**1. Python Basics & Syntax**

1. **Q:** What are the key features of Python?  
   **A:** Python is interpreted, high-level, dynamically typed, and supports object-oriented, functional, and procedural programming paradigms.
2. **Q:** How do you write a comment in Python?  
   **A:** Single-line comments use the # symbol. Multi-line comments can be written using triple quotes (''' or """).
3. **Q:** What is the purpose of indentation in Python?  
   **A:** Indentation defines blocks of code. It replaces braces {} and is mandatory to indicate block structure like loops or conditionals.
4. **Q:** How do you take input from the user in Python?  
   **A:** Use the input() function. Example: name = input("Enter your name: ").
5. **Q:** How do you print output to the screen in Python?  
   **A:** Use the print() function. Example: print("Hello World").

**2. Strings & Numbers**

1. **Q:** How do you check if a string contains only digits?  
   **A:** Use the isdigit() method: "123".isdigit() returns True.
2. **Q:** How do you convert a float to a string?  
   **A:** Use the str() function. Example: str(3.14) returns '3.14'.
3. **Q:** What is the difference between isnumeric() and isdigit()?  
   **A:** isnumeric() returns True for all numeric characters including fractions and superscripts. isdigit() is limited to decimal digits.
4. **Q:** How do you format a string using placeholders?  
   **A:** Use f-strings or str.format(). Example: f"Hello {name}" or "Hello {}".format(name).
5. **Q:** How do you round a number to 2 decimal places in Python?  
   **A:** Use round(number, 2). Example: round(3.14159, 2) returns 3.14.

**3. Lists & Tuples**

1. **Q:** What is the difference between a list and a tuple in Python?  
   **A:** Lists are mutable and defined using [], while tuples are immutable and defined using ().
2. **Q:** How do you add an item to a list?  
   **A:** Use append() to add at the end. Example: my\_list.append(10).
3. **Q:** How do you slice a list in Python?  
   **A:** Use [start:stop:step] syntax. Example: my\_list[1:4] returns items at index 1 to 3.
4. **Q:** How can you unpack a tuple?  
   **A:** Use multiple variables. Example: a, b = (1, 2).
5. **Q:** How do you combine two lists?  
   **A:** Use the + operator. Example: list1 + list2.

**4. Sets & Dictionaries**

**Q:** What is the difference between a set and a dictionary in Python?  
**A:** Sets store unique values without keys, while dictionaries store key-value pairs.

**Q:** How do you add an element to a set?  
**A:** Use the add() method. Example: my\_set.add(5).

**Q:** How do you remove a key from a dictionary?  
**A:** Use the del keyword or pop() method. Example: del my\_dict['key'] or my\_dict.pop('key').

**Q:** What happens when you add a duplicate element to a set?  
**A:** The set ignores duplicates and keeps only unique elements.

**Q:** How do you check if a key exists in a dictionary?  
**A:** Use the in keyword. Example: 'key' in my\_dict.

**5. Conditionals & Loops**

**Q:** What are the different conditional statements in Python?  
**A:** Python uses if, elif, and else for conditional branching.

**Q:** How do you write a while loop?  
**A:** Use while with a condition. Example: while x < 5:.

**Q:** What is the difference between break and continue?  
**A:** break exits the loop, continue skips to the next iteration.

**Q:** How do you loop over a range of numbers?  
**A:** Use the range() function with a for loop. Example: for i in range(5):.

**Q:** What is a nested loop?  
**A:** A loop inside another loop, used for multi-dimensional data.

**6. Functions**

**Q:** How do you define a function in Python?  
**A:** Use the def keyword. Example: def greet():.

**Q:** What is the purpose of the return statement?  
**A:** It returns a value from a function to the caller.

**Q:** What are default arguments in functions?  
**A:** Arguments that take default values if not provided by the caller.

**Q:** What is the difference between \*args and \*\*kwargs?  
**A:** \*args collects positional arguments, \*\*kwargs collects keyword arguments.

**Q:** Can you define a function inside another function?  
**A:** Yes, Python supports nested functions.

**7. Lambda & Map, Filter, Reduce**

**Q:** What is a lambda function in Python?  
**A:** A lambda function is an anonymous, single-expression function defined using the lambda keyword.

**Q:** How is the map() function used in Python?  
**A:** map() applies a function to each item of an iterable. Example: map(str.upper, ['a', 'b']).

**Q:** What does the filter() function do?  
**A:** filter() returns items from an iterable for which a given function returns True.

**Q:** What is the purpose of reduce() in Python?  
**A:** reduce() applies a rolling computation to a sequence and returns a single value. It’s part of the functools module.

**Q:** When should you use lambda over def?  
**A:** Use lambda for short, throwaway functions that are used immediately and don’t require naming.

**8. Comprehensions**

**Q:** What is list comprehension in Python?  
**A:** A concise way to create lists using a single line of code. Example: [x for x in range(5)].

**Q:** How do you write dictionary comprehension?  
**A:** By specifying key-value pairs: {x: x\*x for x in range(3)}.

**Q:** Can you use if-else in list comprehension?  
**A:** Yes. Example: [x if x % 2 == 0 else 0 for x in range(5)].

**Q:** What are the benefits of using comprehensions?  
**A:** They are more concise, readable, and often more efficient than using loops.

**Q:** Does Python support set comprehensions?  
**A:** Yes. Example: {x for x in range(5)} creates a set of numbers from 0 to 4.

**9. Exception Handling**

**Q:** How do you handle exceptions in Python?  
**A:** Using try, except, finally, and optionally else blocks.

**Q:** What is the difference between except Exception and except?  
**A:** except Exception catches specific exceptions, while a bare except catches all errors (not recommended).

**Q:** How do you raise an exception manually?  
**A:** Use the raise keyword. Example: raise ValueError("Invalid input").

**Q:** What is a custom exception in Python?  
**A:** A user-defined exception created by subclassing the Exception class.

**Q:** What is the purpose of the finally block?  
**A:** It executes cleanup code regardless of whether an exception occurred.

**10. Modules & Packages**

**Q:** What is the difference between a module and a package?  
**A:** A module is a single .py file. A package is a directory with an \_\_init\_\_.py file containing multiple modules.

**Q:** How do you import a module in Python?  
**A:** Use import module\_name or from module\_name import item.

**Q:** What is the purpose of \_\_init\_\_.py?  
**A:** It marks a directory as a Python package and can contain initialization code.

**Q:** How can you access functions from another file?  
**A:** Import the file as a module and call the function using dot notation.

**Q:** How do you explore module contents?  
**A:** Use dir(module) to list attributes and help(module) for documentation.

**11. Object-Oriented Programming (OOP) – 1**

**Q:** What is object-oriented programming in Python?  
**A:** OOP is a programming paradigm based on the concept of "objects" which contain data (attributes) and code (methods).

**Q:** How do you define a class in Python?  
**A:** Use the class keyword. Example: class Person:.

**Q:** What is the purpose of the \_\_init\_\_ method?  
**A:** It’s the constructor method that gets called automatically when an object is created.

**Q:** How do you create an object from a class?  
**A:** By calling the class like a function. Example: p = Person().

**Q:** What are instance variables?  
**A:** Variables that are specific to each object, typically defined using self.

**12. Object-Oriented Programming (OOP) – 2**

**Q:** What is inheritance in Python?  
**A:** Inheritance allows one class to inherit attributes and methods from another class.

**Q:** How do you override a method in a child class?  
**A:** Define a method with the same name as in the parent class.

**Q:** What is polymorphism?  
**A:** Polymorphism allows methods with the same name to behave differently depending on the object type.

**Q:** What are class methods and how do you define them?  
**A:** Class methods affect the class as a whole and use the @classmethod decorator and cls as the first parameter.

**Q:** What is encapsulation?  
**A:** Encapsulation hides internal object details using private attributes (e.g., \_\_balance).

**13. File Handling**

**Q:** How do you open a file in Python?  
**A:** Use the open() function. Example: open("file.txt", "r").

**Q:** What are the modes used for file operations?  
**A:** Common modes include "r" (read), "w" (write), "a" (append), and "b" (binary).

**Q:** How do you write content to a file?  
**A:** Use the write() method. Example: f.write("Hello").

**Q:** What is the purpose of using with open()?  
**A:** It automatically handles file closing, even if an exception occurs.

**Q:** How do you read a file line by line?  
**A:** Use a for loop: for line in file: or use readlines().

**14. Working with JSON**

**Q:** How do you convert a Python dictionary to a JSON string?  
**A:** Use json.dumps(dict).

**Q:** How do you convert a JSON string to a Python dictionary?  
**A:** Use json.loads(json\_string).

**Q:** How do you read JSON data from a file?  
**A:** Use json.load(file\_object).

**Q:** How do you write JSON data to a file?  
**A:** Use json.dump(data, file\_object).

**Q:** What module is used to work with JSON in Python?  
**A:** The built-in json module.

**15. Virtual Environments & pip**

**Q:** What is a virtual environment in Python?  
**A:** A virtual environment is an isolated environment that allows you to manage dependencies separately from the system Python.

**Q:** How do you create a virtual environment?  
**A:** Use python -m venv env\_name.

**Q:** How do you activate a virtual environment?  
**A:** On Windows: env\_name\\Scripts\\activate, on Unix: source env\_name/bin/activate.

**Q:** How do you install packages using pip?  
**A:** Use pip install package\_name.

**Q:** How do you freeze and save installed packages to a file?  
**A:** Use pip freeze > requirements.txt.

**16. Web Scraping using BeautifulSoup**

**Q:** What is web scraping in Python?  
**A:** Web scraping is the process of extracting data from websites using tools like requests and BeautifulSoup.

**Q:** What is BeautifulSoup used for?  
**A:** BeautifulSoup is a library used to parse HTML/XML documents and extract data from specific tags or structures.

**Q:** How do you find all <a> tags on a webpage?  
**A:** Use soup.find\_all('a') after parsing the HTML with BeautifulSoup.

**Q:** What is the purpose of the requests module in web scraping?  
**A:** It is used to send HTTP requests and retrieve web page content.

**Q:** How can you get the text content of an HTML element?  
**A:** Use .text or .get\_text() on a BeautifulSoup element.

**17. APIs & Requests**

**Q:** What is an API?  
**A:** API (Application Programming Interface) is a way for programs to communicate with each other, often over the web.

**Q:** How do you make a GET request using Python?  
**A:** Use requests.get(url) from the requests library.

**Q:** How do you send data in a POST request?  
**A:** Use requests.post(url, data=payload) where payload is a dictionary of form data.

**Q:** How do you convert JSON response to a dictionary?  
**A:** Use .json() method on the response object. Example: response.json().

**Q:** How do you pass headers in an API request?  
**A:** Use the headers parameter: requests.get(url, headers={'Authorization': 'Token'}).

**18. Pandas Basics**

**Q:** What is Pandas in Python?  
**A:** Pandas is a library used for data manipulation and analysis, built on top of NumPy.

**Q:** What is a DataFrame?  
**A:** A DataFrame is a 2D labeled data structure with columns of potentially different types.

**Q:** How do you read a CSV file into a DataFrame?  
**A:** Use pd.read\_csv('file.csv').

**Q:** How do you filter rows based on a condition?  
**A:** Use boolean indexing. Example: df[df['Age'] > 30].

**Q:** How do you handle missing values in Pandas?  
**A:** Use df.dropna() to remove or df.fillna(value) to replace missing values.

**19. NumPy Basics**

**Q:** What is NumPy used for?  
**A:** NumPy is used for numerical operations, including working with arrays, matrices, and performing mathematical computations.

**Q:** How do you create a NumPy array?  
**A:** Use np.array([1, 2, 3]).

**Q:** What is the difference between a list and a NumPy array?  
**A:** NumPy arrays are faster, consume less memory, and support element-wise operations.

**Q:** How do you generate a range of numbers in NumPy?  
**A:** Use np.arange(start, stop, step).

**Q:** How do you perform matrix multiplication in NumPy?  
**A:** Use np.dot(A, B) or A @ B.

**20. Matplotlib for Data Visualization**

**Q:** What is Matplotlib?  
**A:** Matplotlib is a Python library for creating static, animated, and interactive visualizations like line plots, bar charts, and scatter plots.

**Q:** How do you create a simple line plot?  
**A:** Use plt.plot(x, y) followed by plt.show().

**Q:** How do you add a title and axis labels to a plot?  
**A:** Use plt.title(), plt.xlabel(), and plt.ylabel().

**Q:** How do you display multiple plots in the same figure?  
**A:** Use plt.subplot() to define multiple subplots.

**Q:** How do you save a plot as an image file?  
**A:** Use plt.savefig('filename.png').

**21. Plotly for Interactive Charts**

**Q:** What is Plotly used for?  
**A:** Plotly is a library for building interactive and web-ready visualizations.

**Q:** How do you create a basic bar chart using Plotly?  
**A:** Use plotly.express.bar() with required data.

**Q:** What makes Plotly charts interactive?  
**A:** Hover tooltips, zooming, and clickable legends are built-in.

**Q:** Can Plotly be used offline?  
**A:** Yes, by importing and using plotly.offline.plot().

**Q:** What is the difference between Plotly and Matplotlib?  
**A:** Matplotlib is static; Plotly supports interactivity and web embedding.

**22. FastAPI Basics**

**Q:** What is FastAPI?  
**A:** FastAPI is a modern web framework for building APIs quickly with Python and automatic documentation.

**Q:** How do you define a GET route in FastAPI?  
**A:** Use @app.get("/route") above a function.

**Q:** What is automatic documentation in FastAPI?  
**A:** FastAPI auto-generates Swagger and Redoc docs at /docs and /redoc.

**Q:** How do you run a FastAPI app?  
**A:** Use uvicorn main:app --reload.

**Q:** How do you validate request data in FastAPI?  
**A:** Use Pydantic models as function parameters.

**23. SQL Basics with Python**

**Q:** How do you connect to an SQLite database in Python?  
**A:** Use sqlite3.connect('database.db').

**Q:** How do you execute a SQL query in Python?  
**A:** Use a cursor object: cursor.execute("SELECT \* FROM table").

**Q:** How do you fetch all results of a query?  
**A:** Use cursor.fetchall().

**Q:** What module is used for working with SQL in Python?  
**A:** sqlite3 for SQLite; other DBs use psycopg2, sqlalchemy, etc.

**Q:** How do you prevent SQL injection?  
**A:** Use parameterized queries. Example: cursor.execute("SELECT \* FROM users WHERE id = ?", (user\_id,)).

**24. Data Validation with Pydantic**

**Q:** What is Pydantic?  
**A:** Pydantic is a library for data validation and settings management using Python type hints.

**Q:** How do you define a model in Pydantic?  
**A:** Inherit from BaseModel. Example: class User(BaseModel): name: str.

**Q:** What happens if validation fails in Pydantic?  
**A:** A ValidationError is raised with detailed error info.

**Q:** How do you provide default values in Pydantic models?  
**A:** Assign values in the class definition. Example: age: int = 18.

**Q:** Can Pydantic parse nested models?  
**A:** Yes, Pydantic supports nested data structures using model composition.

**25. Sending Emails using Python**

**Q:** What module is used to send emails in Python?  
**A:** smtplib is commonly used for sending emails.

**Q:** How do you create an email message?  
**A:** Use email.message.EmailMessage() to compose structured emails.

**Q:** How do you connect to an SMTP server?  
**A:** Use smtplib.SMTP('smtp.gmail.com', 587) and call .starttls().

**Q:** How do you log in to send an email?  
**A:** Use server.login("your\_email", "your\_password").

**Q:** Can you send attachments in Python emails?  
**A:** Yes, using EmailMessage().add\_attachment() or MIME.

**26. Working with PDFs & Excel Files**

**Q:** How do you read a PDF file in Python?  
**A:** Use PyPDF2 or pdfplumber to extract text.

**Q:** How do you read Excel files in Python?  
**A:** Use pandas.read\_excel('file.xlsx').

**Q:** How do you write to an Excel file?  
**A:** Use pandas.DataFrame.to\_excel('output.xlsx').

**Q:** What module is used to manipulate Excel files?  
**A:** openpyxl for .xlsx, xlrd for .xls.

**Q:** How do you merge multiple PDF files?  
**A:** Use PyPDF2.PdfMerger().

**27. Python Automation (Scheduling, Tasks)**

**Q:** What are common automation tasks in Python?  
**A:** File renaming, email alerts, backups, browser automation, and report generation.

**Q:** How do you schedule tasks in Python?  
**A:** Use schedule or APScheduler to run jobs periodically.

**Q:** What is cron and can Python interact with it?  
**A:** Cron is a Unix scheduler. Python scripts can be scheduled via crontab.

**Q:** How do you run a Python script at startup?  
**A:** Add it to OS-specific startup folders or task scheduler.

**Q:** How do you automate web interactions in Python?  
**A:** Use Selenium for browser automation.