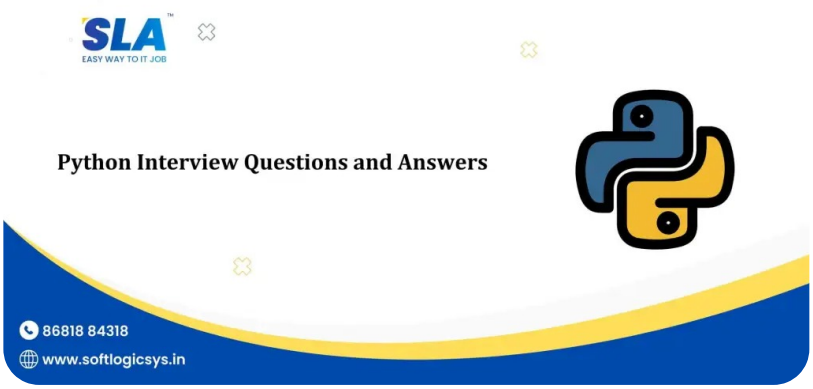

Share on your Social Media

Top 18 Python Interview Questions and Answers

Published On: April 16, 2019

Python Interview Questions and Answers

Python has numerous applications ranging from web development, data analysis, scientific computing to automation etc. This is the reason for Python's widespread popularity. Python simplicity and readability also adds to its already existing esteemed reputation. That's why Python is one of the most preferred programming languages. So, being skilled and employed in a Python-related job will surely lead to an extremely fulfilling job. So, start learning these Python Interview Questions and Answers to get the best chance at securing a Python job.

Download Python Interview Questions PDF

Featured Articles

Related Courses at SLA

- Python Training in Chennai
- Python Online Training

Related Posts

Top 12 Business Intelligence and Data Analytics Interview

Python Interview Questions and Answers

1. What is Python?

Python is a highly popular, user-friendly programming language known for its simplicity, versatility, and broad applicability across web development, data science, AI, and more. It has an extensive standard library, supports multiple programming paradigms, and benefits from a vibrant global community of developers.

2. What are the various programming paradigms supported by Python?

Python supports various programming paradigms, including object-oriented, procedural and functional programming.

3. Should Python be categorized as a compiled language or an interpreted language?

- Python is fundamentally an interpreted language where the Python interpreter executes code directly by reading and processing it line by line.
- However, Python also incorporates a compilation step where source code is automatically compiled into bytecode (.pyc files). This bytecode is then executed by the interpreter, blending the characteristics of both interpreted and compiled languages. This approach ensures Python remains versatile, enabling quick development and testing while leveraging bytecode for enhanced performance during execution.

4. Explain the meaning behind "#" symbol in Python.

n Python, the # symbol is utilized to indicate comments, which are lines of text ignored by the Python interpreter during program execution. These comments are intended solely for developers to

Questions and Answers
Published On: June 19, 2024

Business Intelligence and Data Analytics Interview Questions and Answers Our Business Intelligence and Data Analytics...

Top 20 Azure DevOps Interview Questions and Answers
Published On: June 19, 2024

Azure DevOps Interview Questions and Answers One of the most in-demand skills in the IT...

Top 20 AWS DevOps Interview Questions and Answers
Published On: June 19, 2024

AWS DevOps Interview Questions and Answers The integration of AWS and DevOps is doing magic...

Top 14 MEAN Stack Interview Questions and Answers
Published On: June 19, 2024

annotate and clarify code.

5. How does Python handle argument passing, either by value or by reference?

In Python, argument passing follows the mechanism of passing by object reference. This approach entails passing the reference to the object that a variable points to when it is passed to a function, rather than passing the object's value directly. As a result, functions have the capability to modify the contents of mutable objects that are passed as arguments.

6. Explain List Comprehension in Python.

List comprehension in Python offers a short method to generate lists by applying an expression to each element of an iterable (such as lists, tuples, or ranges) and optionally applying filter conditions.

7. Explain Lambda function in Python.

A lambda function in Python is a compact, anonymous function created using the lambda keyword. It serves the purpose in situations where defining a named function isn't necessary, providing a concise syntax for rapid function definition. Lambda functions excel at generating short, inline functions without the overhead of a fully named function. They are ideal for scenarios where the function's logic can be encapsulated within a single expression.

Download Python Syllabus PDF

8. Explain the concept of pass in Python.

In Python, the pass statement acts as a placeholder where syntactical correctness requires a statement, but no action is necessary or desired. It effectively serves as a no-operation command, enabling Python to smoothly bypass specific code blocks without triggering errors.

MEAN Stack Interview Questions and Answers Since MEAN Stack combines several other applications as part...

9. Explain swapcase function in Python

The swapcase() method in Python serves as a built-in string function designed specifically to invert the case of alphabetic characters within a given string. Here's an in-depth explanation of its functionality:

- **Definition:** swapcase() is utilized as a string method that generates a new string wherein all uppercase letters are converted to lowercase, and vice versa.
- **Syntax:** To apply swapcase(), it is called on a string object:

str.swapcase()

Here, str denotes the string on which the case transformation is to be performed.

- **Example:**

s = "Hello World"

swapped = s.swapcase()

print(swapped) # Output: hELLO wORLD

This example demonstrates that all uppercase letters ('H' and 'W') in the string s are converted to lowercase, while all lowercase letters ('e', 'l', 'o', ' ', 'o', 'r', 'l', 'd') are converted to uppercase.

- **Handling Non-alphabetic Characters:** Non-alphabetic characters such as digits or symbols remain unchanged:

s = "123 ABC!@#"

swapped = s.swapcase()

print(swapped) # Output: 123 abc!@#

In this case, the digits '123' and non-alphabetic characters (' ', '!', '@', '#') remain unaffected by swapcase().

- **Support for Unicode:** swapcase() fully

supports Unicode characters, ensuring accurate handling of cases for characters like accented letters or special symbols.

- **Use Cases:** This method proves particularly valuable for tasks requiring toggling the case of alphabetic characters in strings, such as conducting case-insensitive comparisons or transforming text to enhance readability.

10. Why does indentation matter in Python?

The following are the reasons as to why indentation matters in Python:

- **Syntax Requirement:** Python uses indentation instead of braces {} to delineate code blocks, requiring precise indentation (typically four spaces per level) for correct interpretation.
- **Enhanced Readability:** Indentation organizes code visually, aiding comprehension of statement flow within functions, loops, classes, and conditionals.
- **Promotes Consistency:** Consistent indentation fosters uniform coding styles across projects, facilitating collaboration and efficient code maintenance.
- **Logical Grouping:** Indentation logically groups statements within blocks like functions, loops, or conditionals.
- **Determines Block Scope:** Indentation directly determines the scope of code blocks, influencing program logic and preventing syntax errors.
- **Whitespace Sensitivity:** Python's sensitivity to whitespace encourages clean, organized code that is both visually and logically coherent.

11. Define scope in Python.

Scope in Python defines where variables can be

accessed in a program, influencing their visibility and lifespan. It's crucial for effective variable management and clear, error-free coding. Python uses nested scopes, and understanding scope rules is pivotal for writing well-structured programs.

12. Explain the types of scope in Python.

The following are the types of scope in Python:

- **Local Scope:**
- Variables within a function are locally scoped and accessible only within that function.
- Example:

```
def my_function():
    x = 10  # Local variable
    print(x)  # Accessible within my_function
my_function()
```

- **Enclosing (or non-local) Scope:**
- Applies to nested functions where variables not found locally are searched in enclosing scopes.
- Example

```
def outer_function():
    x = 10
    def inner_function():
        print(x)  # Accesses x from the enclosing scope
    inner_function()
outer_function()
```

- **Global Scope:**
- Variables defined at the top-level of a module are globally scoped, accessible throughout the module.
- Example

- **Built-in Scope:**

- Contains predefined names that are always accessible, forming the built-in scope.
- Example

- **Scope Resolution:**

- LEGB Rule: Defines the order in which Python searches for names:
- Local scope
- Enclosing (non-local) scope
- Global scope
- Built-in scope
- Python searches these scopes in order until it finds the first match or raises a NameError.

13. What is PEP 8?

PEP 8 serves as Python's style guide, providing recommendations and best practices for writing clean and maintainable code. It focuses on conventions for naming, organizing code, and offering general programming advice.

Python Developer Salary

14. What are decorators in Python?

Decorators represent a powerful feature in Python, enabling modification of the behavior of functions or classes. They are commonly used to augment existing code functionality without altering its core structure, employing the @decorator_name syntax for implementation.

15. How do you handle circular imports in Python?

Circular imports arise when multiple modules attempt to import each other directly or indirectly. Effective strategies for managing them include refactoring code to minimize dependencies or employing techniques like deferred importing within functions to mitigate circular import issue

16. How does memory management work in Python?

Python manages memory using a private heap, employing an integrated garbage collector for automatic memory management. Objects are allocated on this heap and are automatically deallocated when they are no longer referenced.

17. What are Python's generators?

Generators in Python are functions that utilize the yield keyword instead of return. They produce a sequence of values iteratively, one value at a time. Generators are particularly efficient with memory, making them suitable for handling large datasets in a memory-efficient manner.

18. What are some built-in data types in Python?

The following are some of the built-in data types in Python:

- **Integer (int):**
- Represents whole numbers, including negative and positive values.
- Example: x = 10
- **Floating-point (float):**
- Represents decimal numbers.
- Example: y = 3.14
- **Complex (complex):**
- Represents numbers in the form a + bi, where a and b are floats and i is the imaginary unit.
- Example: z = 2 + 3j
- **Boolean (bool):**
- Represents truth values True and False.
- Example: is_valid = True
- **String (str):**
- Represents sequences of characters that are

enclosed within single (') or double (") quotes.
- Example: name = 'Alice'

- **List:**

- Ordered collection of items, mutable (can be modified).
- Example: my_list = [1, 2, 3]

- **Tuple:**

- Ordered collection of items, immutable (cannot be modified).
- Example: my_tuple = (1, 2, 3)

- **Set:**

- Unordered collection of unique items.
- Example: my_set = {1, 2, 3}

- **Dictionary (dict):**

- Unordered collection of key-value pairs.
- Example: my_dict = {'name': 'Alice', 'age': 30}

- **NoneType (None):**

- Denotes a null value or an absence of a value.
- Example: x = None

Conclusion

These Python Interview Questions and Answers are the result of a lot of research and investigations into the types of questions that are frequently asked in Python interviews. So by familiarizing these Python Interview Questions students will get an overall knowledge on what kind of questions to expect from the Python interview, that will eventually make them prepared to face any kinds of Python questions easily. So, go ahead and make good use of these Python interview questions and answers to get placed in a fulfilling Python based job.

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Navigation

- About Us
- Blog Posts
- Careers
- Contact
- Placement Training
- Corporate Training
- Hire With Us
- Job Seekers
- SLA's Recently Placed Students
- Reviews
- Sitemap

Important Links

- Disclaimer
- Privacy Policy
- Terms and Conditions

Courses

- Python
- Software Testing
- Full Stack Developer
- Java

Social Media Links

Review Sources

- Google

Power BI

Clinical SAS

Data Science

Embedded

Cloud Computing

Hardware and Networking

VBA Macros

Mobile App Development

DevOps

Trustpilot

Glassdoor

Mouthshut

Sulekha

Justdial

Ambitionbox

Indeed

Software Suggest

Sitejabber