exp() and cmath.exp().
- Consider the code below, an object temp has been initialized to 10 globally and then to 20 on function call. However, the function call didn't change the value of the temp globally. Here, we can observe that Python draws a clear line between global and local variables, treating their namespaces as separate identities.

```
temp = 10   # global-scope variable
def func():
```

```
    temp = 20   # local-scope variable
    print(temp)
print(temp)   # output => 10
func()    # output => 20
print(temp)   # output => 10
```

This behavior can be overridden using the `global` keyword inside the function, as shown in the following example:

```
temp = 10   # global-scope variable
def func():
    global temp
    temp = 20   # local-scope variable
    print(temp)
print(temp)   # output => 10
func()    # output => 20
print(temp)   # output => 20
```

23. What are decorators in Python?

Decorators in Python are essentially functions that add functionality to an existing function in Python without changing the structure of the function itself. They are represented the `@decorator_name` in Python and are called in a bottom-up fashion. For example:

```
# decorator function to convert to lowercase
def lowercase_decorator(function):
  def wrapper():
      func = function()
      string_lowercase = func.lower()
      return string_lowercase
  return wrapper
# decorator function to split words
def splitter_decorator(function):
  def wrapper():
      func = function()
      string_split = func.split()
      return string_split
  return wrapper
@splitter_decorator # this is executed next
@lowercase_decorator # this is executed first
def hello():
  return 'Hello World'
```

```
hello()   # output => [ 'hello' , 'world' ]
```

The beauty of the decorators lies in the fact that besides adding functionality to the output of the method, they can even **accept arguments** for functions and can further modify those arguments before passing it to the function itself. The **inner nested function**, i.e. 'wrapper' function, plays a significant role here. It is implemented to enforce **encapsulation** and thus, keep itself hidden from the global scope.

```
# decorator function to capitalize names
def names_decorator(function):
  def wrapper(arg1, arg2):
      arg1 = arg1.capitalize()
      arg2 = arg2.capitalize()
      string_hello = function(arg1, arg2)
      return string_hello
  return wrapper
@names_decorator
def say_hello(name1, name2):
  return 'Hello ' + name1 + '! Hello ' + name2 + '!'
say_hello('sara', 'ansh')   # output => 'Hello Sara! Hello Ansh!'
```

24. What are Dict and List comprehensions?

Python comprehensions, like decorators, are **syntactic sugar** constructs that help **build altered** and **filtered lists**, dictionaries, or sets from a given list, dictionary, or set. Using comprehensions saves a lot of time and code that might be considerably more verbose (containing more lines of code). Let's check out some examples, where comprehensions can be truly beneficial:

- **Performing mathematical operations on the entire list**

```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list]    # list comprehension
# output => [4 , 9 , 25 , 49 , 121]
squared_dict = {x:x**2 for x in my_list}    # dict comprehension
# output => {11: 121, 2: 4 , 3: 9 , 5: 25 , 7: 49}
```

- **Performing conditional filtering operations on the entire list**

```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list if x%2 != 0]    # list comprehension
# output => [9 , 25 , 49 , 121]
squared_dict = {x:x**2 for x in my_list if x%2 != 0}    # dict comprehension
# output => {11: 121, 3: 9 , 5: 25 , 7: 49}
```

- **Combining multiple lists into one**

Comprehensions allow for multiple iterators and hence, can be used to combine multiple lists into one.

```
a = [1, 2, 3]
b = [7, 8, 9]
[(x + y) for (x,y) in zip(a,b)]  # parallel iterators
# output => [8, 10, 12]
[(x,y) for x in a for y in b]    # nested iterators
# output => [(1, 7), (1, 8), (1, 9), (2, 7), (2, 8), (2, 9), (3, 7), (3, 8), (3, 9)]
```

- **Flattening a multi-dimensional list**

A similar approach of nested iterators (as above) can be applied to flatten a multi-dimensional list or work upon its inner elements.

```
my_list = [[10,20,30],[40,50,60],[70,80,90]]
flattened = [x for temp in my_list for x in temp]
# output => [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Note: *List comprehensions have the same effect as the map method in other languages. They follow the mathematical set builder notation rather than map and filter functions in Python.*

25. What is lambda in Python? Why is it used?

Lambda is an anonymous function in Python, that can accept any number of arguments, but can only have a single expression. It is generally used in situations requiring an anonymous function for a short time period. Lambda functions can be used in either of the two ways:

- Assigning lambda functions to a variable:

```
mul = lambda a, b : a * b
print(mul(2, 5))    # output => 10
```

- Wrapping lambda functions inside another function:

```
def myWrapper(n):
 return lambda a : a * n
mulFive = myWrapper(5)
print(mulFive(2))    # output => 10
```

26. How do you copy an object in Python?

In Python, the assignment statement (= operator) does not copy objects. Instead, it creates a binding between the existing object and the target variable name. To create copies of an object in Python, we need to use the **copy** module. Moreover, there are two ways of creating copies for the given object using the **copy** module -

Shallow Copy is a bit-wise copy of an object. The copied object created has an exact copy of the values in the original object. If either of the values is a reference to other objects, just the reference addresses for the same are copied.

Deep Copy copies all values recursively from source to target object, i.e. it even duplicates the objects referenced by the source object.

```
from copy import copy, deepcopy
list_1 = [1, 2, [3, 5], 4]
## shallow copy
list_2 = copy(list_1)
list_2[3] = 7
list_2[2].append(6)
list_2    # output => [1, 2, [3, 5, 6], 7]
list_1    # output => [1, 2, [3, 5, 6], 4]
## deep copy
list_3 = deepcopy(list_1)
list_3[3] = 8
list_3[2].append(7)
list_3    # output => [1, 2, [3, 5, 6, 7], 8]
list_1    # output => [1, 2, [3, 5, 6], 4]
```

27. What is the difference between xrange and range in Python?

xrange() and **range()** are quite similar in terms of functionality. They both generate a sequence of integers, with the only difference that `range()` returns a **Python list**, whereas, `xrange()` returns an **xrange object**.

So how does that make a difference? It sure does, because unlike range(), xrange() doesn't generate a static list, it creates the value on the go. This technique is commonly used with an object-type **generator** and has been termed as "**yielding**".

Yielding is crucial in applications where memory is a constraint. Creating a static list as in range() can lead to a `Memory Error` in such conditions, while, xrange() can handle it optimally by using just enough memory for the generator (significantly less in comparison).

```
for i in xrange(10):    # numbers from o to 9
  print i       # output => 0 1 2 3 4 5 6 7 8 9
for i in xrange(1,10):    # numbers from 1 to 9
  print i       # output => 1 2 3 4 5 6 7 8 9
for i in xrange(1, 10, 2):    # skip by two for next
  print i       # output => 1 3 5 7 9
```

Note: **xrange** has been **deprecated** as of **Python 3.x**. Now `range` does exactly the same as what `xrange` used to do in **Python 2.x**, since it was way better to use xrange() than the original range() function in Python 2.x.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

28. What is pickling and unpickling?

Python library offers a feature - **serialization** out of the box. Serializing an object refers to transforming it into a format that can be stored, so as to be able to deserialize it, later on, to obtain the original object. Here, the **pickle** module comes into play.

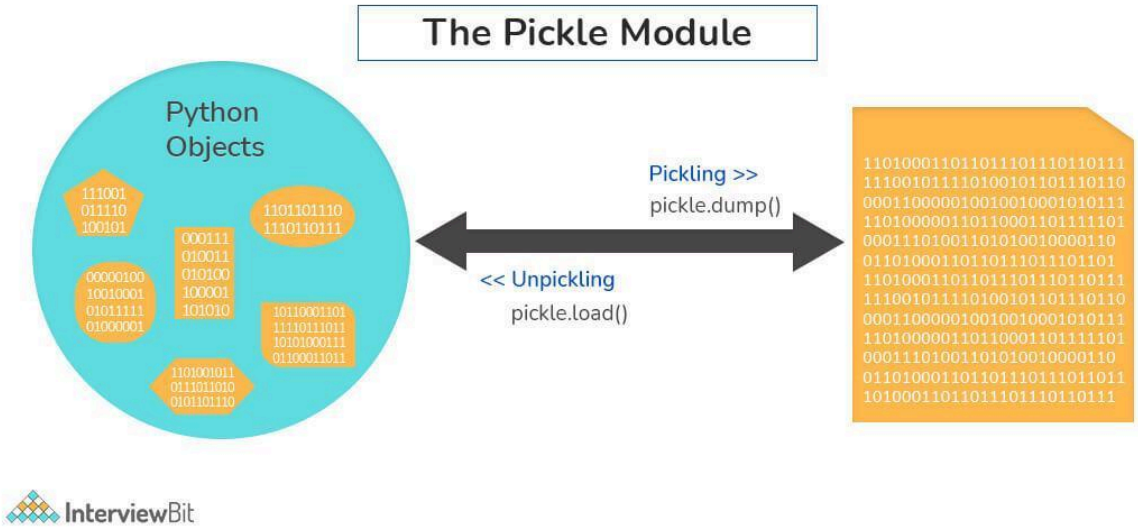
Pickling:

- Pickling is the name of the serialization process in Python. Any object in Python can be serialized into a byte stream and dumped as a file in the memory. The process of pickling is compact but pickle objects can be compressed further. Moreover, pickle keeps track of the objects it has serialized and the serialization is portable across versions.
- The function used for the above process is pickle.dump().

Unpickling:

- Unpickling is the complete inverse of pickling. It deserializes the byte stream to recreate the objects stored in the file and loads the object to memory.
- The function used for the above process is pickle.load().

Note: *Python has another, more primitive, serialization module called* ***marshall****, which exists primarily to* ***support .pyc files*** *in Python and* ***differs significantly from the pickle****.*



29. What are generators in Python?

Generators are functions that return an iterable collection of items, one at a time, in a set manner. Generators, in general, are used to create iterators with a different approach. They employ the use of `yield` keyword rather than `return` to return a **generator** object.
Let's try and build a generator for fibonacci numbers -

```
## generate fibonacci numbers upto n
def fib(n):
  p, q = 0, 1
  while(p < n):
     yield p
     p, q = q, p + q
x = fib(10)    # create generator object

## iterating using __next__(), for Python2, use next()
x.__next__()    # output => 0
x.__next__()    # output => 1
x.__next__()    # output => 1
x.__next__()    # output => 2
x.__next__()    # output => 3
x.__next__()    # output => 5
x.__next__()    # output => 8
x.__next__()    # error

## iterating using loop
for i in fib(10):
  print(i)    # output => 0 1 1 2 3 5 8
```

30. What is PYTHONPATH in Python?

PYTHONPATH is an environment variable which you can set to add additional directories where Python will look for modules and packages. This is especially useful in maintaining Python libraries that you do not wish to install in the global default location.

31. What is the use of help() and dir() functions?

help() function in Python is used to display the documentation of modules, classes, functions, keywords, etc. If no parameter is passed to the `help()` function, then an interactive **help utility** is launched on the console.
dir() function tries to return a valid list of attributes and methods of the object it is called upon. It behaves differently with different objects, as it aims to produce the most relevant data, rather than the complete information.

- For Modules/Library objects, it returns a list of all attributes, contained in that module.
- For Class Objects, it returns a list of all valid attributes and base attributes.
- With no arguments passed, it returns a list of attributes in the current scope.

32. What is the difference between .py and .pyc files?

- .py files contain the source code of a program. Whereas, .pyc file contains the bytecode of your program. We get bytecode after compilation of .py file (source code). .pyc files are not created for all the files that you run. It is only created for the files that you import.
- Before executing a python program python interpreter checks for the compiled files. If the file is present, the virtual machine executes it. If not found, it checks for .py file. If found, compiles it to .pyc file and then python virtual machine executes it.
- Having .pyc file saves you the compilation time.

33. How Python is interpreted?

- Python as a language is not interpreted or compiled. Interpreted or compiled is the property of the implementation. Python is a bytecode(set of interpreter readable instructions) interpreted generally.
- Source code is a file with .py extension.
- Python compiles the source code to a set of instructions for a virtual machine. The Python interpreter is an implementation of that virtual machine. This intermediate format is called "bytecode".
- .py source code is first compiled to give .pyc which is bytecode. This bytecode can be then interpreted by the official CPython or JIT(Just in Time compiler) compiled by PyPy.

34. How are arguments passed by value or by reference in python?

- **Pass by value**: Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.
- **Pass by reference**: Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.

In Python, arguments are passed by reference, i.e., reference to the actual object is passed.

```
def appendNumber(arr):
  arr.append(4)
arr = [1, 2, 3]
print(arr)  #Output: => [1, 2, 3]
appendNumber(arr)
print(arr)  #Output: => [1, 2, 3, 4]
```

35. What are iterators in Python?

- An iterator is an object.
- It remembers its state i.e., where it is during iteration (see code below to see how)
- __iter__() method initializes an iterator.
- It has a __next__() method which returns the next item in iteration and points to the next element. Upon reaching the end of iterable object __next__() must return StopIteration exception.

- It is also self-iterable.
- Iterators are objects with which we can iterate over iterable objects like lists, strings, etc.

```
class ArrayList:
  def __init__(self, number_list):
    self.numbers = number_list
  def __iter__(self):
    self.pos = 0
    return self
  def __next__(self):
    if(self.pos < len(self.numbers)):
      self.pos += 1
      return self.numbers[self.pos - 1]
    else:
      raise StopIteration
array_obj = ArrayList([1, 2, 3])
it = iter(array_obj)
print(next(it)) #output: 2
print(next(it)) #output: 3
print(next(it))
#Throws Exception
#Traceback (most recent call last):
#...
#StopIteration
```

36. Explain how to delete a file in Python?

Use command **os.remove(file_name)**

```
import os
os.remove("ChangedFile.csv")
print("File Removed!")
```

37. Explain split() and join() functions in Python?

- You can use **split()** function to split a string based on a delimiter to a list of strings.
- You can use **join()** function to join a list of strings based on a delimiter to give a single string.

```
string = "This is a string."
string_list = string.split(' ') #delimiter is ‘space’ character or ‘ ‘
print(string_list) #output: ['This', 'is', 'a', 'string.']
print(' '.join(string_list)) #output: This is a string.
```

38. What does *args and **kwargs mean?

***args**

- *args is a special syntax used in the function definition to pass variable-length arguments.
- "*" means variable length and "args" is the name used by convention. You can use any other.

```
def multiply(a, b, *argv):
  mul = a * b
  for num in argv:
    mul *= num
  return mul
print(multiply(1, 2, 3, 4, 5)) #output: 120
```

****kwargs**

- **kwargs is a special syntax used in the function definition to pass variable-length keyworded arguments.
- Here, also, "kwargs" is used just by convention. You can use any other name.
- Keyworded argument means a variable that has a name when passed to a function.
- It is actually a dictionary of the variable names and its value.

```
def tellArguments(**kwargs):
  for key, value in kwargs.items():
    print(key + ": " + value)
tellArguments(arg1 = "argument 1", arg2 = "argument 2", arg3 = "argument 3")
#output:
# arg1: argument 1
# arg2: argument 2
# arg3: argument 3
```

39. What are negative indexes and why are they used?

- Negative indexes are the indexes from the end of the list or tuple or string.
- **Arr[-1]** means the last element of array **Arr[]**

```
arr = [1, 2, 3, 4, 5, 6]
#get the last element
print(arr[-1]) #output 6
#get the second last element
print(arr[-2]) #output 5
```

Python OOPS Interview Questions

40. How do you create a class in Python?

To create a class in python, we use the keyword "class" as shown in the example below:

```
class InterviewbitEmployee:
  def __init__(self, emp_name):
    self.emp_name = emp_name
```

To instantiate or create an object from the class created above, we do the following:

```
emp_1=InterviewbitEmployee("Mr. Employee")
```

To access the name attribute, we just call the attribute using the dot operator as shown below:

```
print(emp_1.name)
# Prints Mr. Employee
```

To create methods inside the class, we include the methods under the scope of the class as shown below:

```
class InterviewbitEmployee:
  def __init__(self, emp_name):
      self.emp_name = emp_name

  def introduce(self):
      print("Hello I am " + self.emp_name)
```

The self parameter in the init and introduce functions represent the reference to the current class instance which is used for accessing attributes and methods of that class. The self parameter has to be the first parameter of any method defined inside the class. The method of the class InterviewbitEmployee can be accessed as shown below:

```
emp_1.introduce()
```

The overall program would look like this:

```
class InterviewbitEmployee:
  def __init__(self, emp_name):
      self.emp_name = emp_name

  def introduce(self):
      print("Hello I am " + self.emp_name)

# create an object of InterviewbitEmployee class
emp_1 = InterviewbitEmployee("Mr Employee")
print(emp_1.emp_name)    #print employee name
emp_1.introduce()        #introduce the employee
```

41. How does inheritance work in python? Explain it with an example.

Inheritance gives the power to a class to access all attributes and methods of another class. It aids in code reusability and helps the developer to maintain applications without redundant code. The class inheriting from another class is a child class or also called a derived class. The class from which a child class derives the members are called parent class or superclass.

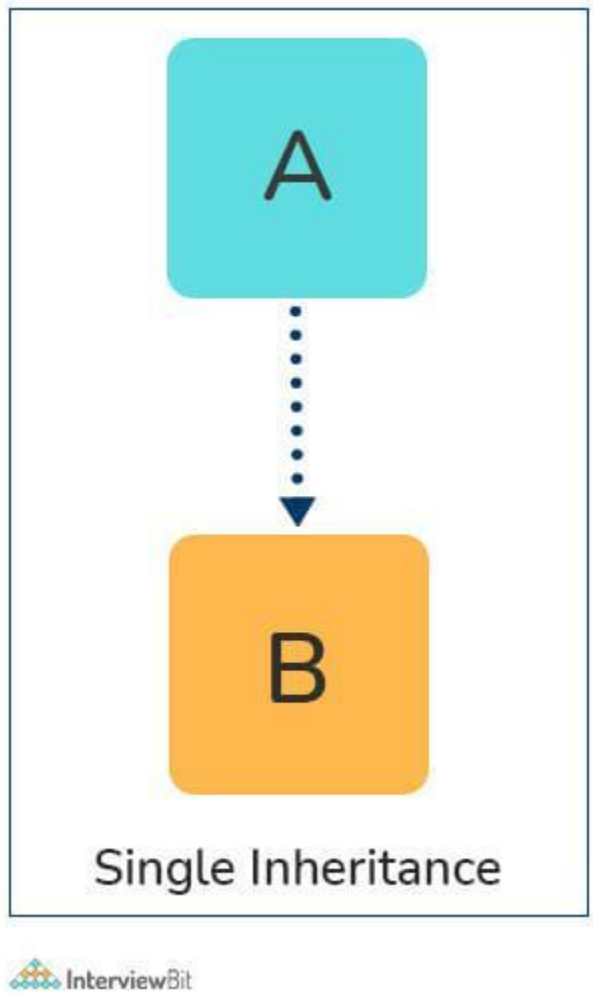
Python supports different kinds of inheritance, they are:

- **Single Inheritance**: Child class derives members of one parent class.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

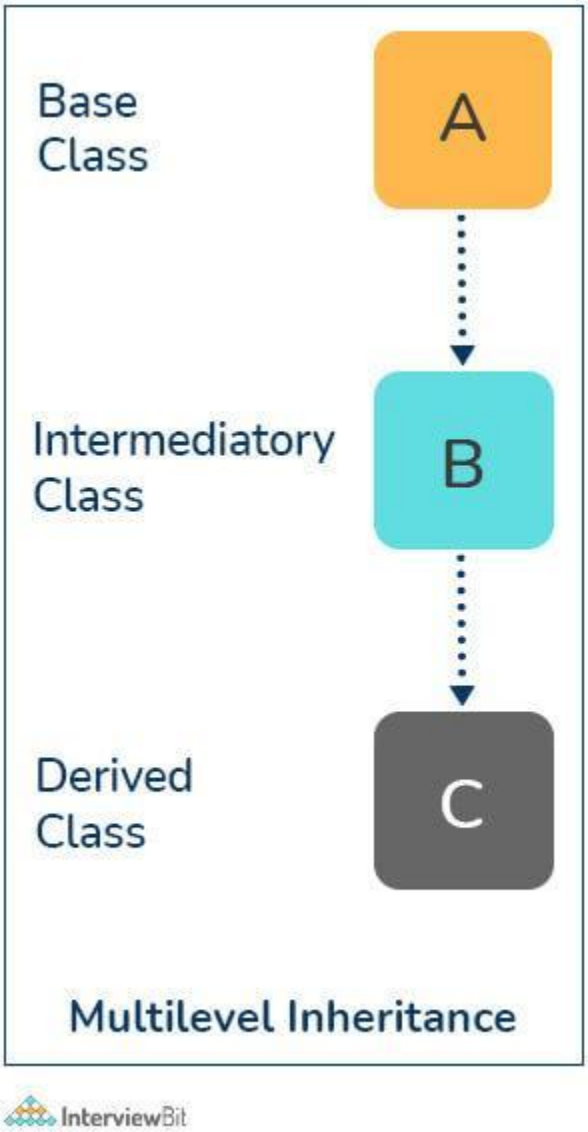
Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3




```
# Parent class
class ParentClass:
    def par_func(self):
        print("I am parent class function")

# Child class
class ChildClass(ParentClass):
    def child_func(self):
        print("I am child class function")

# Driver code
obj1 = ChildClass()
```

```
obj1.par_func()
obj1.child_func()
```

- **Multi-level Inheritance:** The members of the parent class, A, are inherited by child class which is then inherited by another child class, B. The features of the base class and the derived class are further inherited into the new derived class, C. Here, A is the grandfather class of class C.



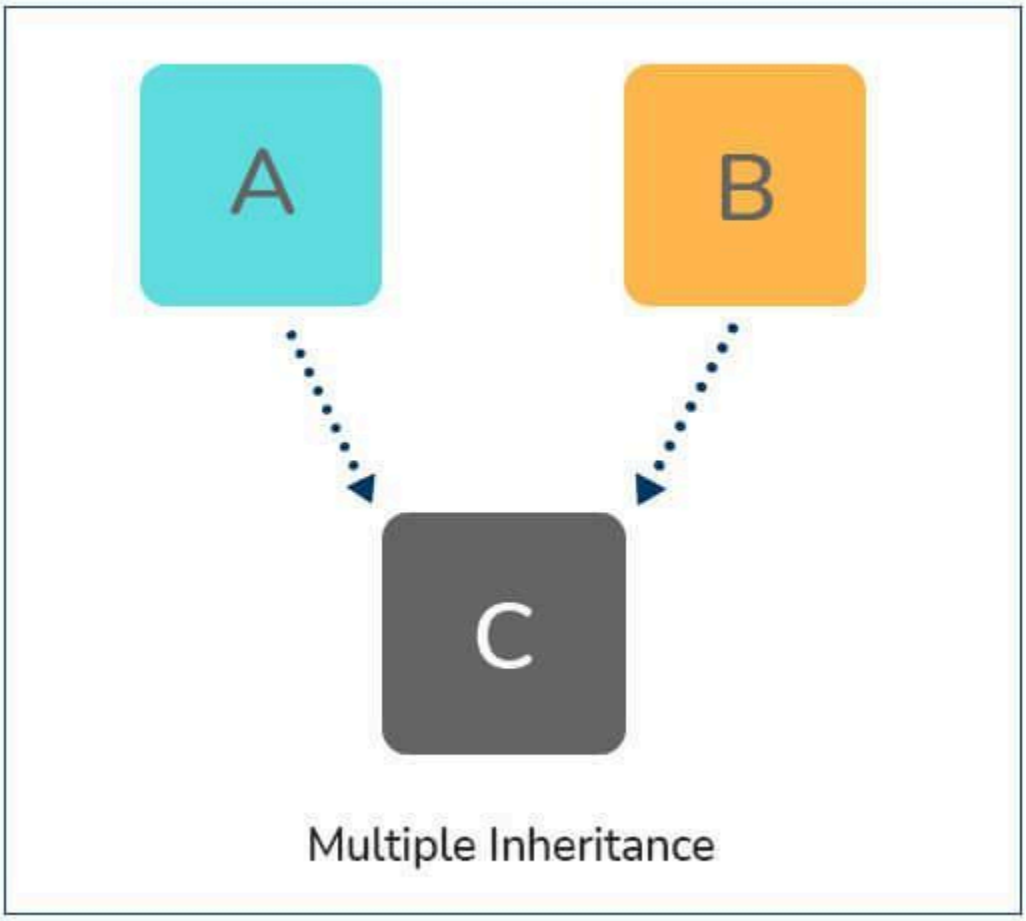

```
# Parent class
class A:
```

```
  def __init__(self, a_name):
      self.a_name = a_name

# Intermediate class
class B(A):
  def __init__(self, b_name, a_name):
      self.b_name = b_name
      # invoke constructor of class A
      A.__init__(self, a_name)

# Child class
class C(B):
  def __init__(self,c_name, b_name, a_name):
      self.c_name = c_name
      # invoke constructor of class B
      B.__init__(self, b_name, a_name)

  def display_names(self):
      print("A name : ", self.a_name)
      print("B name : ", self.b_name)
      print("C name : ", self.c_name)

#  Driver code
obj1 = C('child', 'intermediate', 'parent')
print(obj1.a_name)
obj1.display_names()
```

- **Multiple Inheritance:** This is achieved when one child class derives members from more than one parent class. All features of parent classes are inherited in the child class.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3

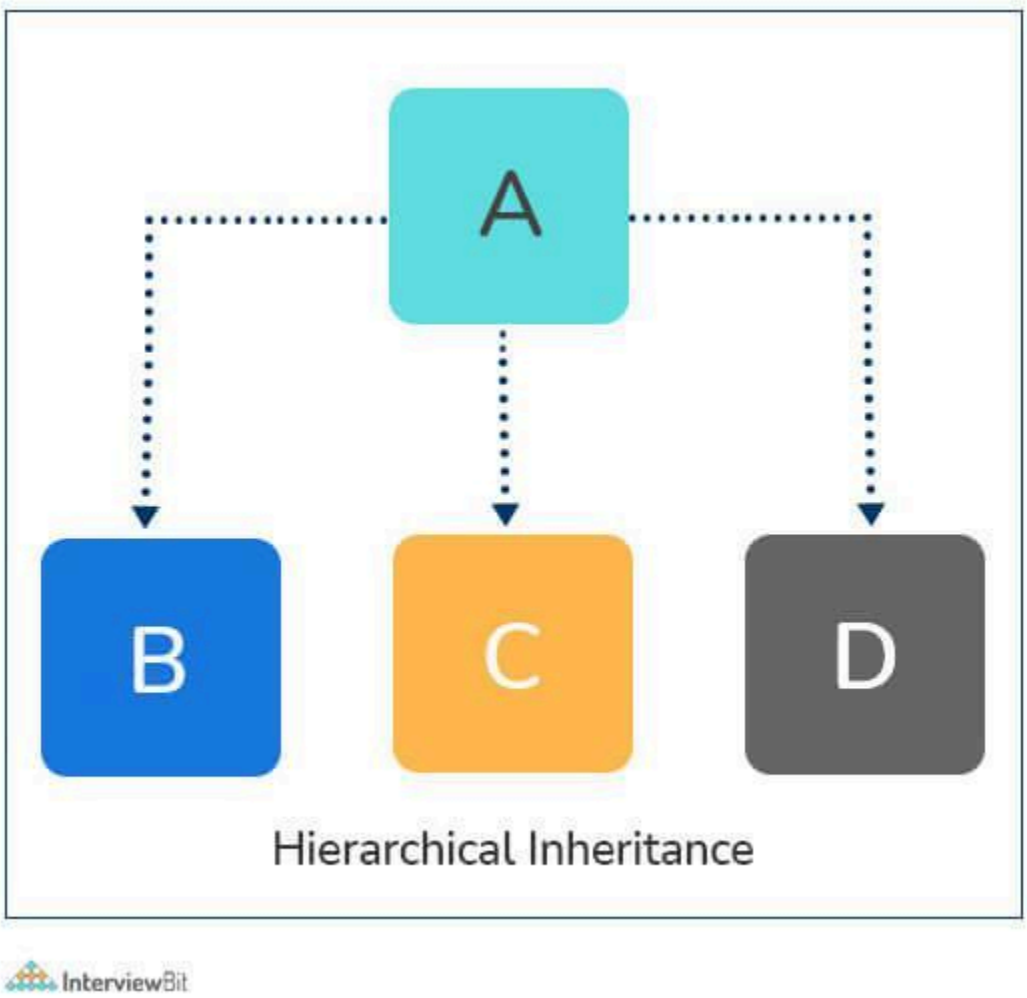


InterviewBit

```
# Parent class1
class Parent1:
  def parent1_func(self):
     print("Hi I am first Parent")

# Parent class2
class Parent2:
  def parent2_func(self):
     print("Hi I am second Parent")

# Child class
class Child(Parent1, Parent2):
```

```
  def child_func(self):
      self.parent1_func()
      self.parent2_func()

# Driver's code
obj1 = Child()
obj1.child_func()
```

- **Hierarchical Inheritance:** When a parent class is derived by more than one child class, it is called hierarchical inheritance.




```
# Base class
class A:
    def a_func(self):
        print("I am from the parent class.")
```

```
# 1st Derived class
class B(A):
    def b_func(self):
        print("I am from the first child.")

# 2nd Derived class
class C(A):
    def c_func(self):
        print("I am from the second child.")

# Driver's code
obj1 = B()
obj2 = C()
obj1.a_func()
obj1.b_func()    #child 1 method
obj2.a_func()
obj2.c_func()    #child 2 method
```

42. How do you access parent members in the child class?

Following are the ways using which you can access parent class members within a child class:

- **By using Parent class name:** You can use the name of the parent class to access the attributes as shown in the example below:

```
class Parent(object):
  # Constructor
  def __init__(self, name):
      self.name = name

class Child(Parent):
  # Constructor
  def __init__(self, name, age):
      Parent.name = name
      self.age = age

  def display(self):
      print(Parent.name, self.age)

# Driver Code
obj = Child("Interviewbit", 6)
obj.display()
```

- **By using super():** The parent class members can be accessed in child class using the super keyword.

```
class Parent(object):
  # Constructor
  def __init__(self, name):
     self.name = name

class Child(Parent):
  # Constructor
  def __init__(self, name, age):
     '''
     In Python 3.x, we can also use super().__init__(name)
     '''
     super(Child, self).__init__(name)
     self.age = age

  def display(self):
    # Note that Parent.name cant be used
    # here since super() is used in the constructor
    print(self.name, self.age)

# Driver Code
obj = Child("Interviewbit", 6)
obj.display()
```

43. Are access specifiers used in python?

Python does not make use of access specifiers specifically like private, public, protected, etc. However, it does not deprive this to any variables. It has the concept of imitating the behaviour of variables by making use of a single (protected) or double underscore (private) as prefixed to the variable names. By default, the variables without prefixed underscores are public.

Example:

```
# to demonstrate access specifiers
class InterviewbitEmployee:

   # protected members
   _emp_name = None
   _age = None

   # private members
   __branch = None

   # constructor
   def __init__(self, emp_name, age, branch):
```

```
        self._emp_name = emp_name
        self._age = age
        self.__branch = branch

    #public member
    def display():
        print(self._emp_name +" "+self._age+" "+self.__branch)
```

44. Is it possible to call parent class without its instance creation?

Yes, it is possible if the base class is instantiated by other child classes or if the base class is a static method.

45. How is an empty class created in python?

An empty class does not have any members defined in it. It is created by using the pass keyword (the pass command does nothing in python). We can create objects for this class outside the class. For example-

```
class EmptyClassDemo:
  pass
obj=EmptyClassDemo()
obj.name="Interviewbit"
print("Name created= ",obj.name)
```

Output:
Name created = Interviewbit

46. Differentiate between new and override modifiers.

The new modifier is used to instruct the compiler to use the new implementation and not the base class function. The Override modifier is useful for overriding a base class function inside the child class.

47. Why is finalize used?

Finalize method is used for freeing up the unmanaged resources and clean up before the garbage collection method is invoked. This helps in performing memory management tasks.

48. What is init method in python?

The **init** method works similarly to the constructors in Java. The method is run as soon as an object is instantiated. It is useful for initializing any attributes or default behaviour of the object at the time of instantiation.
For example:

```
class InterviewbitEmployee:
```

```
  # init method / constructor
  def __init__(self, emp_name):
     self.emp_name = emp_name

  # introduce method
  def introduce(self):
     print('Hello, I am ', self.emp_name)

emp = InterviewbitEmployee('Mr Employee')     # __init__ method is called here and initializes the object name with "Mr Employee"
emp.introduce()
```

49. How will you check if a class is a child of another class?

This is done by using a method called **issubclass()** provided by python. The method tells us if any class is a child of another class by returning true or false accordingly.
For example:

```
class Parent(object):
  pass

class Child(Parent):
  pass

# Driver Code
print(issubclass(Child, Parent))    #True
print(issubclass(Parent, Child))    #False
```

- We can check if an object is an instance of a class by making use of **isinstance()** method:

```
obj1 = Child()
obj2 = Parent()
print(isinstance(obj2, Child))    #False
print(isinstance(obj2, Parent))   #True
```

Python Pandas Interview Questions

50. What do you know about pandas?

- Pandas is an open-source, python-based library used in data manipulation applications requiring high performance. The name is derived from “Panel Data” having multidimensional data. This was developed in 2008 by Wes McKinney and was developed for data analysis.
- Pandas are useful in performing 5 major steps of data analysis - Load the data, clean/manipulate it, prepare it, model it, and analyze the data.

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

51. Define pandas dataframe.

A dataframe is a 2D mutable and tabular structure for representing data labelled with axes - rows and columns.
The syntax for creating dataframe:

```
import pandas as pd
dataframe = pd.DataFrame( data, index, columns, dtype)
```

where:

- data - Represents various forms like series, map, ndarray, lists, dict etc.
- index - Optional argument that represents an index to row labels.
- columns - Optional argument for column labels.
- Dtype - the data type of each column. Again optional.

52. How will you combine different pandas dataframes?
The dataframes can be combines using the below approaches:

- **append() method**: This is used to stack the dataframes horizontally. Syntax:

```
df1.append(df2)
```

- **concat() method:** This is used to stack dataframes vertically. This is best used when the dataframes have the same columns and similar fields. Syntax:

```
pd.concat([df1, df2])
```

- **join() method:** This is used for extracting data from various dataframes having one or more common columns.

```
df1.join(df2)
```

53. Can you create a series from the dictionary object in pandas?
One dimensional array capable of storing different data types is called a series. We can create pandas series from a dictionary object as shown below:

```
import pandas as pd
dict_info = {'key1' : 2.0, 'key2' : 3.1, 'key3' : 2.2}
series_obj = pd.Series(dict_info)
print (series_obj)
Output:
x    2.0
y    3.1
z    2.2
dtype: float64
```

If an index is not specified in the input method, then the keys of the dictionaries are sorted in ascending order for constructing the index. In case the index is passed, then values of the index label will be extracted from the dictionary.

54. How will you identify and deal with missing values in a dataframe?

We can identify if a dataframe has missing values by using the isnull() and isna() methods.

```
missing_data_count=df.isnull().sum()
```

We can handle missing values by either replacing the values in the column with 0 as follows:

```
df[‘column_name’].fillna(0)
```

Or by replacing it with the mean value of the column

```
df[‘column_name’] = df[‘column_name’].fillna((df[‘column_name’].mean()))
```

55. What do you understand by reindexing in pandas?

Reindexing is the process of conforming a dataframe to a new index with optional filling logic. If the values are missing in the previous index, then NaN/NA is placed in the location. A new object is returned unless a new index is produced that is equivalent to the current one. The copy value is set to False. This is also used for changing the index of rows and columns in the dataframe.

56. How to add new column to pandas dataframe?

A new column can be added to a pandas dataframe as follows:

```
import pandas as pd
data_info = {'first' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'second' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(data_info)
#To add new column third
df['third']=pd.Series([10,20,30],index=['a','b','c'])
print (df)
#To add new column fourth
df['fourth']=df['first']+info['third']
print (df)
```

57. How will you delete indices, rows and columns from a dataframe?

To delete an Index:

- Execute del df.index.name for removing the index by name.
- Alternatively, the df.index.name can be assigned to None.
- For example, if you have the below dataframe:

```
        Column 1
Names
John        1
Jack        2
Judy        3
```

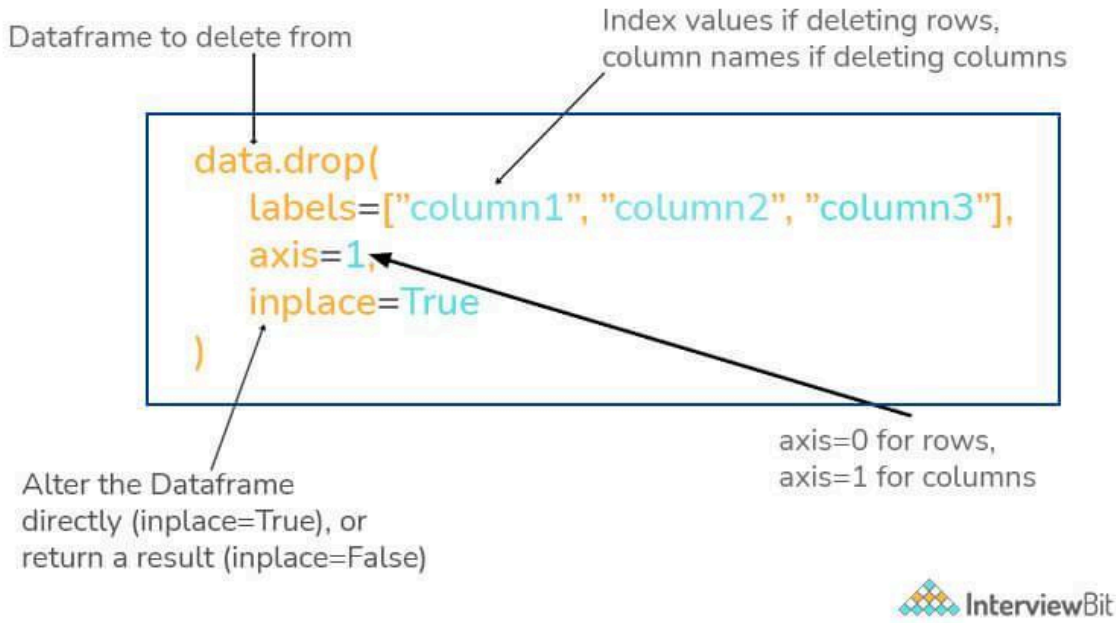
```
Jim          4
```

- To drop the index name "Names":

```
df.index.name = None
# Or run the below:
# del df.index.name
print(df)
      Column 1
John        1
Jack        2
Judy        3
Jim         4
```

To delete row/column from dataframe:

- drop() method is used to delete row/column from dataframe.
- The axis argument is passed to the drop method where if the value is 0, it indicates to drop/delete a row and if 1 it has to drop the column.
- Additionally, we can try to delete the rows/columns in place by setting the value of inplace to True. This makes sure that the job is done without the need for reassignment.
- The duplicate values from the row/column can be deleted by using the drop_duplicates() method.



58. Can you get items of series A that are not available in another series B?

This can be achieved by using the `~` (not/negation symbol) and `isin()` method as shown below.

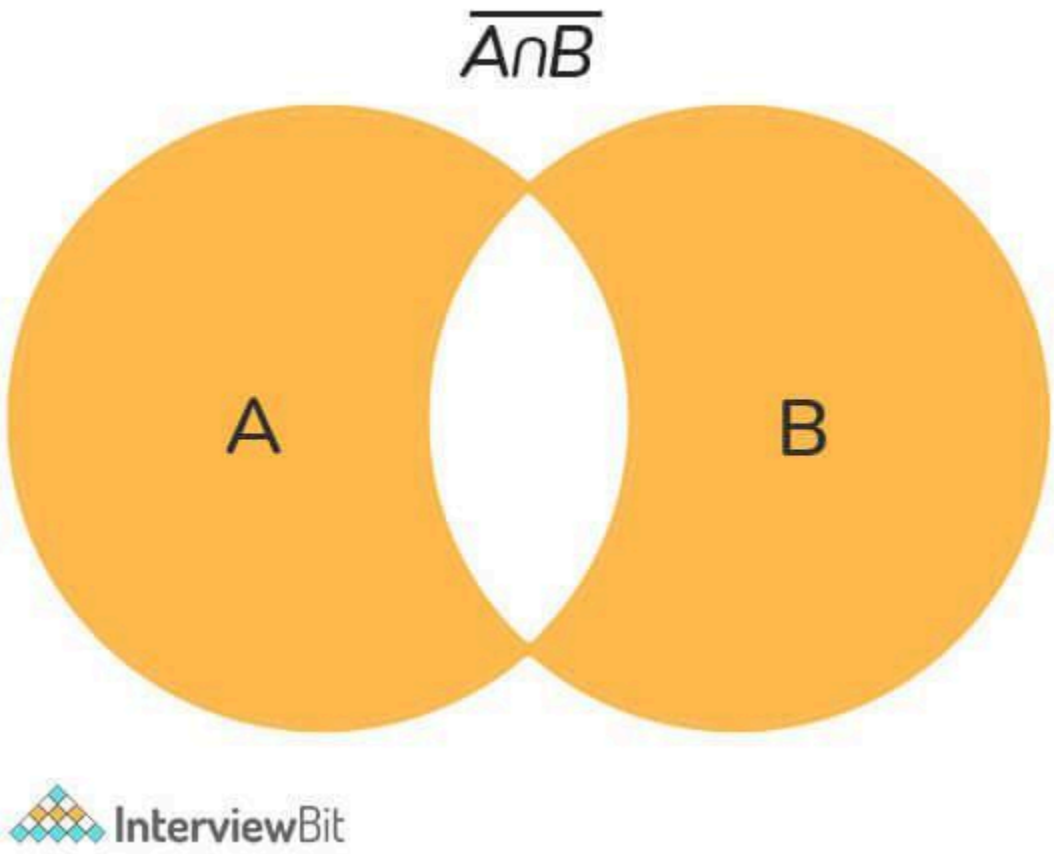
```
import pandas as pd
df1 = pd.Series([2, 4, 8, 10, 12])
df2 = pd.Series([8, 12, 10, 15, 16])
df1=df1[~df1.isin(df2)]
print(df1)
"""
Output:
0    2
1    4
dtype: int64
"""
```

59. How will you get the items that are not common to both the given series A and B?

We can achieve this by first performing the union of both series, then taking the intersection of both series. Then we follow the approach of getting items of union that are not there in the list of the intersection.

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra — Course Code: CSE11410
7. Course : CSE — L: 3
8. Program : BCA — T: 0
9. Target : 60% — P: 0
C: 3



The following code demonstrates this:

```
import pandas as pd
import numpy as np
df1 = pd.Series([2, 4, 5, 8, 10])
df2 = pd.Series([8, 10, 13, 15, 17])
p_union = pd.Series(np.union1d(df1, df2))  # union of series
p_intersect = pd.Series(np.intersect1d(df1, df2))  # intersection of series
unique_elements = p_union[~p_union.isin(p_intersect)]
print(unique_elements)
"""
Output:
0    2
1    4
2    5
5   13
```

```
6    15
7    17
dtype: int64
"""
```

60. While importing data from different sources, can the pandas library recognize dates?

Yes, they can, but with some bit of help. We need to add the parse_dates argument while we are reading data from the sources. Consider an example where we read data from a CSV file, we may encounter different date-time formats that are not readable by the pandas library. In this case, pandas provide flexibility to build our custom date parser with the help of lambda functions as shown below:

```
import pandas as pd
from datetime import datetime
dateparser = lambda date_val: datetime.strptime(date_val, '%Y-%m-%d %H:%M:%S')
df = pd.read_csv("some_file.csv", parse_dates=['datetime_column'], date_parser=dateparser)
```

Numpy Interview Questions

61. What do you understand by NumPy?

NumPy is one of the most popular, easy-to-use, versatile, open-source, python-based, general-purpose package that is used for processing arrays. NumPy is short for NUMerical PYthon. This is very famous for its highly optimized tools that result in high performance and powerful N-Dimensional array processing feature that is designed explicitly to work on complex arrays. Due to its popularity and powerful performance and its flexibility to perform various operations like trigonometric operations, algebraic and statistical computations, it is most commonly used in performing scientific computations and various broadcasting functions. The following image shows the applications of NumPy:

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Previous Year Question Paper

ADAMAS UNIVERSITY
SCHOOL OF ENGINEERING AND TECHNOLOGY
END-SEMESTER EXAMINATION: JULY 2020

Name of the Program: MCA Semester: I Stream: CSE
PAPER TITLE: COMPUTER PROGRAMMING WITH PYTHON
PAPER CODE: **CSE21908**
Maximum Marks: 50 Time duration: 3 hours
Total No of questions: 12 Total No of Pages: 01

Instruction for the Candidate:

1. At top sheet, clearly mention Name, Univ. Roll No., Enrolment No., Paper Name & Code, and Date of Exam.
2. All parts of a Question should be answered consecutively. Each Answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

Section A (Answer All the Questions) (5 x 2 = 10)			
1.	**Describe** the Identifiers, Keywords and Variables in Python programming language with examples.	**U**	**CO1**
2.	**Explain** the basic data types available in Python with examples.	**Evaluate**	**CO1**
3.	**Describe** the difference between set and list datatype.	**U**	**CO1**
4.	**Explain** how slicing operator used on string datatype.	**Evaluate**	**CO2**
5.	**Describe** why strings are immutable with an example.	**U**	**CO2**
	SECTION B (Attempt any **Three Questions) (3 x 5 = 15)**		
6.	**Write** Python program to find the GCD of two positive numbers.	**Ap**	**CO1**
7.	**Examine** whether the given string is a Palindrome or not using slicing.	**Ap**	**CO2**
8.	**Describe** the various file opening mode in Python language.	**U**	**CO6**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

9.	**Describe** with Example: i) try catch block ii) function calling	**U**	**CO3, CO5**
	SECTION C (Answer Any Two Questions) (2 x 12.5 = 25)		
10.	**Write** Pythonic code to sort a sequence of names according to their alphabetical order without using sort () function.	**Ap**	**CO2**
11.	Consider a Rectangle Class and Create Two Rectangle Objects. **Write** Python program to Check Whether the Area of the First Rectangle is Greater than Second by Overloading > Operator.	**Ap**	**CO4**
12.	**Describe** the advantage of functions in Python. **Describe** the scope and lifetimes of Global & Local variables.	**U**	**CO3**

Question Bank Sample

School: SOET **Department: CSE**
Course Code: CSE11006 **Course Name: Python Programming**
Program: B.Tech **Semester: III**

UNIT-I

Sl. No.	Question	Level of Difficulty (Easy/ Medium/ Difficult)	Knowledge Level (Bloom's Taxonomy)	Course Outcome (CO)

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Part A (Multiple Choice Questions) (1 mark each)				
1.	Who developed Python	**Easy**	**R**	**CO1**
2.	**What is the maximum possible length of idenifier**	**Medium**	**R**	**CO1**
3.	**In which language Python is written**	**Difficult**	**R**	**CO1**
	Part B (Definition/Naming Questions) (2 marks each)			
1.	What is the benefit of using Python	**Easy**	**R**	**CO1**
2.	Why Python is adopted now a days	**Medium**	U	**CO2**
3.	What is strongly typed language	**Difficult**	R	**CO1**
Part C (Short Questions) (3-4 marks each)				
1.	Differentiate between static and dynamic typed language	**Easy**	R	**CO1**
2.	Differentiate between local and global scope in python	**Medium**	**U**	**CO3**
3.	Analyze the common built in data type in Python	**Difficult**	**U**	**CO1**
Part D (Explanation Based Questions) (5 marks each)				
1.	Give an example of tuple in Python and is it further changeable or unchangeable suggest your answer.	**Easy**	**Ap**	**CO1**
2.	What is the usage of composite function Give with an example.	**Medium**	**U**	**CO1**
3.	In This tuple, Fruits= ("apple", "banana", "cherry", "apple", "cherry"),If we want to add another fruit “jack fruit” what is the code for it,If we want to delete “cherry” what is the code for it.	**Difficult**	**U**	**CO1**
Part E (Questions Based on Reasoning) (5 marks each)				
1.	NA	**Easy**		
2.	NA	**Medium**		

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

3.	NA	**Difficult**		
Part F (Application Based Questions) (5-10 marks each)				
1.	What is the usage of -self- and -init- in python.	**Easy**	**R**	**CO1**
2.	Write a program in Python to check a number is prime or not prime.	**Medium**	**U**	**CO2**
3.	Write a program in Python to check a number even or not.	**Difficult**	**AP**	**CO3**
Part G (Short Notes) (5 marks each)				
1.	How function is defined in Python design a function to add,multiply 2 numbers.	**Easy**	**R**	**CO1**
2.	How exponentiation and modular division operator is used in Python suggest your answer	**Medium**	**U**	**CO2**
3.	Write a program in Python a person is eligible for vote or not(Voting age in India is 18 years or above)	**Difficult**	**AP**	**CO3**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

School: SOET
Department: CSE
Course Code: CSE11006
Course Name: Python Programming
Program: B.Tech
Semester: III

UNIT-II

Sl. No.	Question	Level of Difficulty (Easy/ Medium/ Difficult)	Knowledge Level (Bloom's Taxonomy)	Course Outcome (CO)
	Part A (Multiple Choice Questions) (1 mark each)			
1.	Try or val which one is key word in python	**Easy**	**U**	**CO1**
2.	Can we start with under score in Python for identifier declaration.	**Medium**	**R**	**CO2**
3.	X y z p = 1000,xyzp-234 which variable declaration is wrong in python suggest your answer	**Difficult**	**A**	**CO3**
	Part B (Definition/Naming Questions) (2 marks each)			
1.	Why Indentation is needed in Python.	**Easy**	U	**CO1**
2.	What is the need of "break" statement in Python	**Medium**	**A**	**CO2**
3.	Why "Continue" is required in python loop.	**Difficult**	**Ap**	**CO4**
	Part C (Short Questions) (3-4 marks each)			
1.	Write a program,in Python to print 1 to 34 by using for loop.	**Easy**	**U**	**CO1**
2.	If a persons mark is greater than 30 then person will pass other wise not Write it with Python.	**Medium**	**R**	**CO2**
3.	Perform the logical AND operation between A=5,B=9 Write Python code for it.	**Difficult**	**A**	**CO3**
	Part D (Explanation Based Questions) (5 marks each)			

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

1.	How casting in python is done Discuss with an example, Define how to check a List length in Python.	**Easy**	**U**	**CO1**
2.	Write a program in Python to check a number is even or not.	**Medium**	**R**	**CO2**
3.	Write a program in Python to find factorial of a number.	**Difficult**	**A**	**CO3**
Part E (Questions Based on Reasoning) (5 marks each)				
1.	NA	**Easy**	U	**CO1**
2.	NA	**Medium**	**A**	**CO2**
3.	NA	**Difficult**	**Ap**	**CO4**
Part F (Application Based Questions) (5-10 marks each)				
1.	Write a program in Python to check Boolean logic.and bit wise OR operation between 2 number.	**Easy**	U	**CO1**
2.	What is the use of Lambda function in Python and how it is wrapped inside a function give an example.	**Medium**	**A**	**CO2**
3.	Write a program in Python to convert Celsisus to farenheit.	**Difficult**	**Ap**	**CO4**
Part G (Short Notes) (5 marks each)				
1.	Write a Program in Python to check Floor value of a number,and how to convert INT to Float in type casting.	**Easy**	U	**CO1**
2.	What is the difference between range and X range python discuss with an example?	**Medium**	**A**	**CO2**
3.	Difference between pickling and unpickling in Python.	**Difficult**	**Ap**	**CO4**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** — **Course Code: CSE11410**
7. **Course : CSE** — **L: 3**
8. **Program : BCA** — **T: 0**
9. **Target : 60%** — **P: 0**
   C: 3

School: SOET — **Department: CSE**
Course Code: CSE11006 — **Course Name: Python Programming**
Program: B.Tech — **Semester: III**

UNIT-III

Sl. No.	Question	Level of Difficulty (Easy/ Medium/ Difficult)	Knowledge Level (Bloom's Taxonomy)	Course Outcome (CO)
Part A (Multiple Choice Questions) (1 mark each)				
1.	What is the Use of Tuple in Python	**Easy**	U	**CO1**
2.	Which of the Python operators is used for power(a,b),	**Medium**	**A**	**CO2**
3.	Which has highest precedence in Python Exponentiation or division.	**Difficult**	**Ap**	**CO4**
	Part B (Definition/Naming Questions) (2 marks each)			
1.	Print() or val() which has built in function in Python.	**Easy**	U	**CO1**
2.	What is the use of len() in set.	**Medium**	**A**	**CO2**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

3.	How we can find common element between 2 set ,use the Logic A=(9,8,76,12,110),B=(110,12,55,03,14)	**Difficult**	**Ap**	**CO4**
Part C (Short Questions) (3-4 marks each)				
1.	What is the role of dir()	**Easy**	U	**CO1**
2.	Explain an interpreted language.	**Medium**	**A**	**CO2**
3.	What is the use of help() and dir() functions?	**Difficult**	**Ap**	**CO4**
Part D (Explanation Based Questions) (5 marks each)				
1.	Explain a SET Union ,difference and Intersection operation For A=(3,5,67,78,87,9,120), B=(45,67,78,9,888,43,32)	**Easy**	U	**CO1**
2.	Explain split() and join() functions in Python**?**	**Medium**	**A**	**CO2**
3.	Write a program in python to generate a Fibonacci series.	**Difficult**	**Ap**	**CO4**
Part E (Questions Based on Reasoning) (5 marks each)				
1.	Write a Program In python to add,delete, and update an item to List,by taking an example,	**Easy**	U	**CO1**
2.	How to Define a matrix in python.	**Medium**	**A**	**CO2**
3.	Design a python code for addition ,subtraction and multiplication of 2 matrx .	**Difficult**	**Ap**	**CO4**
Part F (Application Based Questions) (5-10 marks each)				
1.	The disease and symptom data table is given as below,Create a Python dictionary for this, add the symptom for HEART ATTACK,ACUTE PAIN IN HEART. Delete the MALARIA SYMPTOM FROM THE Dictionary. DISEASE | SYMPTOM MALARIA | FEVER	**Easy**	U	**CO1**

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

	COVID19 / ACUTE SHORE THROAT COMMON COLD / SNEEZE MIGRANE / HEADACHE DYSENTRY / ABDOMENAL PAIN			
2.	The string is "welcome to India" slice it from 1st to 5th position i. What is the value for [3:5] ii. What is the value for [4:6] iii. What is the value for [2:7] iv. What is the value for [3:6]	**Medium**	**A**	**CO2**
3.	The string is "India has a glorious past"In this string search for word"enigmatic" and "glorious" in Python. i. How to concatenate 2 strings in Python a=(mother),b=(earth) How to use format () ,age = 36 ii. txt = "My name is John, and I am {}"	**Difficult**	**Ap**	**CO4**
Part G (Short Notes) (5 marks each)				
1.	By using NUMPY operation create a 1-D and 2-D array in Python.	**Easy**	U	**CO1**
2.	What is the use of Try and final key word in Python describe with an example.	**Medium**	**A**	**CO2**
3.	What is the use of recursion in Python. What is the use of inheritance in Python.	**Difficult**	**Ap**	**CO4**

Roll Number	**Registration Number**	**Name of the Student**	**Internal Assessment (30)**				
			Assignment	**Class Test**	**Case Study**	**etc.**	**Total**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra **Course Code: CSE11410**
7. Course : CSE **L: 3**
8. Program : BCA **T: 0**
9. Target : 60% **P: 0**
C: 3

UG/02/BCA/2020/001	AU/2020/0004253	DEBOJYOTI SAHA	7	14			21
UG/02/BCA/2020/002	AU/2020/0004290	AZMAT ALI	9	20			29
UG/02/BCA/2020/037	AU/2020/0005526	Oliva Dutta	8	19			27
UG/02/BCA/2020/005	AU/2020/0004453	SAYANTAN JANA	9	20			29
UG/02/BCA/2020/006	AU/2020/0004457	SANJUKTA JANA	7	18			25
UG/02/BCA/2020/007	AU/2020/0004458	AYAN RAHAMAN	6	17			23
UG/02/BCA/2020/008	AU/2020/0004461	SUSOVON NANDY	7	15			22
UG/02/BCA/2020/009	AU/2020/0004478	Hritankar Das	8	14			22
UG/02/BCA/2020/010	AU/2020/0004482	SWARNAMOY GHOSH	6	14			20
UG/02/BCA/2020/011	AU/2020/0004483	ANWESHA PRAMANIK	6	13			19
UG/02/BCA/2020/012	AU/2020/0004492	Swapnil Mitra	7	14			21
UG/02/BCA/2020/013	AU/2020/0004496	Suman Ghosh	9	20			29
UG/02/BCA/2020/015	AU/2020/0004498	Anthony Prakash Rozario	8	19			27
UG/02/BCA/2020/016	AU/2020/0004501	MOUSUMI DUTTA	9	20			29
UG/02/BCA/2020/017	AU/2020/0004504	Dhrubajyoti Dey	7	18			25
UG/02/BCA/2020/018	AU/2020/0004507	Pritam Hore	6	17			23
UG/02/BCA/2020/019	AU/2020/0004509	Aratrika Bose	7	15			22
UG/02/BCA/2020/020	AU/2020/0004510	Tithi Paul	8	14			22
UG/02/BCA/2020/021	AU/2020/0004513	Arpan Mondal	6	14			20
UG/02/BCA/2020/022	AU/2020/0004514	Parichoy nandi	7	14			21
UG/02/BCA/2020/023	AU/2020/0004515	Aditya Jaman	9	20			29

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

UG/02/BCA/2020/024	AU/2020/0004517	Aparesh Muhuri	8	19			27
UG/02/BCA/2020/025	AU/2020/0004520	Kosturi Mondal	9	20			29
UG/02/BCA/2020/026	AU/2020/0004522	Aritra Das	7	18			25
UG/02/BCA/2020/028	AU/2020/0004526	Neelash Saha	6	17			23
UG/02/BCA/2020/029	AU/2020/0004533	Bittaswer Ghosh	7	15			22
UG/02/BCA/2020/030	AU/2020/0004535	SUNEET CHOUDHARY	8	14			22
UG/02/BCA/2020/031	AU/2020/0004543	Abhishek Tarafdar	6	14			20
UG/02/BCA/2020/032	AU/2020/0004547	Ayon Chakraborty	6	13			19
UG/02/BCA/2020/033	AU/2020/0004552	JYOTISHKA DE	7	14			21
UG/02/BCA/2020/034	AU/2020/0004564	Asmat Sk	9	20			29
UG/02/BCA/2020/035	AU/2020/0004575	Nikhil Kumar Sah	8	19			27
UG/02/BCA/2020/036	AU/2020/0004582	Suprita Nandy	9	20			29
UG/02/BCA/2020/003	AU/2020/0004448	SATYAJIT GHOSH	7	18			25
UG/02/BCA/2020/004	AU/2020/0004449	DEBDYUTI DAS	6	17			23
UG/02/BCA/2020/027	AU/2020/0004525	RISHI BARUA	7	15			22
UG/02/BCABFSI/2020/001	AU/2020/0004505	Somnath Gayen	8	14			22
UG/02/BCABFSI/2020/002	AU/2020/0004598	BARUN RAJBHAR	6	14			20
UG/02/BCABFSI/2020/003	AU/2020/0004605	RAKIBUL ISLAM	6	13			19
UG/02/BCAGA/2020/006	AU/2020/0004497	Abhishek Mondal	7	14			21
UG/02/BCAGA/2020/002	AU/2020/0004500	Arka Mitra	6	14			20
UG/02/BCAGA/2020/003	AU/2020/0004524	Sourav Mondal	6	13			19

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

UG/02/BCAGA/2020/004	AU/2020/0004539	Ranita Bagchi	7	14			21
UG/02/BCAGA/2020/005	AU/2020/0004568	SUBHAJIT SIRCAR	6	14			20
UG/02/BCAGA/2020/001	AU/2020/0004493	Susmit Shaw	6	14			20
UG/02/BCAGA/2020/002	AU/2020/0004253	Arka Mitra	6	13			19
UG/02/BCAGA/2020/003	AU/2020/0004290	Sourav Mondal	7	14			21
UG/02/BCAGA/2020/004	AU/2020/0005526	Ranita Bagchi	6	14			20

Evaluation Sheet – Internal Assessment

Signature of HOD/Dean **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Evaluation Sheet – Mid Semester

Roll Number	Registration Number	Name of the Student	Marks (20)
UG/02/BCA/2020/001	AU/2020/0004253	DEBOJYOTI SAHA	
UG/02/BCA/2020/002	AU/2020/0004290	AZMAT ALI	
UG/02/BCA/2020/037	AU/2020/0005526	Oliva Dutta	
UG/02/BCA/2020/005	AU/2020/0004453	SAYANTAN JANA	
UG/02/BCA/2020/006	AU/2020/0004457	SANJUKTA JANA	
UG/02/BCA/2020/007	AU/2020/0004458	AYAN RAHAMAN	
UG/02/BCA/2020/008	AU/2020/0004461	SUSOVON NANDY	
UG/02/BCA/2020/009	AU/2020/0004478	Hritankar Das	
UG/02/BCA/2020/010	AU/2020/0004482	SWARNAMOY GHOSH	
UG/02/BCA/2020/011	AU/2020/0004483	ANWESHA PRAMANIK	
UG/02/BCA/2020/012	AU/2020/0004492	Swapnil Mitra	
UG/02/BCA/2020/013	AU/2020/0004496	Suman Ghosh	
UG/02/BCA/2020/015	AU/2020/0004498	Anthony Prakash Rozario	
UG/02/BCA/2020/016	AU/2020/0004501	MOUSUMI DUTTA	
UG/02/BCA/2020/017	AU/2020/0004504	Dhrubajyoti Dey	
UG/02/BCA/2020/018	AU/2020/0004507	Pritam Hore	
UG/02/BCA/2020/019	AU/2020/0004509	Aratrika Bose	
UG/02/BCA/2020/020	AU/2020/0004510	Tithi Paul	
UG/02/BCA/2020/021	AU/2020/0004513	Arpan Mondal	
UG/02/BCA/2020/022	AU/2020/0004514	Parichoy nandi	
UG/02/BCA/2020/023	AU/2020/0004515	Aditya Jaman	
UG/02/BCA/2020/024	AU/2020/0004517	Aparesh Muhuri	
UG/02/BCA/2020/025	AU/2020/0004520	Kosturi Mondal	

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Signature of HOD/Dean **Signature of Faculty**

Date: **Date:**

UG/02/BCA/2020/026	AU/2020/0004522	Aritra Das	
UG/02/BCA/2020/028	AU/2020/0004526	Neelash Saha	
UG/02/BCA/2020/029	AU/2020/0004533	Bittaswer Ghosh	
UG/02/BCA/2020/030	AU/2020/0004535	SUNEET CHOUDHARY	
UG/02/BCA/2020/031	AU/2020/0004543	Abhishek Tarafdar	
UG/02/BCA/2020/032	AU/2020/0004547	Ayon Chakraborty	
UG/02/BCA/2020/033	AU/2020/0004552	JYOTISHKA DE	
UG/02/BCA/2020/034	AU/2020/0004564	Asmat Sk	
UG/02/BCA/2020/035	AU/2020/0004575	Nikhil Kumar Sah	
UG/02/BCA/2020/036	AU/2020/0004582	Suprita Nandy	
UG/02/BCA/2020/003	AU/2020/0004448	SATYAJIT GHOSH	
UG/02/BCA/2020/004	AU/2020/0004449	DEBDYUTI DAS	
UG/02/BCA/2020/027	AU/2020/0004525	RISHI BARUA	
UG/02/BCABFSI/2020/001	AU/2020/0004505	Somnath Gayen	
UG/02/BCABFSI/2020/002	AU/2020/0004598	BARUN RAJBHAR	
UG/02/BCABFSI/2020/003	AU/2020/0004605	RAKIBUL ISLAM	
UG/02/BCAGA/2020/006	AU/2020/0004497	Abhishek Mondal	
UG/02/BCAGA/2020/002	AU/2020/0004500	Arka Mitra	
UG/02/BCAGA/2020/003	AU/2020/0004524	Sourav Mondal	
UG/02/BCAGA/2020/004	AU/2020/0004539	Ranita Bagchi	
UG/02/BCAGA/2020/005	AU/2020/0004568	SUBHAJIT SIRCAR	
UG/02/BCAGA/2020/001	AU/2020/0004493	Susmit Shaw	

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Planning for Remedial Classes – Mid Semester

Sl. No.	Name of Student	Roll No.	Reg. No.	Mid Sem Marks	Remedial Classes Held							Class test on the basis of Remedial Classes	End Sem Marks	Improvement (Y/N)
					Date									
					Venue									
					Time									
1.														
2.														

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Signature of HOD/ Dean **Signature of Faculty**

Date: **Date:**

COURSE END SURVEY

INDIRECT ASSESSMENT

Sample format for Indirect Assessment of Course outcomes:

NAME:
ROLL NO.:
REG. NO..:
COURSE:
PROGRAM:

Please rate the following aspects of course outcomes of

Use the scale 1-5 (Poor – Excellent)

Course Outcomes	Statement	1	2	3	4	5
CO1						

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

CO2						
CO3						
CO4						
CO5						

INDIRECT ASSESSMENT CONSOLIDATION

ADAMAS UNIVERSITY, KOLKATA SCHOOL OF DEPARTMENT OF CO Indirect Assessment		
Programme: Batch: 2020-22		Academic Year:2020-21
Course Code & Name:		
Course Outcome	Students Feed Back (5)	Attainment (100)
CO1		
CO2		
CO3		
CO4		
CO5		
etc.		
Signature of HOD/Dean Date:		Signature of Faculty Date:

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Evaluation Sheet (End Semester)

Roll Number	Registration Number	Name of the Student	Marks (50)

Signature of HOD/Dean

Date:

Signature of Faculty

Date:

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Planning for Remedial Classes – End Semester

Sl. No.	Name of Student	Roll No.	Reg. No.	End Sem Marks	Remedial Classes Held							Class test on the basis of Remedial Classes	Supple Exam Marks	Improvement (Y/N)
					Date									
					Venue									
					Time									
1.														
2.														

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Signature of HOD/ Dean

Signature of Faculty

Date

Date

Consolidated Mark Statement

Roll Number	Registration Number	Name of the Student	Total Marks			
			Mid Semester (20)	Internal Assessment (30)	End Semester (50)	Total (100)
UG/02/BCA/2020/001	AU/2020/0004253	DEBOJYOTI SAHA				
UG/02/BCA/2020/002	AU/2020/0004290	AZMAT ALI				
UG/02/BCA/2020/037	AU/2020/0005526	Oliva Dutta				
UG/02/BCA/2020/005	AU/2020/0004453	SAYANTAN JANA				
UG/02/BCA/2020/006	AU/2020/0004457	SANJUKTA JANA				
UG/02/BCA/2020/007	AU/2020/0004458	AYAN RAHAMAN				
UG/02/BCA/2020/008	AU/2020/0004461	SUSOVON NANDY				
UG/02/BCA/2020/009	AU/2020/0004478	Hritankar Das				
UG/02/BCA/2020/010	AU/2020/0004482	SWARNAMOY GHOSH				

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

UG/02/BCA/2020/011	AU/2020/0004483	ANWESHA PRAMANIK				
UG/02/BCA/2020/012	AU/2020/0004492	Swapnil Mitra				
UG/02/BCA/2020/013	AU/2020/0004496	Suman Ghosh				
UG/02/BCA/2020/015	AU/2020/0004498	Anthony Prakash Rozario				
UG/02/BCA/2020/016	AU/2020/0004501	MOUSUMI DUTTA				
UG/02/BCA/2020/017	AU/2020/0004504	Dhrubajyoti Dey				
UG/02/BCA/2020/018	AU/2020/0004507	Pritam Hore				
UG/02/BCA/2020/019	AU/2020/0004509	Aratrika Bose				
UG/02/BCA/2020/020	AU/2020/0004510	Tithi Paul				
UG/02/BCA/2020/021	AU/2020/0004513	Arpan Mondal				
UG/02/BCA/2020/022	AU/2020/0004514	Parichoy nandi				
UG/02/BCA/2020/023	AU/2020/0004515	Aditya Jaman				
UG/02/BCA/2020/024	AU/2020/0004517	Aparesh Muhuri				
UG/02/BCA/2020/025	AU/2020/0004520	Kosturi Mondal				
UG/02/BCA/2020/026	AU/2020/0004522	Aritra Das				
UG/02/BCA/2020/028	AU/2020/0004526	Neelash Saha				
UG/02/BCA/2020/029	AU/2020/0004533	Bittaswer Ghosh				
UG/02/BCA/2020/030	AU/2020/0004535	SUNEET CHOUDHARY				
UG/02/BCA/2020/031	AU/2020/0004543	Abhishek Tarafdar				
UG/02/BCA/2020/032	AU/2020/0004547	Ayon Chakraborty				

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

UG/02/BCA/2020/033	AU/2020/0004552	JYOTISHKA DE				
UG/02/BCA/2020/034	AU/2020/0004564	Asmat Sk				
UG/02/BCA/2020/035	AU/2020/0004575	Nikhil Kumar Sah				
UG/02/BCA/2020/036	AU/2020/0004582	Suprita Nandy				
UG/02/BCA/2020/003	AU/2020/0004448	SATYAJIT GHOSH				
UG/02/BCA/2020/004	AU/2020/0004449	DEBDYUTI DAS				
UG/02/BCA/2020/027	AU/2020/0004525	RISHI BARUA				
UG/02/BCABFSI/2020/001	AU/2020/0004505	Somnath Gayen				
UG/02/BCABFSI/2020/002	AU/2020/0004598	BARUN RAJBHAR				
UG/02/BCABFSI/2020/003	AU/2020/0004605	RAKIBUL ISLAM				
UG/02/BCAGA/2020/006	AU/2020/0004497	Abhishek Mondal				
UG/02/BCAGA/2020/002	AU/2020/0004500	Arka Mitra				
UG/02/BCAGA/2020/003	AU/2020/0004524	Sourav Mondal				
UG/02/BCAGA/2020/004	AU/2020/0004539	Ranita Bagchi				
UG/02/BCAGA/2020/005	AU/2020/0004568	SUBHAJIT SIRCAR				
UG/02/BCAGA/2020/001	AU/2020/0004493	Susmit Shaw				

Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Signature of Dean/HOD **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

CO ATTAINMENT – GAP ANALYSIS & REMEDIAL MEASURES

ADAMAS UNIVERSITY, KOLKATA SCHOOL OF DEPARTMENT OF CO ATTAINMENT - GAP ANALYSIS & REMEDIAL MEASURES							
Batch :	2020-22					Academic Year: 2020-21	
Course Code & Name			Name of the Coordinator			Year & Semester	
CSE11410			SUBHASISH MOHAPATRA			I & I	
CO	Direct Assessment	Indirect Assessment	CO Attainment	Target	CO Attainment Gaps	Action for Bridge the Gap	Target Modification
CO1							
CO2							
CO3							
CO4							
CO5							

Signature of HOD/Dean

Date:

Signature of Faculty

Date:

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

CO-PO ATTAINMENT

ADAMAS UNIVERSITY, KOLKATA
SCHOOL OF
DEPARTMENT OF
CO-PO ATTAINMENT

Programme	Year & Sem: I I& III	Academic Year: 2020-21	Batch:2020-22

Course Code	Course Name	CO-PO	PO1	PO 2	PO 3	PO4	PO5	PO6	PO 7	PO8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CSE11410	PYTHON	Relationship	Relationship	CO2, CO3, CO4, CO5	CO1, CO2, CO3	CO1,CO2, CO3, CO4, CO5	NA	CO4	NA	NA	NA	NA	NA	CO5	CO3,CO4	CO5	NA
		Mapping Value	Mapping Value	3	3	2	NA	2	NA	NA	NA	NA	NA	2	3	2	NA
		Attainment	Attainment	2.4	2.4	1.6	NA	1.6	NA	NA	NA	NA	NA	1.6	2.4	1.6	*

Signature of HOD/Dean

Signature of Faculty

Year: 2021
Semester: III

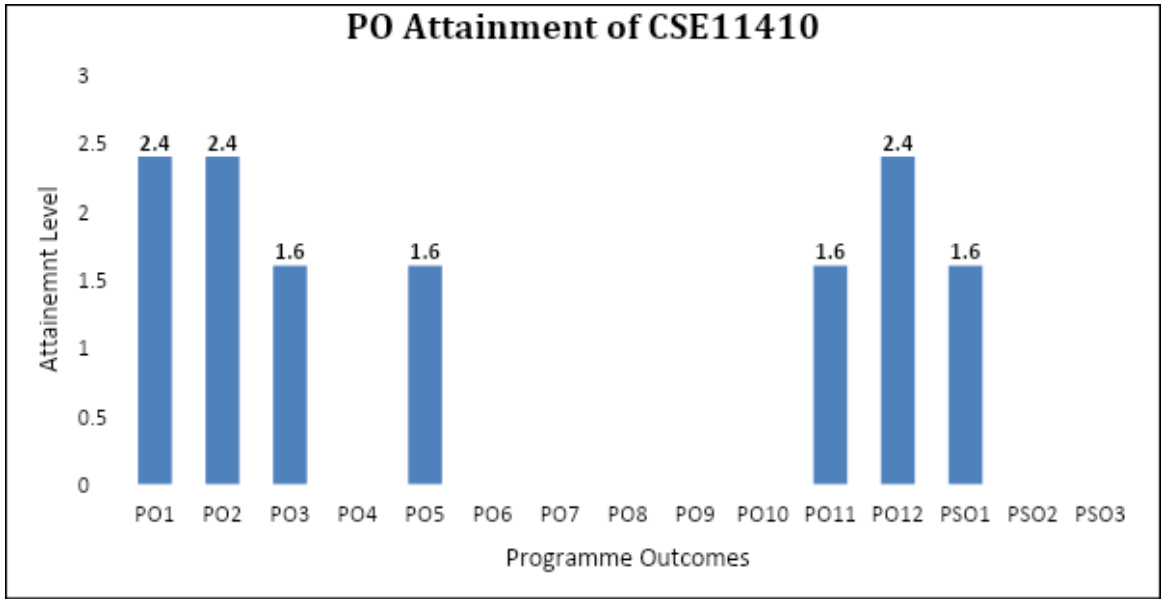
6. Name of the Faculty: Subhasish Mohapatra
7. Course : CSE
8. Program : BCA
9. Target : 60%

Course Code: CSE11410
L: 3
T: 0
P: 0
C: 3

Date:

Date:

PO ATTAINMENT OF THE COURSE



Year: 2021
Semester: III

6. **Name of the Faculty: Subhasish Mohapatra** **Course Code: CSE11410**
7. **Course : CSE** **L: 3**
8. **Program : BCA** **T: 0**
9. **Target : 60%** **P: 0**
C: 3

Signature of HOD/Dean **Signature of Faculty**

Date: **Date:**

Year: 2021
Semester: III

6. Name of the Faculty: Subhasish Mohapatra		**Course Code: CSE11410**
7. Course	**: CSE**	**L: 3**
8. Program	**: BCA**	**T: 0**
9. Target	**: 60%**	**P: 0**
		C: 3

INSTRUCTIONS FOR FACULTY

Instructions for Faculty

- Faculty should keep track of the students with low attendance and counsel them regularly.
- Course coordinator will arrange to communicate the short attendance (as per University policy) cases to the students and their parents monthly.
- Topics covered in each class should be recorded in the table of RECORD OF CLASS TEACHING (Suggested Format).
- Internal assessment marks should be communicated to the students twice in a semester.
- The file will be audited by respective Academic Monitoring and Review Committee (AMRC) members for theory as well as for lab as per AMRC schedule.
- The faculty is required to maintain these files for a period of at least three years.
- This register should be handed over to the head of department, whenever the faculty member goes on long leave or leaves the Colleges/University.
- For labs, continuous evaluation format (break-up given in the guidelines for result preparation in the same file) should be followed.
- Department should monitor the actual execution of the components of continuous lab evaluation regularly.
- Instructor should maintain record of experiments conducted by the students in the lab weekly.
- Instructor should promote students for self-study and to make concept diary, due weightage in the internal should be given under faculty assessment for the same.
- Course outcome assessment: To assess the fulfilment of course outcomes two different approaches have been decided. Degree of fulfillment of course outcomes will be assessed in different ways through direct assessment and indirect assessment. In Direct Assessment, it is measured through quizzes, tests, assignment, Mid-term and/or End-term examinations. It is suggested that each examination is designed in such a way that it can address one or two outcomes (depending upon the course completion). Indirect assessment is done through the student survey which needs to be designed by the faculty (sample format is given below) and it shall be conducted towards the end of course completion. The evaluation of the achievement of the Course Outcomes shall be done by analyzing the inputs received through Direct and Indirect Assessments and then corrective actions suggested for further improvement.
- **Submission Targets of Course Contents:**
  - **S. No. 1 to 8 : Before Starting the Course**
  - **S. No. 9 & 10 : After Mid Semester Examination**
  - **S. No. 11 to 18 : Immediately After End Semester Examination**
  - **S. No. 19 to 22 : After Declaration of Result of the Course**