Top 45 Python Full Stack Developer Interview Questions

26 Oct 2024 3 min read

Table of contents

The role of a Python Full Stack Developer offers a diverse range of skills and responsibilities, in both front-end and back-end development. As businesses increasingly grow on Python to develop robust applications, the demand for skilled full-stack developers proficient in Python is rising. Preparing for an interview in this field can be challenging due to the in-depth knowledge required. In this article, we will explore the most commonly asked Python full-stack interview questions and provide strategies to prepare them effectively.

Understanding the Role of a Python Fu

Developer

A Python Full Stack Developer is expected to handle both client-side and server-side development tasks. This role involves working with various technologies to create cohesive web applications. On the front end, developers use HTML, CSS, and JavaScript, often with frameworks like React or Angular. On the back end, Python, along with frameworks such as Django or Flask, is used to build the server-side logic, manage databases, and handle user authentication and authorization.

How to Prepare for a Python Full Stack Developer Interview

Preparing for a Python Full Stack Developer interview involves a combination of skills and knowledge across both front-end and back-end development. Here's a structured approach to get ready:

- Understand the Job Requirements
- Expertise with Core Python Skills such as Language Fundamentals, Libraries, and Frameworks
- Learn Frameworks, Database, Authentication, and Testing in backend development
- Proficiency with HTML/CSS, JavaScript, and Front-End Frameworks such as React, Angular, or Vue.js.
- Be aware of APIs and Deployment to connect and deploy applications.
- Be proficient with version control using Git
- Understand data structure and algorithms for problem-solving
- Understand the basics of system design and patterns

Top Most Asked Python Interview Questions

Here are the most asked Python full stack interview questions:

1. How does Python 2 differ from Python 3?

Python 3 introduced several improvements and changes compared to Python 2, such as print being a function (print()), integer division behavior (/ returns a float, // returns an integer), and new syntax features like f-strings.

2. How do you handle errors in Python?

Errors can be handled using try-except blocks. For example:

```
try:
    # code that might raise an exception
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Chat with us

3. What are the built-in data types in Python?

Common built-in data types include int, float, str, list, tuple, dict, set, and bool.

4. What is a list in Python?

A list is a mutable, ordered collection of items, which can be of different types. Example:

```python
my_list = [1, 2, 3, 'apple']
```

5. What is the difference between a tuple and a list?

Lists are mutable (can be changed), while tuples are immutable (cannot be changed after creation). Example:

```python
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
```

6. How do you import a module in Python?

Modules are imported using the import statement. Example:

```python
import math
print(math.sqrt(16)) # 4.0
```

7. What is the difference between `staticmethod` and `classmethod`?

- A `staticmethod` is a method that does not receive an implicit first argument. It functions like a regular method but exists within the class's namespace without access to the instance (`self`) or the class (`cls`).
- A `classmethod` takes the class (`cls`) as the first argument. It can access and modify class state, making it useful for factory methods or modifying class variables across all instances.

8. What is the function of the `pass` statement in Python?

The `pass` statement performs no action. It's typically used as a placeholder in code where a statement is syntactically required but no action needs to be performed, such as in empty function definitions or loops.

9. How are function arguments passed in Python (Pass by reference or pass by value)?

Python uses a model called pass-by-object-reference. This means that mutable objects like lists or dictionaries can be modified within a function, while immutable objects like integers, strings, or tup Python passes references to objects, but whether the object itself can be modified depe y.

10. What is a namespace in Python?

A namespace is a container that holds names of identifiers and ensures that they are unique within a certain scope. In Python, namespaces exist at different levels:

- **Local namespace:** Contains names defined within a function.
- **Global namespace:** Contains names defined at the module level.
- **Built-in namespace:** Contains Python's built-in functions and exceptions.

The scope determines the visibility of these namespaces and is defined in the LEGB (Local, Enclosing, Global, Built-in) rule.

11. Explain list comprehension and provide an example.

List comprehension provides a concise way to create lists.

Syntax: [expression for item in iterable if condition].

Example:

```
[x * 2 for x in range(5)] results in [0, 2, 4, 6, 8]
```

12. What is a lambda function?

A lambda function is a type of anonymous function created using the lambda keyword. It can accept multiple arguments but is limited to a single expression.

Example:

```
lambda x: x * 2
```

13. What is the purpose of the `__init__` method in classes?

`__init__` is the constructor method for initializing objects in a class. It sets the initial state of an object by assigning values to object properties.

14. Explain the difference between `append()` and `extend()` in a list.

The `append()` method adds one element to the end of a list, while the `extend()` method adds all elements from an iterable (such as another list) to the end of the list.

15. How do you handle exceptions in Python?

Exceptions in Python are handled using try, except, else, and finally blocks

Chat with us

Example:

```
try:
        # Code that may raise an exception
    except Exception as e:
        # Code to handle the exception
    else:
        # Code that runs if no exceptions occur
    finally:
        # Code that runs no matter what
```

16. What is the difference between GET and POST methods in HTTP?

- **GET:** Used to request data from a server. The data is appended to the URL and is visible, making it less secure for sensitive data.
- **POST:** Used to send data to the server to create or update resources. The data is sent in the request body, making it more secure than GET for transmitting sensitive information.

17. How do you manage states in a web application?

In web development, state management can be handled in various ways:

- **Client-Side:** Using cookies, local storage, and session storage to store the state data in the browser.
- **Server-Side:** Using server-side sessions (Django sessions) or databases to track the state across multiple client requests.

18. What is full-stack development?

Full-stack development refers to the development of both the front-end (client-side) and back-end (server-side) parts of a web application. A full-stack developer works with databases, servers, system engineering, and clients. In the context of Python, a full-stack developer would typically use frameworks like Django or Flask for the back-end and front-end technologies like HTML, CSS, JavaScript (React or Angular).

19. What front-end technologies are you familiar with, and how do they interact with Django?

Front-end technologies include HTML, CSS, JavaScript, and modern frameworks like React or Angular. These are used to create the user interface. They can interact with Django via APIs, typically using Django REST Framework to send and receive JSON data, or by rendering Django templates in the case of server-side rendering.

20. What is the virtual environment in Python, and why is it important?

A virtual environment is a tool to keep dependencies required by different projects in separate places, by creating isolated Python environments. It ensures that packages required for one projec
other projects. This is especially important for full-stack development where different
different versions of libraries or frameworks.

Python Interview Questions For Freshers

Here, are some Python interview questions asked for freshers during the interview:

1. What is Python?

Python is a widely used, high-level programming language known for its readability and simplicity. Python, developed by Guido van Rossum and initially released in 1991, supports various programming paradigms such as procedural, object-oriented, and functional programming.

2. What are the key features of Python?

Python is recognized for its ease of use, clarity, and wide range of available libraries. Key features include dynamic typing, interpreted nature, and a vast standard library.

3. What are the benefits of using Python language?

The benefits of using Python include:

- Easy to learn and read
- High-level language
- Large standard library
- Extensive third-party packages.
- Cross-platform
- Strong community support
- Good for prototyping
- Easily integrated with other languages.
- It is freely available for use and distribution, including for commercial applications.

4. What are some common Python libraries used in full-stack development?

For back-end development, Django and Flask are popular. For front-end development, libraries like jQuery or frameworks like React may be used, though these are not Python-based.

5. How will you check if a class is a child of another class?

In Python, you can check if a class is a subclass of another class using the issubclass() function. This method returns True if the first class is a subclass of the second class, else returns False.

Python Code:

```
class Parent:
    pass
class Child(Parent):
    pass
# Child is a subclass of Parent
```

Chat with us

```
print(issubclass(Child, Parent))
# Parent is a subclass of Child
print(issubclass(Parent, Child))
```

6. Why is ‘finalize’ used in Python?

In Python, the ‘finalize’ method is used in the context of resource management and garbage collection. It's part of the ‘weakref’ module, allowing objects to perform cleanup actions before they are destroyed by the garbage collector. It’s generally used to release unmanaged resources.

7. Are access specifiers used in Python?

Python does not use access specifiers like private, protected, and public. Instead, it uses naming conventions to indicate the intended visibility

8. What does the ‘#’ symbol do in Python?

In Python, the ‘#’ symbol is used to indicate a comment. Everything following the ‘#’ on that line is ignored by the Python interpreter. Comments are used to explain code and make it more understandable to humans.

9. What is the difference between a mutable and an immutable data type in Python?

- **Mutable Data Types:** These can be modified after they are created. Examples include lists ‘(list)’ and dictionaries ‘(dict)’.
- **Immutable Data Types:** These cannot be modified after their creation. Examples include strings ‘(str)’ and tuples ‘(tuple)’.

10. How are arguments passed in Python: by value or by reference?

In Python, all arguments are passed by reference, but it’s important to understand that Python variables hold references to objects. This means that while you cannot change the reference itself (i.e., the variable points to a different object), you can modify the object if it is mutable.

11. What is encapsulation?

Encapsulation is the concept of wrapping data (variables) and methods into a single unit (class) and restricting access to some of the object's components.

12. What is inheritance and its types?

Inheritance in object-oriented programming (OOP) allows classes to derive attributes and methods from other classes, facilitating code reusability and a hierarchical class structure. The types of inheritance in Python are:

- **Single Inheritance:** A class (subclass) inherits from one and only one parent class (Chat with us
- **Multiple Inheritance:** It occurs when a class inherits from more than one parent class. This allows a subclass to combine functionality from multiple superclasses.

- **Multilevel Inheritance:** A class derives from one class, which itself is derived from another class. This creates a chain of inheritance.
- **Hierarchical Inheritance:** It occurs when multiple subclasses inherit from a single parent class. This allows different subclasses to share the same parent class functionality.
- **Hybrid Inheritance:** It involves merging two or more forms of inheritance. It can involve multiple, multilevel, hierarchical, or any other forms of inheritance.

13. What is the difference between `__init__` and `__new__` methods in Python?

- `__init__` is the initializer (or constructor) method of a Python class, called when a new object is created. It initializes the attributes of the object after it has been created.
- `__new__` is responsible for creating a new instance of the class. It's called before `__init__`, and it determines the instance of `__init__` works on. `__new__` is rarely overridden unless works with immutable objects like tuples or strings.

14. Explain the use of the `with` statement in Python.

The `with` statement is used to wrap the execution of a block of code in methods defined by a context manager (using `__enter__` and `__exit__`). It simplifies resource management, such as opening and closing files, ensuring that resources are properly cleaned up after use (e.g., files are closed after reading or writing). The `with` statement eliminates the need for explicit cleanup code like `file.close()`.

15. What are Python's built-in data types?

Python's built-in data types include:

- **Numeric types:** int, float, complex.
- **Sequence types:** list, tuple, range, str.
- **Mapping type:** dict.
- **Set types:** set, frozen set.
- **Boolean type:** bool.
- **Binary types:** bytes, byte array, memory view.

software!

Talk to a career expert

Python Interview Questions For Experienced

Here are the Python interview questions for experienced professionals:

1. Write a program to generate Fibonacci numbers.

```
def fibonacci_generator():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

# Usage
fib_gen = fibonacci_generator()
for _ in range(10):
    print(next(fib_gen))
```

Output:

```
0
1
1
2
3
5
8
13
21
34
```

2. Difference Between Deep Copy and Shallow Copy

- **Shallow Copy:** A shallow copy generates a new object, but includes references to the objects contained in the original. If the original object contains other objects (like lists), the copied object will refer to the same internal objects. This can be done using the copy module's 'copy()' function.

```
import copy
original = [1, [2, 3]]
shallow = copy.copy(original)
```

Chat with us

- **Deep Copy:** A deep copy creates a new object and recursively copies all objects found in the original, so no references to the original objects are maintained. This can be done using the copy module's 'deepcopy()' function.

```
import copy
original = [1, [2, 3]]
deep = copy.deepcopy(original)
```

3. Write a Python function to merge two sorted lists.

```
def merge_sorted_lists(l1, l2):
    merged_list = []
    i = j = 0
    while i < len(l1) and j < len(l2):
        if l1[i] < l2[j]:
            merged_list.append(l1[i])
            i += 1
        else:
            merged_list.append(l2[j])
            j += 1
    merged_list.extend(l1[i:])
    merged_list.extend(l2[j:])
    return merged_list

# Example
l1 = [1, 3, 5]
l2 = [2, 4, 6]
print(merge_sorted_lists(l1, l2))  # Output: [1, 2, 3, 4, 5, 6]
```

4. Write a Python function to implement a binary search.

```
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# Example
arr = [1, 2, 3, 4, 5, 6]
target = 4
print(binary_search(arr, target))  # Output: 3
```

Chat with us

5. Write a Python function to check if the two strings are anagram

```
def are_anagrams(s1, s2):
    return sorted(s1) == sorted(s2)

# Example
print(are_anagrams("listen", "silent"))  # Output: True
print(are_anagrams("hello", "world"))    # Output: False
```

6. Write a Python function to find the largest element in an array.

```
def find_largest(arr):
    return max(arr)
# Example
arr = [3, 1, 4, 1, 5, 9, 2]
print(find_largest(arr))  # Output: 9
```

7. Write a Python function to perform a quicksort on the list.

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

# Example
arr = [3, 6, 8, 10, 1, 2, 1]
print(quicksort(arr))  # Output: [1, 1, 2, 3, 6, 8, 10]
```

8. Write a Python function to check if a given string is a palindrome.

```
def is_palindrome(s):
    return s == s[::-1]
# Example
print(is_palindrome('radar'))  # Output: True
print(is_palindrome('hello'))  # Output: False
```

9. What is the Global Interpreter Lock (GIL) in Python?

The Global Interpreter Lock (GIL) is a mutex in CPython that protects access to Python objects, preventing multiple native threads from executing Python bytecodes at once. It ensures that only or Python code at a time, even if run on multi-core processors. The GIL can be a problem programs but does not affect I/O-bound programs significantly.

Chat with us

10. What are decorators in Python, and how do they work?

Decorators are a way to modify or extend the behavior of functions or methods. They take a function as input, extend its behavior, and return a new function. In Python, decorators are applied using the `@decorator_name` syntax before the function definition. Commonly used for logging, access control, or memoization.

Conclusion

In conclusion, the journey of becoming a Python Full Stack Developer involves mastering both front-end and back-end technologies, with Python and Django forming the core of backend development. In interviews, candidates can expect questions that test their understanding of frameworks, databases, APIs, and integration techniques. For freshers, a strong grasp of foundational concepts and hands-on experience is crucial. Preparing well for Python Full Stack Developer interview questions will not only increase your confidence but also enhance your chances of securing your desired role.

Frequently Asked Questions

1. What are the key areas to focus on when preparing for Python Full Stack Developer interviews?

Focus on core Python concepts, Django, front-end technologies (HTML, CSS, JavaScript), databases (SQL and NoSQL), REST APIs, and deployment strategies.

2. What kind of Python Full Stack interview questions can I expect as a fresher?

Interview questions for freshers may include basic Python questions, database handlir Chat with us
integration.

Read More Articles