ChatGPT

Python Topics and Interview Questions for Senior Data Engineers

Relevant Python Topics (Beginner to Advanced)

- **Core Python Programming:** Syntax, variables, data types, control flow (if/else, loops), functions, modules, exceptions, common built-ins, memory management.
- **Data Structures:** Lists, tuples, dictionaries, sets, and their operations (indexing, slicing, methods like append, extend, comprehensions).
- **Object-Oriented Programming (OOP):** Classes and objects, inheritance, polymorphism, encapsulation, special methods (`__init__`, `__new__`, `__str__`, etc.), design patterns (e.g., Singleton).
- **Functional Programming:** Lambdas, higher-order functions (`map()`, `filter()`, `reduce()`), list/set/dict comprehensions, closures, generator expressions, use of `functools`.
- **Error Handling:** `try` / `except` / `finally` blocks, raising and custom exceptions, error propagation, logging of errors, use of `with` context managers for resource management.
- **File I/O and Data Formats:** Reading/writing files (text, binary), file modes (`r`, `w`, `a`, `rb`, `wb`, etc.), context managers (`with open(...)`), CSV and JSON parsing (built-in `csv` and `json` modules), handling large files and streaming data.
- **Iterators & Generators:** Iterable vs iterator protocols, implementing `__iter__()` and `__next__()`, generator functions with `yield`, generator expressions, differences between lists and generators (memory efficiency), use of `itertools`.
- **Decorators:** Function decorators (`@decorator` syntax), creating and applying decorators (with and without arguments), use of built-in decorators (`@staticmethod`, `@classmethod`, `@property`), chaining decorators, decorator patterns.
- **Context Managers:** Writing classes with `__enter__` and `__exit__` methods, using `contextlib` for creating context managers, the `with` statement, ensuring resource cleanup (files, locks, DB transactions).
- **Pythonic Idioms:** List/dict/set comprehensions, unpacking operators (`*`, `**`), `enumerate()` and `zip()`, EAFP ("Easier to Ask Forgiveness than Permission") vs LBYL (Look Before You Leap), use of built-ins (`any()`, `all()`, `get()` on dicts).
- **Concurrency & Parallelism:** Differences between concurrency and parallelism, Python's Global Interpreter Lock (GIL), when to use multithreading (I/O-bound tasks) vs multiprocessing (CPU-bound tasks), `threading` and `multiprocessing` modules, synchronization primitives (Locks, Queues), introduction to `asyncio` (`async` / `await`) and async I/O.
- **Performance Optimization & Memory Management:** Profiling (`cProfile`, `timeit`), algorithmic complexity, using efficient data structures, avoiding Python overhead in loops, employing C-optimized libraries (NumPy, pandas), caching/memoization (e.g., `functools.lru_cache`), processing large datasets in chunks (using generators or libraries like Dask), leveraging parallelism.
- **Type Hints and Static Analysis:** Using type annotations (PEP 484) for functions and variables, static type checkers (e.g., MyPy), benefits of type hints in large codebases, `typing` module generics, enforcing types in CI.

- **Testing:** Writing unit and integration tests (`unittest`, `pytest`), test-driven development (TDD) practices, mocking and patching, assertions vs exceptions, test coverage, continuous integration.
- **Package and Environment Management:** Using `pip` for package installation, creating isolated environments with `venv` or `conda`, specifying dependencies (`requirements.txt`, `pyproject.toml`), packaging and distributing libraries, version management.
- **Standard Library and Common Modules:** Familiarity with modules like `os`, `sys`, `logging`, `datetime`, `re` (regex), `json`, `csv`, `io`, `subprocess`, `functools`, `itertools`, etc.
- **ETL and Data Pipelines:** Concepts of Extract-Transform-Load (ETL), designing data pipelines, using frameworks (Airflow, Luigi, etc.) in Python, incremental vs full data loads, orchestration, logging, and error handling in pipelines.
- **Working with APIs and Data Formats:** Consuming REST APIs with `requests` or `httpx`, handling JSON/XML data, authentication (API keys, OAuth), pagination of API results, rate limiting, scraping web data (e.g., with `BeautifulSoup`), and parsing data formats (JSON with `json` module, XML with `xml.etree` or `lxml`).
- **Database Interaction (SQL and NoSQL):** Using Python database drivers (`psycopg2` for PostgreSQL, `mysql-connector`, etc.), ORM libraries like SQLAlchemy, executing SQL queries, preventing SQL injection (parameterized queries), transactions (`commit()` / `rollback()`), reading/writing with Pandas (`read_sql`, `to_sql`), connecting to NoSQL stores (e.g., `pymongo` for MongoDB), query optimization.
- **Pandas and Large Data Processing:** Data manipulation with Pandas (`Series`, `DataFrame`), reading/writing structured data (`read_csv`, `to_sql`), filtering, grouping, merging/joining, handling missing or categorical data, vectorized operations vs Python loops, working with larger-than-memory data (chunking, Dask).
- **Automation and Scripting:** Writing scripts to automate tasks (file management, API data pulls), scheduling jobs (cron, Task Scheduler, Python's `schedule` library), working with file and OS operations (`os`, `glob`), sending alerts (email via `smtplib` or messaging), subprocess execution, handling file paths and permissions.

Python Interview Questions (Grouped by Topic)

Core Python Programming

- What is Python? [1]
- What are the key features of Python? [2]
- How do you declare a variable in Python? [3]
- What is PEP 8 and why is it important? [4]
- How does Python manage memory? [5]
- What are Python's built-in types (e.g. int, list, dict, etc.)? [6]
- How do you handle exceptions in Python (try/except)? [7]
- What does the `id()` function do in Python? [8]
- How do you test Python code (which testing frameworks would you use)? [9]
- What is the purpose of the `inspect` module? [10]

Data Structures (Lists, Dictionaries, Sets, Tuples)

- What is the difference between a list and a tuple in Python? [11]
- How do list comprehensions work, and when should you use them? [12]
- What is a dictionary in Python, and how do you create one? [13]
- What is a set in Python, and how do you create one? [14]

- What is the difference between the `append()` and `extend()` methods of a list? 15

Object-Oriented Programming (OOP)

- What is the difference between `__new__` and `__init__` methods when creating a class instance? 16
- How would you implement the Singleton design pattern in Python? 17
- What is Python's data model and why is it significant? (Discuss special methods and object creation) 18

Functional Programming

- What is a lambda (anonymous) function in Python? 19
- What is a generator in Python, and how is it different from a list? 20
- What is the difference between a normal function and a generator function? 21
- Can you explain what a closure is in Python? 22
- How does the `map()` function work in Python? 23
- What does the `filter()` function do in Python? 24
- What does the `reduce()` function do in Python, and which module is it in? 25 26
- What is the purpose of the `functools` module? (Name some tools it provides) 27

Error Handling

- How do you handle exceptions and errors in Python (using try/except/finally)? 7
- *(See also Context Managers for resource cleanup)*

File I/O and Data Formats

- Which function is used to open a file for reading in Python? 28
- Which mode is used to open a file for writing in Python? 29
- What function do you use to read the entire contents of a file as a string? 30
- In Python, what does the `readline()` function do? 31
- How do you open and read a file using a context manager (`with` statement)? 32
- How do you write data to a file in Python? 33
- How do you append data to an existing file without overwriting it? 34
- How do you handle CSV files in Python? (e.g., using `csv` module or Pandas) 35
- How do you handle JSON files in Python? (Parsing JSON data) 36

Iterators and Generators

- What is the purpose of the `yield` keyword in Python? 37
- What is the difference between an iterator and a generator in Python? 38
- What is the difference between the `__iter__()` and `__next__()` methods? 39

Decorators

- What is a decorator in Python and how do you use it? (Explain the `@decorator` syntax) 40

Context Managers

- What is a context manager in Python, and how is it different from using `try...finally`? 41

- Which methods must a class implement to be used as a context manager (`with` statement)? [41]
- Why are `with` blocks considered more Pythonic than `try/finally`? [42]
- Give an example use case for a context manager in Python. (Describe when custom context managers are helpful) [43]

Pythonic Idioms

- *(Common idioms like EAFP vs LBYL, tuple/list unpacking,* `enumerate`*,* `zip` *– no specific citations)*

Concurrency (Threads, Async)

- What is the difference between concurrency and parallelism? [44]
- What is the Global Interpreter Lock (GIL) in Python and why does it matter? [45]
- When would you use multi-threading in Python? (Give examples of I/O-bound tasks) [46]
- How do you start threads in Python? (Example with `threading.Thread`) [47]
- When would you use multiprocessing in Python? (Give examples of CPU-bound tasks) [48]
- How do you use the `multiprocessing` module in Python, and how does it differ from `threading`? [49]
- How is `asyncio` (async/await) used in Python? (E.g., what are `async` and `await` keywords?) [50]
- What is an event loop in Python's `asyncio`, and how does `asyncio.run()` work? (No direct citation but conceptually important)

Performance Optimization & Memory

- How do you optimize Python code for better performance? (Use built-in libraries, avoid global variables, etc.) [51]
- How would you handle large datasets in Python without running out of memory? (Using generators, NumPy, Dask, etc.) [52]

Testing

- How do you test Python code? Which frameworks can you use? [9]
- What is the `unittest` module in Python and how do you write tests with it? [53]

Package Management

- What is `pip` and what is it used for? [54]

Standard Library

- Which Python standard library is commonly used for regular expressions? (Answer: `re`) [55]

Regular Expressions (Regex)

- What is a regular expression in Python? [56]
- Which Python library provides support for regular expressions? [55]
- What does the `re.search()` function do in Python? [57]

ETL and Data Pipelines

- What is an ETL (Extract-Transform-Load) pipeline, and how would you build one in Python? [58]

- How do you handle failures or errors in an ETL pipeline? (e.g., logging, retries) 59
- How do you handle duplicate or bad data in an ETL process? (e.g., using `drop_duplicates()`, validation) 60
- How do you handle time-based or incremental loads in an ETL process? 61
- What is the difference between batch and streaming data pipelines? 62
- How do you test an ETL pipeline to ensure data quality? (Unit tests, sample data, etc.) 63
- How would you design a fault-tolerant, scalable ETL pipeline in Python? (Mention orchestrators like Airflow, parallel processing with Dask, error handling) 64

APIs and Web Scraping

- How do you make an HTTP GET request to a web API in Python? 65
- How do you handle API errors or failed requests in Python? (Timeouts, status codes, `try/except`) 66
- How do you include headers and query parameters in a Python API request? (e.g., authentication tokens) 67
- How do you handle paginated API responses in Python? 68
- How do you authenticate with an API that requires a token or API key? 69
- How do you scrape a webpage using BeautifulSoup (or similar tools) in Python? 70
- How can you avoid getting blocked when web scraping (e.g., user agents, delays, proxies)? 71

Database Interaction (SQL & NoSQL)

- How do you connect to an SQL database (e.g., PostgreSQL, MySQL) in Python? 72
- What is SQLAlchemy and why might a data engineer use it? 73
- How do you insert data from a Pandas DataFrame into an SQL table in Python? 74
- How do you prevent SQL injection in Python database code? (Use parameterized queries) 75
- How do you efficiently load large query results from SQL in Python? (e.g., using `chunksize` in pandas) 76
- How do you handle database transactions in Python? (commit, rollback patterns) 77
- What is the difference between relational (SQL) and non-relational (NoSQL) databases? 78
- How do you connect to and query a MongoDB (NoSQL) database from Python? 79

Pandas and Large Data Processing

- What is pandas and why is it useful for data engineers? 80
- What are a pandas Series and DataFrame? 81
- How do you read data from a CSV file using pandas? 82
- How do you filter rows in a pandas DataFrame? 83
- How do you select specific columns from a DataFrame? 84
- How do you handle missing values in pandas? (e.g., `dropna()`, `fillna()`) 85
- How do you change the data type of a pandas column? (e.g., `astype()`) 86
- How do you sort a DataFrame by one or more columns? 87
- How do you remove duplicate rows in a DataFrame? 88
- How do you group data in pandas and apply aggregations? (Explain `groupby` → apply → combine) 89
- How do you merge or join two DataFrames? (Use `pd.merge`, SQL-style joins) 90
- How do you apply a function to a DataFrame column or row? (e.g., using `apply()`) 91
- How do you detect and remove outliers in pandas? (e.g., using quantiles, IQR) 92

Automation and Scripting

- How do you automate a daily task using Python (e.g., scheduling with cron)? [93]
- How do you rename multiple files in a folder using Python? (e.g., `os.rename`, `glob`) [94]
- How do you send an email alert from a Python script? (Using `smtplib` and `email` modules) [95]
- How do you run a Python script from another Python script? (Import vs `subprocess`) [96]
- How do you schedule a Python script to run hourly or at regular intervals? (Cron or scheduling libraries) [97]

Sources: Interview question examples and discussions from Medium, GeeksforGeeks, Analytics Vidhya, DataCamp, InterviewQuery, ResumeGuru, and other reputable Python and data engineering Q&A resources [1] [72]. These cover the breadth of topics listed above and are cited accordingly.

---

[1] [2] [3] [4] [5] [6] [7] [9] [10] [11] [13] [14] [15] [16] [17] [18] [19] [27] [37] [45] [49] [51] [54] **Top 100 Python Interview Questions for Data Engineers | by Rahul Sounder | Medium**
https://medium.com/@sounder.rahul/top-100-python-interview-questions-for-data-engineers-2c87b9646db5

[8] [12] [38] [39] [41] [42] [43] [50] [53] **Python Interview Questions. Part II. Middle**
https://luminousmen.com/post/python-interview-questions-middle/

[20] [21] [40] **Python interview Questions — Intermediate level | by chanduthedev | Medium**
https://chanduthedev.medium.com/python-interview-questions-intermediate-level-c7ce66342e2d

[22] **Python Closures | GeeksforGeeks**
https://www.geeksforgeeks.org/python-closures/

[23] [24] [25] [26] **30+ MCQs on Python Map, Filter and Reduce Functions**
https://www.analyticsvidhya.com/blog/2024/02/mcqs-on-python-map-filter-and-reduce-functions/

[28] [29] [30] [31] **30+ MCQs on Python File I/O**
https://www.analyticsvidhya.com/blog/2024/02/mcqs-on-python-file-i-o/

120+ Python Interview Questions For Data Engineer
https://resumeguru.in/python-interview-questions-for-data-engineer/?srsltid=AfmBOooSmxXCGGLgROhbFz3jTHXKXH4dSH8sQ2zBEQ8fTOHYId3JhMlE

41 Python Data Engineer Interview Questions (2025 Update) | Examples & Answers
https://www.interviewquery.com/p/data-engineer-python-questions

30+ MCQs on Python Regular Expression
https://www.analyticsvidhya.com/blog/2024/02/mcqs-on-python-regular-expression/