Dr. SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE (Autonomous)
Accredited by NAAC (Cycle-IV) with 'A+' Grade,
(Recognized by UGC & Approved by AICTE, New Delhi and Affiliated to Bharathiar University, Coimbatore)
486, Thudiyalur-Saravanampatti Road, Chinnavedampatti (Post), Coimbatore - 641 049.

Subject: GENERIC / CLUSTER CORE -3: OOPS WITH PYTHON PROGRAMMING **Code:** 25UCU409

QUESTION AND ANSWER
UNIT: 1

1. Describe one real-world situation where its readable syntax helps teams write and maintain code faster. (Google, 2024)

2. Describe the major version differences between Python 2 and Python 3 and explain how developers can check which version is currently running in their script or environment. (Microsoft, 2023)

3. Write a small Python code snippet that swaps the values of two variables without using any extra temporary variable. (Amazon, 2025)

4. Compare mutable data types (like list and dictionary) with immutable types (like tuple and string) and discuss how this difference affects memory usage and program behavior. (Meta, 2024)

5. Evaluate whether using floating-point numbers instead of integers in arithmetic operations can lead to precision errors and give one practical example from financial calculations. (Infosys, 2023)

6. Write a logical expression using and/or/not operators that checks whether a person's age is between 18 and 60 inclusive and explain what the expression returns in different cases. (Apple, 2025)

7. List at least four popular IDEs or code editors used for Python development. (Oracle, 2024)

8. Explain how the if-elif-else ladder works in Python and provide one realistic scenario where using nested if statements becomes necessary instead of a flat structure. (TCS, 2023)

9. Write a for loop that iterates through a list of integers, calculates their total sum, and uses the break statement to stop early if the sum exceeds 100. (Wipro, 2025)

10. Analyze the common causes that lead to infinite while loops in student programs and suggest at least two practical techniques to guarantee loop termination. (Netflix, 2024)

11. Compare the advantages and disadvantages of using input() function versus sys.argv for accepting data when writing command-line utility scripts in Python. (Google, 2023)

12. Create a small function that accepts one integer from the user and prints whether the number is even or odd, including basic handling for non-numeric input. (Microsoft, 2025)

13. Recall the standard operator precedence rules in Python and demonstrate with one expression how adding parentheses can completely change the final result. (Amazon, 2024)

14. Explain how to use the print() function with custom separators and end characters to display multiple variables in a clean, tabular-like format on one line. (Meta, 2023)

15. Discuss how Python’s support for arbitrary precision integers and floating-point limitations can affect financial applications, and suggest best practices when precision is critical. (Amazon, 2024)

16. Outline the complete process of accepting multi-line input from the user until they type a specific sentinel value (e.g., “quit”) and storing it in a list. (TCS, 2025)

17. Analyze why Python does not have traditional switch-case statements and explain three common idiomatic ways developers replace switch-case logic. (Google, 2024)

18. Evaluate the readability and maintainability trade-offs when using ternary conditional expressions versus regular if-else blocks in production code. (Microsoft, 2023)

19. Create a number guessing game (1–100) that uses a while loop, random module, and appropriate feedback messages (“too high”, “too low”, “correct”). (Infosys, 2025)

20. Explain how f-strings (formatted string literals) introduced in Python 3.6 improve code clarity compared to older methods (.format() and % operator). (Meta, 2024)

21. Write nested loop code to print a right-angled triangle pattern of asterisks (*) with height provided by the user. (Wipro, 2025)

22. Compare the memory and performance implications of using break/continue/pass statements versus restructuring loops to avoid them. (Apple, 2023)

23. Design a simple menu-driven program using while True and if-elif that performs addition, subtraction, multiplication, or division until the user chooses exit. (Oracle, 2024)

24. Recall and demonstrate the difference between is and == operators with examples involving integers, strings, lists, and None. (Netflix, 2025)

25. Design a command-line Python application for a retail store that takes user input for product prices and quantities via command-line arguments, applies discounts using conditional statements and loops, and outputs the total bill with GST. (Amazon, 2025)

26. Explain with code how recursion could be misused here compared to loops, and justify the better approach for efficiency. A bank requires a simple interest calculator function that accepts principal, rate, and time as parameters. (TCS, 2025)

27. Analyze why Python's dynamic typing might cause issues in a large-scale financial reporting script using variables and operators; propose safeguards using type hints or checks. (Infosys, 2025)

28. Create a Python script using loops and conditionals to validate user login credentials (username/password) from input, allowing 3 attempts before lockout. Discuss real-time applicability in web auth systems. (Wipro, 2025)

29. Explain the role of expressions and operators in building efficient control structures; illustrate with a program that categorizes employee performance scores into grades. (Accenture, 2025)

30. Apply Python basics to design a function that processes a list of temperatures (input via stdin) and identifies heatwave days (>35°C) using loops. (Google, 2025)

31. Discuss how command-line arguments could automate batch processing in a student management system, design functions to read student details from input and compute averages using operators. (Meta, 2025)

Dr. SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE (Autonomous)
Accredited by NAAC (Cycle-IV) with 'A+' Grade,
(Recognized by UGC & Approved by AICTE, New Delhi and Affiliated to Bharathiar University, Coimbatore)
486, Thudiyalur-Saravanampatti Road, Chinnavedampatti (Post), Coimbatore - 641 049.

Subject: GENERIC / CLUSTER CORE -3: OOPS WITH PYTHON PROGRAMMING **Code:** 25UCU409

QUESTION AND ANSWER
UNIT: 2

1. Describe the different categories of functions available in Python (built-in, user-defined, anonymous) and give one clear example of each type. (Infosys, 2024)

2. Explain the purpose of default parameters, keyword arguments, and variable-length arguments in Python functions and show how they make functions more flexible. (Apple, 2023)

3. Write a recursive function that calculates the factorial of a given positive integer and clearly indicate which line is the base case that stops recursion. (Oracle, 2025)

4. Analyze what happens to variable visibility when the same name is used in both global and local scope, and explain how the global and nonlocal keywords change behavior. (TCS, 2024)

5. Evaluate whether it is better to use a lambda function or a regular def function when you only need a short, one-time-use function inside sorted() or map(). (Wipro, 2023)

6. Write a lambda expression that takes a list of numbers and returns only the even numbers using the filter() function. (Netflix, 2025)

7. Explain the two main syntax styles for importing modules (import module vs from module import name) and discuss when one style is preferred over the other. (Google, 2024)

8. Describe the steps required to create your own custom module containing several utility functions and then successfully import and use it in another script. (Microsoft, 2023)

9. Write a small program that imports the math module and uses it to calculate square root, power, and trigonometric values for given user inputs. (Amazon, 2025)

10. Compare the random.randint(), random.randrange(), and random.choice() functions and explain in which situations each one is most appropriate. (Meta, 2024)

11. Evaluate how the datetime module can help handle different time zones when building applications used by users across multiple countries. (Infosys, 2023)

12. Create a simple package structure that contains at least one submodule and demonstrate how to import and call a function from inside that submodule. (Apple, 2025)

13. Explain how sys.argv is used to read command-line arguments and write a short example that prints all arguments passed when the script is executed. (Oracle, 2024)

14. Describe what Python virtual environments are, why they are strongly recommended, and name two popular tools used to create and manage them. (TCS, 2023)

15. Explain how *args and **kwargs allow functions to accept a variable number of positional and keyword arguments, with one realistic example of each. (Google, 2024)

16. Write a recursive function to compute the nth Fibonacci number and discuss why naive recursion becomes inefficient for large n. (Amazon, 2025)

17. Analyze the potential problems caused by modifying a mutable default argument (e.g., def func(lst=[])) and explain the correct way to avoid this pitfall. (Microsoft, 2024)

18. Evaluate when it is appropriate to use a lambda function inside sorted(key=…) versus defining a separate named function for complex sorting logic. (Meta, 2023)

19. Create a higher-order function that accepts another function as an argument and applies it to every element of a list (similar to map but custom). (Infosys, 2025)

20. Describe how to create and use your own module that contains constants, functions, and classes, and how to prevent code from running when the module is imported. (TCS, 2024)

21. Write code that uses the os and pathlib modules to recursively list all .py files in a directory and its subdirectories. (Wipro, 2025)

22. Compare the datetime.date, datetime.time, datetime.datetime, and datetime.timedelta classes with respect to usage and available methods. (Apple, 2024)

23. Design a small utility package called “myutils” containing date formatting, string cleaning, and math helper functions. (Oracle, 2023)

24. Explain how virtual environments prevent “dependency hell” and demonstrate creating and activating one using venv (command-line steps). (Netflix, 2025)

25. Design a modular Python package for a logistics company using math and random modules to simulate delivery routes and estimate costs; include lambda functions for quick calculations. Explain scope issues in large teams. (Amazon, 2025)

26. Evaluate the use of virtual environments and packages when a team at a fintech firm needs consistent datetime and os module behavior across developers' machines. (TCS, 2025)

27. Analyze a scenario where recursion in a file-search function (using os module) causes stack overflow; propose an iterative alternative with sys module limits. (Infosys, 2025)

28. Apply anonymous functions (lambda) and modules to sort employee records by salary using random for tie-breaking simulation in HR analytics. (Accenture, 2025)

29. Create a custom module for date-time utilities using datetime and sys; demonstrate importing and using it in a main script for attendance tracking. (Wipro, 2025)

30. Understand and explain how variable scope in nested functions affects a recursive factorial implementation imported from a module; provide debug steps. (Meta, 2025)

31. A startup needs to generate secure random passwords; design using random modules and recursion/lambda, discussing package structure for reusability. (Google, 2025)

Dr. SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE (Autonomous)
Accredited by NAAC (Cycle-IV) with 'A+' Grade,
(Recognized by UGC & Approved by AICTE, New Delhi and Affiliated to Bharathiar University, Coimbatore)
486, Thudiyalur-Saravanampatti Road, Chinnavedampatti (Post), Coimbatore - 641 049.

Subject: GENERIC / CLUSTER CORE -3: OOPS WITH PYTHON PROGRAMMING **Code:** 25UCU409

QUESTION AND ANSWER
UNIT: 3

1. Explain the core concepts of classes and objects in object-oriented programming in Python and demonstrate how to create and use an object from a class. (Wipro, 2024)

2. Describe the purpose of the __init__ constructor method and the __del__ destructor method and explain when each of them is automatically called. (Netflix, 2023)

3. Create a parent class Animal and a child class Dog using single inheritance, then show how the child class can use and extend the parent's methods. (Google, 2025)

4. Compare single inheritance, multiple inheritance, and multilevel inheritance with respect to code reusability, complexity, and potential problems. (Microsoft, 2024)

5. Demonstrate method overriding by creating a base class and a derived class where the same method name behaves differently depending on the object type. (Amazon, 2025)

6. Explain how abstract base classes (using the abc module) can be used to force subclasses to implement certain methods and show a small example. (Meta, 2024)

7. Write the complete syntax of a try-except-else-finally block and explain the purpose of each clause in exception handling. (Infosys, 2024)

8. Describe how encapsulation is achieved in Python using naming conventions (_ and __) and getter/setter methods, and discuss its benefits. (Apple, 2023)

9. Write code that opens a text file in read mode, reads all lines, and writes them to another file after converting every line to uppercase. (Oracle, 2025)

10. Explain how the pickle module is used to serialize and deserialize Python objects to and from binary files, including advantages and security concerns. (TCS, 2024)

11. Evaluate the situations in which raising a custom exception is more appropriate than using built-in exceptions like ValueError or TypeError. (Wipro, 2023)

12. Design a small class hierarchy (at least three levels) that demonstrates method overriding in a realistic domain such as different types of employees. (Netflix, 2025)

13. Explain what the Method Resolution Order (MRO) is in multiple inheritance and show how to view the MRO of a class using the __mro__ attribute or mro() method. (Google, 2024)

14. Describe how abstraction is implemented in Python using abstract methods and abstract classes, and explain why it improves code design. (Microsoft, 2023)

15. Discuss the diamond problem in multiple inheritance and how Python's Method Resolution Order (C3 linearization) resolves it. (Google, 2024)

16. Implement a Shape abstract base class with an abstract area() method, then create concrete Circle and Rectangle classes that implement it. (Amazon, 2025)

17. Analyze the trade-offs between composition and inheritance when designing class relationships (favor composition over inheritance principle). (Microsoft, 2023)

18. Evaluate the usefulness of __slots__ for memory optimization in classes that create millions of instances (e.g., data points in scientific computing). (Meta, 2024)

19. Create a BankAccount class with deposit, withdraw, and get_balance methods, including proper validation and custom InsufficientFunds exception. (Infosys, 2025)

20. Explain how context managers (with statement) work internally using __enter__ and __exit__ methods, and why they are preferred for file/database handling. (TCS, 2024)

21. Write code that reads a CSV-like text file using csv module and converts it into a list of dictionaries. (Wipro, 2025)

22. Compare text file encoding options (utf-8, latin-1, ascii) and discuss what can go wrong when the wrong encoding is used. (Apple, 2023)

23. Design a logging system that writes timestamped messages to a file and also prints severe messages to console using file handling. (Oracle, 2024)

24. Explain how to safely handle multiple exceptions in one except clause versus separate except blocks, including best practices for exception ordering. (Netflix, 2025)

25. Design a class hierarchy for an e-commerce platform (Product → Electronics → Smartphone) demonstrating single/multiple inheritance, polymorphism, and method overriding; handle file I/O for inventory persistence. (Amazon, 2025)

26. Evaluate encapsulation and abstraction in a banking system class with private attributes; justify exception handling for invalid withdrawals. (TCS, 2025)

27. Analyze polymorphism issues when overriding methods in multilevel inheritance for a vehicle rental app; propose solutions with super(). (Infosys, 2025)

28. Apply OOP to create classes for student records with constructors/destructors, saving/loading via text/binary files, and custom exceptions for invalid marks. (Accenture, 2025)

29. In a hospital management case, design exception handling for file not found when reading patient records; discuss try-except-else-finally best practices. (Wipro, 2025)

30. Understand constructors in multiple inheritance; illustrate with a diamond problem scenario in employee-payroll classes. (Meta, 2025)

31. Create a Python program using classes and file handling to manage library books, with polymorphism for different book types and exception handling for overdue fines. (Google, 2025)

Dr. SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE (Autonomous)
Accredited by NAAC (Cycle-IV) with 'A+' Grade,
(Recognized by UGC & Approved by AICTE, New Delhi and Affiliated to Bharathiar University, Coimbatore)
486, Thudiyalur-Saravanampatti Road, Chinnavedampatti (Post), Coimbatore - 641 049.

Subject: GENERIC / CLUSTER CORE -3: OOPS WITH PYTHON PROGRAMMING **Code:** 25UCU409

QUESTION AND ANSWER
UNIT: 4

1. Explain the fundamental difference between iterators and generators in Python and clarify why generators use the yield keyword instead of return. (Amazon, 2024)

2. Describe what decorators are, how the @ syntax works, and write a simple decorator that prints the execution time of any function. (Meta, 2023)

3. Write a regular expression pattern that validates whether a given string is a valid email address according to common industry rules. (Infosys, 2025)

4. Compare the performance and memory usage of collections.deque with a regular Python list when implementing queue and stack operations. (Apple, 2024)

5. Explain how itertools.product() and itertools.permutations() can be used to generate combinations useful in testing and simulation tasks. (Oracle, 2023)

6. Write a generator function that yields the first n numbers in the Fibonacci sequence without storing the entire sequence in memory. (TCS, 2025)

7. Describe the different ways to create a NumPy array and explain at least three basic slicing and indexing operations on multi-dimensional arrays. (Wipro, 2024)

8. Explain what a Pandas Series is and demonstrate how to create one from a Python dictionary and perform basic indexing and slicing. (Netflix, 2023)

9. Write Pandas code that creates a DataFrame and filters rows where a particular column value is greater than a given threshold. (Google, 2025)

10. Describe how Matplotlib can be used to create a line plot, bar chart, and scatter plot from NumPy arrays or Pandas DataFrames. (Microsoft, 2024)

11. Explain how functools.partial can be used to create new functions with some arguments already fixed and give one practical use case. (Amazon, 2023)

12. Write a decorator that logs the name of the function and the time it was called every time the decorated function runs. (Meta, 2025)

13. List and briefly describe at least five important metacharacters used in regular expressions along with their meaning. (Infosys, 2024)

14. Explain how functools.lru_cache decorator can dramatically improve performance in recursive or repeated function calls. (Apple, 2023)

15. Discuss why generators are memory-efficient compared to list comprehensions when processing very large datasets or infinite sequences. (Amazon, 2024)

16. Write a decorator that measures and prints the execution time of the wrapped function using time.perf_counter(). (Meta, 2025)

17. Create a regular expression that matches Indian mobile numbers (10 digits starting with 6–9) and validates several test strings. (Infosys, 2025)

18. Compare collections.namedtuple, dataclasses.dataclass (Python 3.7+), and regular classes for representing simple data records. (Google, 2024)

19. Explain how functools.reduce() works and use it to find the product of all elements in a list and to concatenate strings. (Microsoft, 2023)

20. Write NumPy code to create a 5×5 identity matrix, multiply every element by 4, and then extract the diagonal elements. (TCS, 2025)

21. Design a Pandas script that reads a CSV, groups data by category, calculates mean and count for each group, and sorts by mean descending. (Wipro, 2025)

22. Explain the concept of broadcasting in NumPy and demonstrate it with an example of adding a 1D array to every row of a 2D array. (Apple, 2024)

23. Compare matplotlib.pyplot (stateful) interface versus object-oriented matplotlib.axes usage in terms of flexibility and clarity. (Oracle, 2023)

24. Create a closure-based counter function that maintains its own internal count and returns the next value each time it is called. (Netflix, 2025)

25. Write code using itertools.groupby() to group and count consecutive identical elements in a list (run-length encoding style). (Google, 2025)

26. Compare the memory usage and performance of a generator expression versus a list comprehension when filtering a very large dataset that you only need to iterate once. (Microsoft, 2025)

27. Evaluate whether using @lru_cache on a recursive function is always a safe and effective optimization, and mention one situation where it can cause incorrect results. (Meta, 2025)

28. Create a custom iterator class that generates prime numbers up to a given limit using the Sieve of Eratosthenes logic inside __next__. (TCS, 2025)

29. Explain how NumPy’s vectorized operations achieve much higher performance than Python for-loops when performing element-wise arithmetic on large arrays. (Infosys, 2025)

30. Write Pandas code that merges two DataFrames on a common key column and then fills missing values in the resulting DataFrame using forward fill. (Wipro, 2025)

31. Analyze the difference between df.loc[] and df.iloc[] in Pandas and give examples where using the wrong one would produce incorrect results. (Cognizant, 2025)

32. Design a small script that uses Matplotlib to create a subplot grid showing line plot, bar chart, scatter plot, and histogram from the same dataset. (Accenture, 2025)

33. Explain how to create a custom decorator that can accept parameters (e.g., @repeat(3)) to repeat the decorated function multiple times. (Capgemini, 2025)

34. Write regular expression code that finds all floating-point numbers (with optional sign and decimal part) in a long string containing mixed text and numbers. (IBM, 2025)

35. Design a generator-based system for processing large log files (using itertools) in a cloud monitoring tool; compare with iterators for memory efficiency. (Amazon, 2025)

36. Evaluate decorators for timing and logging in a data pipeline function using functools; apply to Pandas DataFrame operations. (TCS, 2025)

37. Analyze regular expressions to validate email patterns in user registration; discuss limitations and security risks in web apps. (Infosys, 2025)

38. Apply NumPy arrays and operations to normalize sales data for a retail dashboard; visualize trends using Matplotlib. (Accenture, 2025)

39. In a case study for stock market analysis (industry-based), use Pandas Series/DataFrames to clean and analyze CSV data, then plot with Matplotlib; handle missing values. (Google, 2025)

40. Understand advanced modules like collections.defaultdict in frequency counting for text analytics; compare with regular dict. (Meta, 2025)

41. Create a decorator that caches function results (using functools.lru_cache) for expensive NumPy computations in scientific simulations. (Wipro, 2025)

Dr. SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE (Autonomous)
Accredited by NAAC (Cycle-IV) with 'A+' Grade,
(Recognized by UGC & Approved by AICTE, New Delhi and Affiliated to Bharathiar University, Coimbatore)
486, Thudiyalur-Saravanampatti Road, Chinnavedampatti (Post), Coimbatore - 641 049.

Subject: GENERIC / CLUSTER CORE -3: OOPS WITH PYTHON PROGRAMMING **Code:** 25UCU409

QUESTION AND ANSWER
UNIT: 5

1. Describe how to establish a connection to an SQLite database using Python's sqlite3 module and execute a simple SELECT query. (Oracle, 2024)

2. Explain the basic structure of a Flask web application and write the minimal code required to create a route that returns "Hello, World!". (TCS, 2023)

3. Write Python code using mysql-connector-python that inserts a new record into a students table with name, age, and grade columns. (Wipro, 2025)

4. Compare the MTV (Model-Template-View) architecture used in Django with the traditional MVC pattern found in many other web frameworks. (Netflix, 2024)

5. Explain how a basic linear regression model can be trained and evaluated using scikit-learn on a small synthetic dataset. (Google, 2023)

6. Outline the major steps and technologies you would use to build a simple student record management system that stores data in a text file or SQLite database. (Microsoft, 2025)

7. Describe the key components (models, views, templates, URLs) needed to create a basic CRUD application using the Django web framework. (Amazon, 2024)

8. Explain the typical steps involved in a machine learning project using Python, from data collection to model deployment. (Meta, 2023)

9. Write a small Flask REST API endpoint that accepts a GET request and returns a JSON response containing student details. (Infosys, 2025)

10. Analyze the advantages and limitations of building a simple rule-based chatbot using only if-elif conditions and string matching in Python. (Apple, 2024)

11. Describe how you would use Pandas to load a CSV file, clean missing values, and generate basic statistical summary of the dataset. (Oracle, 2023)

12. Outline the important features you would include in a basic web-based student attendance system built using Django or Flask. (TCS, 2025)

13. List and briefly explain the purpose of at least four commonly used scikit-learn classifiers suitable for the famous Iris dataset. (Wipro, 2024)

14. Explain how to execute parameterized SQL SELECT, INSERT, UPDATE, and DELETE queries safely using the sqlite3 module in Python. (Netflix, 2023)

15. Discuss the differences between Flask (micro-framework) and Django (full-stack framework) and when one should choose one over the other. (Google, 2024)

16. Write a Flask application with routes for home (/), about (/about), and a dynamic route (/user/) that greets the user. (Amazon, 2025)

17. Explain how Django models, migrations, and the ORM abstract SQL queries, and show a simple model definition with fields and relationships. (Microsoft, 2023)

18. Compare SQLite, MySQL, and PostgreSQL from the perspective of a small-to-medium Python web application in terms of setup, performance, and features. (Meta, 2024)

19. Design the database schema and Python code for a simple Library Management System (books, members, borrow/return) using SQLite. (Infosys, 2025)

20. Explain the basic supervised learning workflow using scikit-learn: load data → split train/test → scale → fit model → predict → evaluate. (TCS, 2024)

21. Write code using scikit-learn to perform k-means clustering on a small 2D dataset and visualize the clusters with matplotlib. (Wipro, 2025)

22. Discuss common data preprocessing steps (handling missing values, encoding categoricals, feature scaling) and why they are important before model training. (Apple, 2023)

23. Outline a complete mini-project plan for building a “Movie Recommendation System” using Pandas for data handling and simple similarity measures. (Netflix, 2025)

24. Explain how to create, read, update, and delete (CRUD) operations using Django’s ORM on a simple Todo model. (Oracle, 2024)

25. Describe the basic steps to connect to a MySQL database using mysql-connector-python, execute a SELECT query with parameters, and fetch results safely. (Oracle India, 2025)

26. Write a Flask route that accepts POST JSON data containing name and age, validates the input, and returns a success/failure JSON response. (Zoho, 2025)

27. Explain the purpose of Django’s settings.py file and list at least six important settings that almost every production Django project modifies. (Freshworks, 2025)

28. Compare the pros and cons of using SQLite versus PostgreSQL when building a small-to-medium internal company tool in Python. (Chargebee, 2025)

29. Create a simple scikit-learn pipeline that includes StandardScaler, PCA (2 components), and LogisticRegression, then fit it on a small dataset. (Fractal Analytics, 2025)

30. Discuss why feature scaling is usually necessary before applying distance-based algorithms like K-Nearest Neighbors or SVM. (Mu Sigma, 2025)

31. Outline the high-level architecture you would choose for a small internal dashboard that shows daily sales summary (Flask + SQLite + Pandas + Plotly). (Swiggy-style analytics, 2025)

32. Explain how Django’s class-based views (especially ListView, DetailView, CreateView) reduce boilerplate code compared to function-based views. (Zerodha tech team, 2025)

33. Write code that uses sqlite3 to create a table called employees, insert 5 sample rows, and then query employees whose salary is above average. (Paytm tech, 2025)

34. Analyze the trade-offs between building a simple chatbot using rule-based if-elif patterns versus using a pre-trained small language model via Hugging Face transformers. (Yellow.ai / Haptik style, 2025)

35. Design a SQLite database connectivity script for a student record system; implement CRUD operations and discuss migration to MySQL for production. (TCS, 2025)

36. Evaluate Flask vs. Django for a simple REST API serving student grades; justify choice for a startup with limited resources. (Infosys, 2025)

37. In an industry case for customer churn prediction, use Scikit-learn basics (e.g., train-test split, simple classifier) on Pandas data; discuss ethical considerations. (Amazon, 2025)

38. Apply Matplotlib and Pandas to visualize sales trends from a CSV in a business intelligence mini-project; extend to basic ML clustering. (Accenture, 2025)

39. Create a mini chatbot using basic Python (loops, conditionals, functions) that queries a SQLite DB for FAQs in customer support. (Wipro, 2025)

40. Understand Python's role in data science pipelines; explain integration of NumPy/Pandas with Scikit-learn for a predictive model in e-commerce. (Meta, 2025)

41. Design and propose a complete mini-project architecture (e.g., web app for data analysis using Flask, Pandas, and SQLite) for a small business analytics tool; include deployment considerations. (Google, 2025)