INDEX

NAME: S. Balaji STD: B.Tech SEC: IT ROLL NO.: Python Programming

S.No.	Date	Title	Page No.	Remarks
I.		PROBLEM SOLVING		
1.		Definition, Example		
2.		Techniques		
A.		ALGORITHM		
1.		Definition, Purpose, Example		
2.		Algorithmic Strategy, Solution		
3.		Analysis of Algorithm		
4.		Efficiency (Complexity) of Algorithm		
4.1		Time Complexity		
4.2		Space Complexity		
4.3		Methods for determining efficiency		
5.		Asymptotic Notations		
5.1		Big O		
5.2		Big Ω		
5.3		Big Θ		
6.		Characteristics (or) Qualities of good (or) Best Algorithm		
7.		Space-Time Tradeoff		
8.		Algorithm Flowchart		
9.		Building Blocks of Algorithm		
10.		Searching Techniques		
11.		Sorting Techniques		
B.		FLOWCHART		
1		Definition, Symbols, Table, Example		
2		Rules for Drawing Flowchart		
C.		PSEUDO CODE		
1		Definition, Example		
2		Guidelines for Writing Pseudo Code		
3		Common Keywords In Pseudo Code		
4		Syntax of Pseudo Code		
5		Advantages, Disadvantages		
-		Algorithm vs Flowchart vs Pseudocode		

S.No.	Title	Page No.	Completion Status
D.	PROGRAM / PROGRAMMING LANGUAGE		
1	Definition, Example		
3	Types of Programming Languages		
3.1	Machine (or) Binary (or) Low-Level-Language		
3.2	Assembly (or) Intermediate Language		
3.3	High-level- language		
3.3.1	Compiler		
3.3.2	Interpreter		
3.3.3	Advantages, Disadvantages		
2	Computer System Level Hierarchy		
4	Categories (or) Classification of High Level Language		
4.1	Basis: Commercial		
4.1.1	Commercial Programming Language		
4.1.2	Scientific Programming Language		
4.1.3	Special Purpose Programming Language		
4.1.4	General Purpose Programming Language		
4.2	Basis: Design Paradigm		
4.2.1	Markup Programming Language		
4.2.2	Concurrent (or) Parallel Programming Language		
4.2.3	Scripting Programming Language		
4.2.4	Compiled Programming Language		
4.2.5	Interpreted Programming Language		
4.2.6	Procedural Programming Language		
4.2.7	Functional (or) Modular Programming Language		
4.2.8	Object-oriented Programming Language		
5	Algorithmic Problem Solving		
6	Simple Strategies for Developing Algorithms		
6.1	Iteration		
6.2	Recursion		
6.3	Decision-Making		
9	Building Blocks of Algorithm		
9.1	Statement		
9.2	State		
9.3	Control Flow		
9.3.1	Sequence		
9.3.2	Selection		
9.3.3	Iteration (or) Looping		
~~9.4~~	~~Functions~~		
~~9.5~~	~~Modules~~		
~~9.6~~	~~Classes and Objects~~		

2.2.3.3.3 Docstring Comments
2.2.3.4 Form Feed
2.2.4 Block (or) Suite
2.2.5 Tokens
2.2.5.1 Identifier
2.2.5.1.1 Naming Conventions
2.2.5.1.2 Variables, Functions, Modules, Classes, (Declaration Syntax)
2.2.5.2 Keywords (Types: 10)
2.2.5.2.1 Value Keywords
2.2.5.2.2 Operator Keywords
2.2.5.2.3 Branching Keywords
2.2.5.2.4 Iteration Keywords
2.2.5.2.5 Structure Keywords
2.2.5.2.6 Returning Keywords
2.2.5.2.7 Variable Handling Keywords
2.2.5.2.8 Import Keywords
2.2.5.2.9 Exceptional Handling Keywords
2.2.5.2.10 Asynchronous Keywords
2.2.5.3 Delimiters
2.2.5.4 Literals (or) Data (or) Constants (or) Datatypes
2.2.5.4.1 Numeric
2.2.5.4.1.1 Integer - • Decimal • octal • Hexadecimal • Binary
2.2.5.4.2.2 Float - • Floating Point • Scientific (or) Exponent
2.2.5.4.3.3 Complex
2.2.5.4.2 Boolean (or) Logical
2.2.5.4.3 Binary
2.2.5.4.3.1 Bytes
2.2.5.4.3.2 Bytearray
2.2.5.4.3.3 Memoryview
2.2.5.4.4 None
2.2.5.4.5 Ordered Collection (or) Sequence
2.2.5.4.5.1 String
2.2.5.4.5.2 List
2.2.5.4.5.3 Tuple
2.2.5.4.5.4 Range
2.2.5.4.6 Unordered Collection (or) Non-Sequence
2.2.5.4.6.1 Sets - • Set • Frozen Set
2.2.5.4.6.2 Mapping - • Dictionary
2.2.5.5 Operators
2.2.5.5.1 Arithmetic
2.2.5.5.2 Relational (or) Comparison
2.2.5.5.3 Logical
2.2.5.5.4 Bitwise
2.2.5.5.5 Assignment
2.2.5.5.6 Membership
2.2.5.5.7 Identity
2.2.5.5.8 Conditional (or) Ternary
2.2.5.5.9 Miscellaneous - • dot • grouping • subscript • slice • ellipsis • decorator
2.2.5.5.10 Operator Precedence
2.2.5.5.10.1 Operators Priority Table
2.2.5.5.10.2 Expression, Evaluation of Expression
3 State
4 Control Flow
4.1 Sequential
4.2 Alternative (or) Branching
4.2.1 Selection (or) Conditional Branching

4.2.1.1	Conditional (or) Simple If
4.2.1.2	Alternative If .. Else
4.2.1.3	Chained If -- Elif - Else
4.2.1.4	Nested If -- Else (or) Nested conditionals
4.2.1.5	Match case • match expression • case - pattern (5 types of pattern)
4.2.2	Jump (or) Unconditional Branching
4.2.2.1	Break
4.2.2.2	Continue
4.2.2.3	Pass
4.3	Iterative (or) Looping (or) Repetitive
4.3.1	For loop
4.3.2	While loop • Initialization • Condition • updation
4.3.3	Nested Loop
4.3.3.1	Nested For
4.3.3.2	Nested While
4.3.3.3	Nested For-While
4.3.3.4	Nested While-For
5	Exceptional Handling
5.1	Errors, Exceptions
5.2	Types of errors (Basis: Situation)
5.2.1	Compile-time
5.2.2	Logical
5.2.3	Run-time
5.3	Types of statement (Basis : Error)
5.3.1	Normal Statement
5.3.2	Critical Statement
5.4	Exceptional Handling
5.4.1	Try Block
5.4.2	Except Block
5.4.3	Finally Block
5.4.4	User defined exception
5.4.4.1	Raise keyword
5.4.4.2	Assert keyword

Veera's Student's Choice

Name : S. Balaji

II Year IT

STD ~~IV~~ Sem SEC : B ROLL NO. : VM13475

S.No	Date	TITLE	Page No.	Teacher's Sign
B.		MODULAR PROGRAMMING APPROACH		
1		Functions		
1.1		Definition		
1.2		Need, Advantages		
1.3		Elements of Function		
1.3.1		Function Definition		
1.3.1.1		Function Header		
1.3.1.1.1		Def Keyword		
1.3.1.1.2		Function Name		
1.3.1.1.3		Parameter List		
1.3.1.1.4		Colon		
1.3.1.2		Function Body		
1.3.1.2.1		Indentation, Block		
1.3.1.2.2		Nested Block		
1.3.1.2.3		Return Statement (or) Value		
1.3.1.2.4		Docstring Statement		
1.3.2		Function Call		
1.3.1.1		Function Name		
1.3.1.2		Argument List		
1.3.3		Basis : Return Value		
1.3.3.1		Fruitful Function		
1.3.2		Void Function		
1.3.4		Basis: Purpose (or) Usage (or) Concept		
1.4.1		Built-in (or) Pre-Defined Function		
1.4.2		User-Defined Function		

1.3.4.2.1 Normal Functions
1.3.4.2.2 Nested Functions
1.3.4.2.3 Recursive Functions - Base Condition - Recursive condition
1.3.4.2.4 Decorators (or) Wrapper Functions
1.3.4.2.5 Generators
1.3.4.3 Lambda Functions (or) Anonymous Functions
1.3.4.3.1 Filter
1.3.4.3.2 Map
1.3.4.3.3 Reduce
1.3.4.2.1.1 Parameters
1.3.4.2.1.2 Arguments
1.3.4.2.1.3 Binding of parameters to arguments (or) Function Prototypes
1.3.4.2.1.3.1 No arguments - No return value
1.3.4.2.1.3.2 With arguments - No return value
1.3.4.2.1.3.3 No arguments - With return value
1.3.4.2.1.3.4 With arguments - With return value
1.3.4.2.1.4 Types of Arguments (or) Parameters (Basis: Specification)
1.3.4.2.1.4.1 Formal Parameters (Arguments)
1.3.4.2.1.4.2 Actual Parameters (Parameters)
1.3.4.2.1.5 Types of Arguments (Basis: Flexibility)
1.3.4.2.1.5.1 Fixed Function Arguments
1.3.4.2.1.5.1.1 Positional (or) Required Arguments
1.3.4.2.1.5.2 Variable Function Arguments
1.3.4.2.1.5.2.1 Keyword Arguments
1.3.4.2.1.5.2.2.2 Default Arguments
1.3.4.2.1.5.2.2.3 Variable Length (or) Arbitrary Arguments
1.3.4.2.1.5.2.2.3.1 Non-keyword Variable Length Arguments
1.3.4.2.1.5.2.2.3.2 Keyword Variable Length Arguments
1.3.4.2.1.6 Argument Passing (or) Calling Methods
1.3.4.2.1.6.1 Call (or) Pass By Value
1.3.4.2.1.6.2 Call (or) Pass By Reference
1.3.4.2.1.7 Scope of Variables
1.3.4.2.1.7.1 Global
1.3.4.2.1.7.2 Local
1.3.4.2.1.7.3 Non-local (or) Enclosed (or) Enclosing
1.3.4.2.1.7.4 Built-In (or) Library
1.3.4.2.1.7.5 Hierarchy of Scopes (LEGB Rule)
1.3.4.2.1.8 Function Composition
1.3.4.2.3.1 Basis: Calling Function
1.3.4.2.3.1.1 Direct Recursive Function
1.3.4.2.3.1.2 Indirect Recursive Function
1.3.4.2.3.2 Basis:
1.3.4.2.3.2.1 Tail Recursion
1.3.4.2.3.2.2 Non-Tail Recursion
1.3.5 Basis: Dependency of Return Value On Argument
1.3.5.1 Pure Function
1.3.5.2 Impure Function

2 Modules
2.1 Definition
2.2 Advantages
2.3 Accessing members of modules
2.4 Types of Modules
2.4.1 Built-In Modules
2.4.1.1 math module
2.4.1.2 random module
2.4.1.3 os module
2.4.1.4 sys module
2.4.1.5 calendar module
2.4.1.6 time module
2.4.1.7 datetime module
2.4.1.8 string module
2.4.1.9 numbers module
2.4.1.10
2.4.2 User-Defined Modules
2.4.2.1 Import Statements (or) Ways to Import
2.4.2.1.1 import
2.4.2.1.2 import .. as ..
2.4.2.1.3.1 from .. import ..
2.4.2.1.3.2 from .. import *
2.4.2.1.4 from .. import .. as
2.4.2.2 dir() function
3 Packages
3.1 Definition
3.2 Advantages
3.3.1 Creation of Packages
3.3.2 Running of Packages
3.4 Package Manager
3.4.1 PIP
3.4.1.1 Definition
3.4.1.2 PIP Commands in CMD (Windows)
3.4.1.2.1 Help for Commands
3.4.1.2.2 Check Version
3.4.1.2.3 Check compatible Dependencies of Installed Packages
3.4.1.2.4 List Installed Packages with their Versions
3.4.1.2.5 Show Information of Installed Packages
3.4.1.2.6 Output Installed Packages in Requirements Format
3.4.1.2.7 Download Packages
3.4.1.2.8 Install Packages
3.4.1.2.9 Upgrade Packages
3.4.1.2.10 Downgrade Packages
3.4.1.2.11 Uninstall Packages
3.4.1.2.12 Get More Information of A Specific Package (Metadata)
3.4.1.2.13 Generate Requirements.txt file
3.4.1.2.14 Get All Dependencies Installed Into System
3.4.2 Conda

Libraries

5 Important (or) Popular Functions, Modules, Packages And Libraries

5.1 Data Science.

5.2 Machine learning

5.3 Deep learning

5.4 Transfer Learning

5.5 NLP

5.6 Computer Vision

5.7 Image Processing

5.8 Data Visualization

5.9 Data Analytics (Analysis) (or) Statistical Analysis

5.10 Automation

5.11 Automation Testing

5.12 Chatbot Development

5.13 GUI

5.14 Web Development (Back-end Development)

5.15 Mobile App Development

5.16 Database connectivity

5.17 Ethical Hacking

5.18 others

5.18.1 QR Code Generator

5.18.2 Music

5.18.3 Chess

INDEX

NAME: S. Balaji B.Tech YEAR: I-Year SEC: IT-B ROLL NO.: 13475 SUB.: VM Object Oriented Programming

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
C.		OBJECT ORIENTED PROGRAMMING APPROACH		
1		Definition, Aim, Features		
2		Class		
2-1		Definition		
2-2		Class Members		
2-2-1		~~(Static Variable)~~ Class Variable (Attribute)		
2-2-2		Class Method (~~Behaviour~~)		
2-3		Creation of Class		
3		Object		
3-1		Definition		
3-2		Class Instantiation (Creation of Object)		
3-3		Accessing Class Members ~~Using Object~~		
2-4		Namespace		
2-4-1		Class (or) ~~Static~~ Namespace		
2-4-2		Object (or) Instance Namespace		
3-4		~~Object~~ Members		
3-4-1		Instance Variable		1-2
3-4-2		Instance Method		
2-5		Static Method		
2-6		Nested Class		
2-7		Constructor		
2-8		Destructor		
3		Encapsulation (or) Access Specifier		
3-1		Public		
3-2		Private		
3-3		Protected		
4		Abstraction		

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
4		Abstraction		
4.1		Abstract Class		
4.2		Abstract Method		
5		Inheritance		
5.1		Parent Class		
5.2		Child Class		
5.3		Types of Inheritance		
5.3.1		Single Level		
5.3.2		Multi Level		
5.3.3		Multiple		
5.3.4		Hierarchical		
5.3.5		Hybrid		
5.4		super() method		
5.5		MRO		
5.6		isinstance() method		
6		Polymorphism		
6.1		Duck Typing		
6.2		Operator Overloading		
6.3		Method Overloading • Default arg. • Variable length arg		
6.4		Method Overriding • Built-in • User-defined		
6.4.1		Magical Methods		
7		Iterator		
2.2.3		Accessing Class Members		
2.2.3.1		Using Class Name in both inside and outside class		
2.2.3.2		Using self parameter inside class (creating new instance variable)		
2.2.3.3		Using object name outside class		
3.4.3		Accessing Object (Instance) Members		
3.4.3.1		Using self parameter inside class		
3.4.3.2		Using object name outside class		

INDEX

In Python [redacted] (1)

NAME: S. Balaji STD: B-Tech SEC: IT ROLL NO.: Data Structures

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks
E.		DATA STRUCTURES		
1.		Data		
2.		Structure		
2.1		Node		
2.1.1		Orphaned Nodes		
2.1.2		Null Nodes (None Nodes)		
2.2		Node Structure		
2.2.1		Data		
2.2.2		Link (or) Reference		
3		Real-life Applications		
3.1		Online Ticket Booking System		
3.2		Web Browser		
3.3		Bank, Theatre, Restaurant, College		
3.4		Music Player		
3.5		Google Map		
3.6		Database Indexing, File Explorer, SEO		
3.7		Computer graphics Rendering, Routing Table		
3.8		Web Search, Process Scheduling		
3.9		Data Analysis, Data Mining		
3.10		Text-full-text-search (MS Word)		
3.11		Auto-completion, Dictionary Application		
3.12		Database, Google Search, Password Hash		
3.13		Heap Sort, K[th] smallest/largest of no.		
3.14		Cell Phone Contacts, Speech Processing		
3.15		Search Engine, Spell Checkers, Digital Forensics IDE		
4		Primitive Data Structure		
4.1		Integer		
4.2		Float		
4.3		Complex		
4.4		Boolean		
4.5		String		

5 Non-Primitive Data Structure

5.1 Built-In Data Structure

5.1.1 List

5.1.1.1 Definition, Example

5.1.1.2 List Structure

5.1.1.3 Mutability

5.1.1.4 List Operations

5.1.1.4.1 Creation

5.1.1.4.2 Indexing

5.1.1.4.3 Concatenation

5.1.1.4.4 Appending

5.1.1.4.5 Repetition

5.1.1.4.6 Updation (or) Modification

5.1.1.4.7 Membership

5.1.1.4.8 Comparison

5.1.1.4.9 Slicing, Stride When Slicing

5.1.1.4.10 Copying (or) Aliasing

5.1.1.4.11 Cloning (or) Duplication

5.1.1.4.12 Deletion

5.1.1.4.13 Advanced List Processing

5.1.1.4.13.1 List Comprehension

5.1.1.4.14 Accessing List Elements

5.1.1 Definition, Example

5.1.2 Tuple (same as list except Slicing, Modification)

5.1.2.2 Tuple Structure

5.1.2.3 Immutability

5.1.2.4 Tuple Operations (except modification, slicing)

5.1.2.4.12 Advanced Tuple Processing

5.1.2.4.12.1 Tuple Comprehension

5.1.2.4.13 Accessing Tuple Elements

5.1.2.4.14 Tuple as argument

5.1.2.4.15 Tuple as return value

5.1.3 Set

5.1.3.1 Definition, Example

5.1.3.2 Set Structure (Hashing) (or) (Non-Sequencing)

5.1.3.3 Mutability

5.1.3.4

INDEX

NAME: Balaji S STD.: ____ SEC.: IT ROLL NO.: ____ SUB.: Data Structures in Python-(2)

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
6.1		User Defined Data Structure		
6.1.1		Array (1D, 2D to List)		
6.1.2		Linked List		
6.1.2.1		Singly Linked List (SLL)		
6.1.2.1.1		SLL Representation		
6.1.2.1.2		Node Class Creation		
6.1.2.1.3		SLL Class Creation		
6.1.2.1.4		Node, SLL Objects Creation		
6.1.2.1.5		Allocating memory for nodes		
6.1.2.1.6		Linking each node		
6.1.2.1.7		SLL Operations		
6.1.2.1.7.1		Traversing, Displaying (or) Printing All Node Data		
6.1.2.1.7.2		Inserting A Node At The Beginning Of A SLL		
6.1.2.1.7.3		Inserting A Node At The End Of A SLL		
6.1.2.1.7.4		Inserting A Node At The Middle Of A SLL		
6.1.2.1.7.5		Searching A Node In A SLL		
6.1.2.1.7.6		Deleting A Node At The Beginning Of A SLL		
6.1.2.1.7.7		Deleting A Node At The End Of A SLL		
6.1.2.1.7.8		Deleting A Node At The Middle Of A SLL		
6.1.2.1.8		Time Complexity		
6.1.2.1.9		Space Complexity		
6.1.2.2		Doubly Linked List (DLL)		
6.1.2.3		Circular Linked List (CLL)		
6.1.2.3.1		Circular Singly Linked List (CSLL)		
6.1.2.3.2		Circular Doubly Linked List (CDLL)		
6.1.2.3.3		Real Life Applications		
6.1.2.1.7.9		Mirroring (or) Reversing A SLL		

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
6.1.3		Stacks		