# Comprehensive Unit-Wise Question Bank

Generated strictly according to the provided syllabus blueprint.

## Unit I: Computer Systems, Software, and Networks

### *1 Mark Questions (1 Mark Each)*

**Q1.** Define the term 'Computer Generation'.

**Q2.** What is the main electronic component used in first-generation computers?

**Q3.** Name the generation of computers that introduced microprocessors.

**Q4.** Draw a very basic symbolic box representing the Central Processing Unit (CPU).

**Q5.** What is the full form of ALU?

**Q6.** What is the primary function of the Control Unit (CU)?

**Q7.** Define volatile memory with an example.

**Q8.** State the primary difference between RAM and ROM.

**Q9.** What is a CPU register?

**Q10.** Why is cache memory used in computer systems?

**Q11.** Name any two secondary storage devices.

**Q12.** What do you mean by sequential access method?

**Q13.** Give an example of a direct-access storage device.

**Q14.** Classify an optical mouse as an input or output device.

**Q15.** What is the purpose of a Visual Display Unit (VDU)?

**Q16.** Define System Software.

**Q17.** Give two examples of application software.

**Q18.** What is utility software?

**Q19.** Define firmware.

**Q20.** Name any two network devices used to connect computers.

### *2 Marks Questions (2 Marks Each)*

**Q1.** Differentiate between primary memory and secondary memory.

**Q2.** Explain the concept of memory hierarchy in terms of speed and capacity.

**Q3.** Briefly explain the role of Cache Memory in accelerating CPU performance.

**Q4.** Distinguish between System Software and Application Software with examples.

**Q5.** What is the importance of a computer network in modern computing environments?

**Q6.** Differentiate between LAN and WAN based on geographical area and data transmission rate.

**Q7.** Explain the function of a Repeater in a network.

**Q8.** How does a Network Bridge differ from a Hub?

**Q9.** What is a Network Switch and why is it called an intelligent hub?

**Q10.** Explain the primary role of a Router in traffic routing across networks.

**Q11.** Define a Gateway device and explain its significance in heterogeneous networks.

**Q12.** Briefly describe the functions of the Input Unit and Output Unit.

**Q13.** What is the purpose of registers inside a CPU? Name any two types of registers.

**Q14.** Explain the characteristics of magnetic tapes as secondary storage devices.

**Q15.** What is Metropolitan Area Network (MAN)? State its typical coverage scope.

## *5 Marks Questions (5 Marks Each)*

**Q1.** Draw and explain the detailed block diagram of a digital computer system, showing all its functional units.

**Q2.** Elaborate on the evolution of computers across all five generations, highlighting the key technologies used.

**Q3.** Provide an in-depth classification of computer memory. Diagrammatically illustrate the memory hierarchy.

**Q4.** Discuss the different types of software (System, Application, and Utility) with appropriate real-world examples.

**Q5.** What is Firmware? Discuss its role, characteristics, and how it differs from traditional software and hardware.

**Q6.** Explain the concept of a Computer Network. Describe the importance, benefits, and challenges of networking.

**Q7.** Compare and contrast LAN, MAN, WAN, and the Internet based on size, ownership, speed, and technology.

**Q8.** Explain the working principles, features, and differences between a Hub, a Switch, and a Router.

**Q9.** Describe the following network internetworking devices in detail: (a) Repeater, (b) Bridge, (c) Gateway.

**Q10.** Explain the storage hierarchy. Compare primary and secondary storage technologies based on cost, speed, capacity, and volatility.

## *8 Marks Questions (8 Marks Each)*

**Q1.** Describe in comprehensive detail the hardware architecture of a modern computer system. Draw a neat block diagram and fully explain the core roles of the CPU, control unit, ALU, registers, primary storage, and secondary storage systems.

**Q2.** Provide an exhaustive comparative analysis of all five generations of computers. Your answer must critically analyze the switching circuits, primary/secondary storage media, operating systems, languages used, speed, and examples for each generation.

**Q3.** Discuss computer software comprehensively. Classify software into its major domains, explain system software components (OS, device drivers), analyze specific categories of application software, and detail utility applications and firmware functionality.

**Q4.** Write a detailed essay on Computer Networks. Define its core objectives, explain the architectural layout of LAN, MAN, and WAN networks, and comprehensively discuss how data flows across the Internet using routers and gateways.

**Q5.** Compare and contrast the following components comprehensively: (a) Cache Memory vs Registers, (b) RAM vs ROM, (c) Hub vs Switch, (d) Sequential Access vs Direct Access storage methods. Include analytical parameters for each comparison.

# Unit II: Program Development, Algorithms, and Languages

### *1 Mark Questions (1 Mark Each)*

**Q1.** What is the first step in the program development lifecycle?

**Q2.** Define the term 'Algorithm'.

**Q3.** What is pseudo code?

**Q4.** Name the flowchart symbol used to represent a decision/condition.

**Q5.** Which flowchart symbol is used for input/output operations?

**Q6.** What does a rectangle symbol represent in a standard flowchart?

**Q7.** Define structured programming paradigm.

**Q8.** What is the core focus of Object-Oriented Programming (OOP)?

**Q9.** Define a low-level programming language.

**Q10.** What is a high-level programming language?

**Q11.** What is the role of an Assembler?

**Q12.** Define a Compiler.

**Q13.** What is an Interpreter?

**Q14.** State one difference between a compiler and an interpreter.

**Q15.** What is a linker?

**Q16.** What is the primary function of a loader?

**Q17.** Define an Integrated Development Environment (IDE).

**Q18.** What is a syntax error?

**Q19.** Give an example of a logical error.

**Q20.** What do you mean by debugging?

### *2 Marks Questions (2 Marks Each)*

**Q1.** Explain the 'Problem Analysis' phase in program development.

**Q2.** List any four key characteristics of a good computer program.

**Q3.** Differentiate between Top-down and Bottom-up design approaches.

**Q4.** What are Control Structures? Name the three basic types used in algorithms.

**Q5.** Explain the significance of using standard symbols when creating a flowchart.

**Q6.** Differentiate between low-level and high-level languages with examples.

**Q7.** Explain how a Linker and a Loader work together to prepare an executable program.

**Q8.** What is an IDE? Name any two popular IDEs used for programming.

**Q9.** Differentiate between syntax errors and logical errors.

**Q10.** What is a semantic error? Provide a brief example.

**Q11.** Distinguish between compile-time errors and run-time errors.

**Q12.** What is a link-time error and when does it occur?

**Q13.** Briefly define environmental errors and input/output errors.

**Q14.** What is the main difference between software testing and software debugging?

**Q15.** Why is pseudo code often preferred over flowcharts during early design phases?

## *5 Marks Questions (5 Marks Each)*

**Q1.** Detail all successive phases of the Program Development Life Cycle (PDLC) with a clear descriptive overview of each.

**Q2.** Define an Algorithm. Discuss the essential characteristics and guidelines for writing an efficient algorithm.

**Q3.** What is a Flowchart? Draw and describe the major standard ANSI/ISO flowchart symbols and their specific use cases.

**Q4.** Compare and contrast the Structured Programming paradigm with the Object-Oriented Programming (OOP) paradigm.

**Q5.** Explain Top-down and Bottom-up software design methodologies. Discuss their respective advantages and limitations.

**Q6.** Discuss the evolution of programming languages across different generations (from 1GL to 5GL).

**Q7.** Provide a comprehensive comparison between Assemblers, Compilers, and Interpreters. Explain their translation mechanisms.

**Q8.** Explain the technical roles of Editors, Linkers, Loaders, and Debuggers inside a modern Integrated Development Environment (IDE).

**Q9.** Classify and describe the various types of programming errors: Syntax, Semantic, Logical, Compile-time, and Run-time errors.

**Q10.** Explain the systematic processes of Software Testing and Debugging. Discuss how they ensure program robustness.

## *8 Marks Questions (8 Marks Each)*

**Q1.** Explain the program development ecosystem thoroughly. Provide a deep analysis of problem solving through algorithms, pseudo code, and flowcharts. Design a detailed algorithm and a matching flowchart to solve a complex decision-making problem.

**Q2.** Write an exhaustive essay comparing Programming Paradigms. Contrast Structured Programming against Object-Oriented Programming. Analyze Top-down vs. Bottom-up design techniques, and formulate the essential metrics that define a high-quality program.

**Q3.** Analyze the entire translation and execution pipeline of a program. Trace a source code file from text creation in an Editor, through translation (Compiler/Interpreter), linking via Linker, loading via Loader, into memory execution.

**Q4.** Provide a comprehensive treatise on programming languages and their translators. Detail the structural differences between Low-level (Machine/Assembly) and High-level languages across all historical generations, explaining why modern software relies on compilers and interpreters.

**Q5.** Develop an exhaustive taxonomy of programming errors. Discuss the direct causes, detection mechanisms, and remediation strategies for Syntax, Semantic, Logical, Compile-time, Run-time, Link-time, Environmental, and I/O errors. Contrast testing vs. debugging.

# Unit III: Python Basics, Strings, and Control Instructions

### *1 Mark Questions (1 Mark Each)*

**Q1.** What is a Python identifier?

**Q2.** Can a Python keyword be used as a variable name? (Yes/No)

**Q3.** Name any two built-in basic data types in Python.

**Q4.** Is a Python string mutable or immutable?

**Q5.** Are lists mutable or immutable in Python?

**Q6.** What is dynamic typing in Python?

**Q7.** Which operator is used for exponentiation (power) in Python?

**Q8.** What is the result of 11 // 2 in Python?

**Q9.** State the precedence order between multiplication (*) and addition (+).

**Q10.** What is the purpose of the type() function?

**Q11.** How do you import a module in Python?

**Q12.** What character is used to start a single-line comment in Python?

**Q13.** How can a single statement span multiple lines in a Python script?

**Q14.** What is the default string format conversion function in Python?

**Q15.** Which function is used to read data from the standard console input?

**Q16.** What is the function of the 'pass' statement in Python?

**Q17.** What does the all() built-in function return if all elements are true?

**Q18.** When does the any() function return True?

**Q19.** Which statement terminates the current loop immediately?

**Q20.** What is the purpose of the 'continue' statement?

### *2 Marks Questions (2 Marks Each)*

**Q1.** What are keywords in Python? List any four common examples.

**Q2.** Explain mutable and immutable data types with examples of each.

**Q3.** How does variable assignment and type inference work in Python?

**Q4.** Explain operator precedence and associativity with a small example expression.

**Q5.** What is explicit type conversion (casting) in Python? Give a code snippet.

**Q6.** Explain the role and value of comments and proper indentation in Python scripts.

**Q7.** How are string elements accessed using positive and negative indexing?

**Q8.** Name four built-in string methods and briefly state what they do.

**Q9.** Explain formatted printing using the string .format() method or f-strings.

**Q10.** How do conditional expressions (ternary operators) work in Python? Give an example.

**Q11.** Explain the behavior of all() and any() functions when applied to an iterable.

**Q12.** What is the significance of the 'else' block when combined with a 'while' or 'for' loop?

**Q13.** Differentiate between the 'break' and 'continue' control statements within a loop.

**Q14.** How does Python handle integer ranges and floating-point precision internally?

**Q15.** What is a module in Python and how do you access its built-in functions?

### *5 Marks Questions (5 Marks Each)*

**Q1.** Discuss basic Python building blocks: Identifiers, Keywords, Literals, Variables, and Statement Indentation rules.

**Q2.** Explain Python's core data types category-wise. Differentiate strictly between mutable and immutable data objects.

**Q3.** Elaborate on Arithmetic Operators in Python. Discuss floor division, modulus, exponentiation, operator precedence, and associativity.

**Q4.** What are Python Modules? Explain how to create, import, and utilize built-in modules with specific code examples.

**Q5.** Describe the properties of Python Strings. Explain indexing, slicing, concatenation, and element accessibility with examples.

**Q6.** Discuss at least five major built-in string methods (e.g., split, join, replace, upper, lower) with code snippets.

**Q7.** Explain Console Input/Output operations. Detail formatted printing using f-strings, format() qualifiers, and escape characters.

**Q8.** Describe Decision Control structures in Python. Explain if, if-else, if-elif-else frameworks and conditional expressions.

**Q9.** Explain the working mechanisms of Python loops (for and while). Show how loops can be controlled dynamically.

**Q10.** Detail the advanced loop structures in Python: explain the exact usage of break, continue, pass, and the loop-else block.

### *8 Marks Questions (8 Marks Each)*

**Q1.** Write a comprehensive guide to Python Basics. Discuss identifiers, keywords, basic types, dynamic typing, variable assignment, and internal reference mechanisms. Explain mutability vs. immutability using memory maps and ID analysis.

**Q2.** Provide an exhaustive analysis of Operators, Precedence, and Expressions in Python. Cover arithmetic, logical, relational, bitwise, and conditional operators. Explain complex expression evaluation based on precedence and associativity tables.

**Q3.** Explain Python String manipulation to its deepest level. Discuss storage properties, immutability constraints, positive/negative slicing bounds, comparison operations, and provide functional scripts showcasing ten distinct built-in methods.

**Q4.** Develop a comprehensive overview of Decision Control and Repetition Control Instructions in Python. Explain multi-way conditional benchmarking, logical evaluations using all() and any(), and construct deep,

nested looping architectures.

**Q5.** Analyze the control transfer framework in Python loops. Write comprehensive code blocks demonstrating the technical execution paths of break, continue, pass, and loop-else constructs. Graphically describe how the else block reacts to abnormal loop breaks.

# Unit IV: Data Structures (Containers), Comprehensions, and Functions

### *1 Mark Questions (1 Mark Each)*

**Q1.** How do you create an empty list in Python?

**Q2.** What is the main structural difference between a list and a tuple?

**Q3.** Are duplicate elements allowed in a Python set?

**Q4.** How do you access a value in a dictionary?

**Q5.** Which built-in function returns the total number of items in a container?

**Q6.** What method is used to add an item to the end of a list?

**Q7.** How do you remove all items from a dictionary?

**Q8.** What is a set mathematical union operator symbol in Python?

**Q9.** Write the basic syntax outline of a list comprehension.

**Q10.** Can a dictionary comprehension be used to swap keys and values? (Yes/No)

**Q11.** What keyword is used to define a user-defined function in Python?

**Q12.** What is a built-in function? Give an example.

**Q13.** What value does a Python function return by default if no return statement is given?

**Q14.** What is an argument in a Python function call?

**Q15.** What operator is used for unpacking arguments from a tuple or list?

**Q16.** What does **kwargs unpack in a function definition?

**Q17.** Define a recursive function.

**Q18.** What is the base case in recursion?

**Q19.** State one primary disadvantage of a recursive function over iteration.

**Q20.** Which data structure is implicitly used by the system during recursive execution?

### *2 Marks Questions (2 Marks Each)*

**Q1.** Differentiate between a list and a dictionary in terms of element access.

**Q2.** What is a tuple and when should it be used instead of a list?

**Q3.** Explain how sets perform mathematical operations like intersection and difference.

**Q4.** How do you add, update, and delete key-value pairs in a Python dictionary?

**Q5.** What are container types? Name four built-in containers available in Python.

**Q6.** Explain the basic concept of a List Comprehension with a simple example.

**Q7.** How do Dictionary Comprehensions work? Provide a brief code line example.

**Q8.** Differentiate between built-in functions and user-defined functions.

**Q9.** What is positional argument passing and how does it differ from keyword arguments?

**Q10.** Explain argument unpacking using the asterisk (*) operator in function calls.

**Q11.** What is recursion? State the two essential components of a valid recursive function.

**Q12.** Briefly compare iteration vs recursion based on performance and memory usage.

**Q13.** How can you loop through both keys and values simultaneously in a Python dictionary?

**Q14.** What is a Set Comprehension? Give a quick syntactical example.

**Q15.** Explain the concept of an infinite recursive loop and what error it raises in Python.

## *5 Marks Questions (5 Marks Each)*

**Q1.** Compare Lists, Tuples, Sets, and Dictionaries based on syntax, mutability, ordering, duplicates, and primary use cases.

**Q2.** Explain the major built-in functions and operations available for Lists and Tuples (slicing, appending, extending, sorting, packing).

**Q3.** Discuss Python Dictionaries in detail. Explain key-value access, internal ordering properties, and key dictionary methods.

**Q4.** Explain Python Sets. Detail how mathematical operations like Union, Intersection, Difference, and Symmetric Difference are performed.

**Q5.** Elaborate on Comprehensions in Python. Provide clear examples for List Comprehensions, Set Comprehensions, and Dictionary Comprehensions.

**Q6.** Define Functions in Python. Discuss the benefits of modular programming, function definition syntax, and parameters vs arguments.

**Q7.** Explain the different types of arguments in Python functions: Positional, Keyword, Default, and Variable-length (*args, **kwargs).

**Q8.** What is argument unpacking? Explain how list/tuple unpacking (*) and dictionary unpacking (**) are utilized during function invocation.

**Q9.** Explain Recursion. Provide a complete, annotated python function to find the factorial of a number or Fibonacci series using recursion.

**Q10.** Provide a detailed comparison between Iteration and Recursion. Analyze their respective execution stack behaviors and stack overflow risks.

## *8 Marks Questions (8 Marks Each)*

**Q1.** Write an exhaustive technical treatise on Python's advanced container types. Construct complete analytical data rows tracking Lists, Tuples, Sets, and Dictionaries across mutability, indexing, performance cost, and native methods. Provide functional examples.

**Q2.** Mastering Comprehensions: Write a detailed analytical essay exploring List, Set, and Dictionary comprehensions. Include complex conditional mapping filters, nested comprehension layers, and compare performance against classic for-loops.

**Q3.** Develop a comprehensive guide to Python Functions. Detail structural scope boundaries, global vs local state interactions, argument passing models (pass-by-object-reference), and elaborate on custom setups featuring default configurations, *args, and **kwargs.

**Q4.** Write a deep-dive essay on Argument Unpacking mechanics in Python. Detail how sequential containers and mapping containers are parsed using * and ** operators within both definitions and invocations. Provide comprehensive multi-turn edge cases.

**Q5.** Examine Recursion vs Iteration profoundly. Define the memory mechanics of activation records on the call stack, explain why base cases prevent stack overflow execution crashes, and provide fully mapped, side-by-side recursive and iterative program scripts.