30 Python Coding Interview Questions for Beginners

Ayushi Trivedi
Last Updated : 01 May, 2025

116

Understanding Python coding interview questions is crucial as they serve as a gateway to opportunities in software development and data science careers. Mastering these questions not only showcases problem-solving abilities and Python proficiency but also enhances overall programming skills. By familiarizing oneself with common challenges and honing problem-solving strategies, candidates can confidently navigate technical interviews, demonstrating readiness for diverse roles in the tech industry. In this article we will explore Python coding interview questions for beginners which may help you in preparing for your interviews.

Q1. Write a Python program to Reverse a String?

Solution:

With Indexing:

Copy Code

```
def reverse_string(s):
    return s[::-1]
```

Output:

```
Original string: Hello, World!
Reversed string: !dlroW ,olleH
```

Without Indexing:

Copy Code

```
def reverse_string(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)
print("Original string:", input_string)
print("Reversed string:", reversed_string)
```

Output:

```
Original string: Hello, World!
Reversed string: !dlroW ,olleH
```

Q2. Write a Python program to Check Palindrome?

Solution:

For String:

Copy Code

```
def is_palindrome(s):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    s = s.replace(" ", "").lower()
    return s == s[::-1]
```

Output:

```
The string is not a palindrome.
```

For Number:

Copy Code

```
def is_palindrome(number):
    # Convert number to string for easy manipulation
    num_str = str(number)
    return num_str == num_str[::-1]

# Example usage
input_number = 12321
if is_palindrome(input_number):
    print("The number is a palindrome.")
else:
    print("The number is not a palindrome.")
```

Output:

```
The number is not a palindrome.
```

Q3. Write a Python program to Count Vowels in a String?

Solution:

Copy Code

```
def count_vowels(s):
    # Define vowels
    vowels = "aeiouAEIOU"
    # Initialize count
    count = 0
    # Count vowels
    for char in s:
        if char in vowels:
            count += 1
```

Output:

```
Number of vowels in the string:3
```

Q4. Write a Python program to find Factorial with Recursion?

Solution:

With Function:

Copy Code

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage
number = 5
result = factorial(number)
print("Factorial of", number, "is", result)
```

Output:

```
Factorial of 5 is 120
```

Without Function:

Copy Code

```python
number = 5
factorial = 1

if number < 0:
    print("Factorial is not defined for negative numbers.")
elif number == 0:
    print("Factorial of 0 is 1")
else:
```

```
Factorial of 5 is 120
```

Q5. Write a Python program to find Fibonacci Sequence?

Solution:

Copy Code

```
def fibonacci(n):
    fib_sequence = [0, 1]  # Initialize the sequence with the first two terms
    for i in range(2, n):
        next_term = fib_sequence[-1] + fib_sequence[-2]
        fib_sequence.append(next_term)
    return fib_sequence

# Example usage
num_terms = 10
fib_sequence = fibonacci(num_terms)
print("Fibonacci sequence up to", num_terms, "terms:", fib_sequence)
```

Output:

```
Fibonacci sequence up to 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Q6. Write a Python program to find Maximum Element in a List?

Solution:

Using Built-in Function:

Copy Code

```
# Example list
my_list = [10, 23, 45, 67, 12, 89, 34]

# Find maximum element
max_element = max(my_list)
```

Using User-defined Function:

Copy Code

```
def find_max_element(lst):
    if not lst:  # If the list is empty
        return None  # Return None since there is no maximum element
    max_element = lst[0]  # Initialize max_element with the first element of the
    for num in lst:
        if num > max_element:
            max_element = num
    return max_element

# Example usage
my_list = [10, 23, 45, 67, 12, 89, 34]
max_element = find_max_element(my_list)
print("Maximum element in the list:", max_element)
```

Output:

```
Maximum element in the list: 89
```

Q7. Write a Python program to find Anagram Check?

Solution:

Copy Code

```
def is_anagram(str1, str2):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()
    # Check if the sorted forms of both strings are equal
    return sorted(str1) == sorted(str2)

# Example usage
string1 = "listen"
string2 = "silent"
if is_anagram(string1, string2):
```

```
'listen' and 'silent' are anagrams.
```

Q8. Write a Python program to find Prime Numbers?

Solution:

Copy Code

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

def find_primes(start, end):
    primes = []
    for num in range(start, end + 1):
        if is_prime(num):
            primes.append(num)
    return primes

# Example usage
start_range = 1
end_range = 50
prime_numbers = find_primes(start_range, end_range)
print("Prime numbers between", start_range, "and", end_range, "are:", prime_numbe
```

Output:

```
Prime numbers between 1 and 50 are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
 37, 41, 43, 47]
```

Q9. Write a Python program to check for Pangram?

```
    sentence = sentence.lower()
    # Create a set of unique characters in the sentence
    unique_chars = set(sentence)
    # Remove non-alphabetic characters and spaces
    unique_chars.discard(" ")
    unique_chars.difference_update(set(string.punctuation))
    # Check if all letters of the alphabet are present
    return len(unique_chars) == 26

# Example usage
input_sentence = "The quick brown fox jumps over the lazy dog"
if is_pangram(input_sentence):
    print("The sentence is a pangram.")
else:
    print("The sentence is not a pangram.")
```

Output:

```
The sentence is a pangram.
```

Q10. Write a Python program to basic Data Structure Operations (e.g., list manipulation, string manipulation)?

Solution:

Copy Code

```
# List manipulation
my_list = [1, 2, 3, 4, 5]

# Append an element to the list
my_list.append(6)
print("After appending 6:", my_list)

# Remove an element from the list
my_list.remove(3)
print("After removing 3:", my_list)

# Access elements by index
```

```
words = my_string.split()
print("Split string into words:", words)

# Join elements of a list into a single string
new_string = "-".join(words)
print("Joined words with '-':", new_string)

# Convert string to uppercase
upper_string = my_string.upper()
print("Uppercase string:", upper_string)

# Replace a substring
replaced_string = my_string.replace("World", "Universe")
print("After replacing 'World' with 'Universe':", replaced_string)
```

Output:

```
After appending 6: [1, 2, 3, 4, 5, 6]
After removing 3: [1, 2, 4, 5, 6]
Element at index 2: 4
Split string into words: ['Hello,', 'World!']
Joined words with '-': Hello,-World!
Uppercase string: HELLO, WORLD!
After replacing 'World' with 'Universe': Hello, Universe!
```

Q11. Write a Python program to find Minimum Element in a List?

Solution:

Using User-defined:

Copy Code

```
def find_min_element(lst):
    if not lst:  # If the list is empty
        return None  # Return None since there is no minimum element
    min_element = lst[0]  # Initialize min_element with the first element of the
    for num in lst:
```

```
min_element = find_min_element(my_list)
print("Minimum element in the list:", min_element)
```

Output:

```
Minimum element in the list: 10
```

Using Built-in Function:

Copy Code

```
my_list = [10, 23, 45, 67, 12, 89, 34]
min_element = min(my_list)
print("Minimum element in the list:", min_element)
```

Output:

```
Minimum element in the list: 10
```

Q12. Write a Python program to calculate Sum of Digits in a Number?

Solution:

Copy Code

```
def sum_of_digits(number):
    # Convert number to string to iterate through its digits
    num_str = str(number)
    # Initialize sum
    digit_sum = 0
    # Iterate through each digit and add it to the sum
    for digit in num_str:
        digit_sum += int(digit)
    return digit_sum

# Example usage
input_number = 12345
result = sum_of_digits(input_number)
```

Q13. Write a Python program to check for Armstrong Number?

Solution:

Copy Code

```
def is_armstrong(number):
    # Convert number to string to get its length
    num_str = str(number)
    # Get the number of digits
    num_digits = len(num_str)
    # Initialize sum
    armstrong_sum = 0
    # Calculate the sum of digits raised to the power of the number of digits
    for digit in num_str:
        armstrong_sum += int(digit) ** num_digits
    # Check if the sum is equal to the original number
    return armstrong_sum == number

# Example usage
input_number = 153
if is_armstrong(input_number):
    print(input_number, "is an Armstrong number.")
else:
    print(input_number, "is not an Armstrong number.")
```

Output:

```
153 is an Armstrong number.
```

Q14. Write a Python program to check for Leap Year?

Solution:

Copy Code

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    print(input_year, "is a leap year.")
else:
    print(input_year, "is not a leap year.")
```

Output:

```
2024 is a leap year.
```

Q15. Write a Python program to calculate Factorial without Recursion?

Solution:

Copy Code

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

# Example usage
number = 5
result = factorial(number)
print("Factorial of", number, "is", result)
```

Output:

```
Factorial of 5 is 120
```

Q16. Write a Python program to find Average of Numbers in a List?

Solution:

Copy Code

```
def find_average(numbers):
    if not numbers:  # If the list is empty
        return None  # Return None since there are no numbers to average
average = find_average(number_list)
if average is not None:
    print("Average of numbers in the list:", average)
else:
    print("The list is empty.")
```

Output:

```
Average of numbers in the list: 30.0
```

Q17. Write a Python program to Merge Two Sorted Lists?

Solution:

Copy Code

```
def merge_sorted_lists(list1, list2):
    merged_list = []
    i = j = 0

    while i < len(list1) and j < len(list2):
        if list1[i] < list2[j]:
            merged_list.append(list1[i])
            i += 1
        else:
            merged_list.append(list2[j])
            j += 1

    # Append remaining elements from list1, if any
    while i < len(list1):
        merged_list.append(list1[i])
        i += 1

    # Append remaining elements from list2, if any
    while j < len(list2):
        merged_list.append(list2[j])
        j += 1

    return merged_list
```

Output:

```
Merged sorted list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Q18. Write a Python program to Remove Duplicates from a String?

Solution:

Copy Code

```
def remove_duplicates(input_string):
    # Initialize an empty set to store unique characters
    unique_chars = set()
    # Initialize an empty string to store the result
    result = ""
    # Iterate through each character in the input string
    for char in input_string:
        # Add the character to the result string if it's not already in the set
        if char not in unique_chars:
            result += char
            unique_chars.add(char)
    return result

# Example usage
input_string = "hello world"
result = remove_duplicates(input_string)
print("String with duplicates removed:", result)
```

Output:

```
String with duplicates removed: helo wrd
```

Q19. Write a Python program to Check for Perfect Number?

Solution:

```
        if number % i == 0:
            divisor_sum += i
    # Check if the sum of proper divisors equals the number
    return divisor_sum == number

# Example usage
input_number = 28
if is_perfect_number(input_number):
    print(input_number, "is a perfect number.")
else:
    print(input_number, "is not a perfect number.")
```

Output:

```
28 is a perfect number.
```

Q20. Write a Python program to Find Maximum Difference between Two Elements in a List?

Solution:

Copy Code

```
def max_difference(nums):
    if len(nums) < 2:
        return None  # If the list has less than two elements, return None
    min_element = float('inf')  # Initialize min_element to positive infinity
    max_difference = float('-inf')  # Initialize max_difference to negative infin
    for num in nums:
        min_element = min(min_element, num)
        max_difference = max(max_difference, num - min_element)
    return max_difference

# Example usage
numbers = [7, 1, 5, 3, 6, 4]
result = max_difference(numbers)
if result is not None:
    print("Maximum difference between two elements in the list:", result)
else:
```

Q21. Write a Python program to check if a Number is Even or Odd?

Solution:

With User-defined Function:

Copy Code

```
def check_even_odd(number):
    if number % 2 == 0:
        return "Even"
    else:
        return "Odd"

# Example usage
input_number = 7
result = check_even_odd(input_number)
print(input_number, "is", result)
```

Output:

```
7 is Odd
```

Without Function:

Copy Code

```
number = 7

if number % 2 == 0:
    print(number, "is Even")
else:
    print(number, "is Odd")
```

Output:

```
7 is Odd
```

Q22. Write a Python program to Count Words in a Sentence?

Copy Code

```
def count_words(sentence):
    # Split the sentence into words using whitespace as the delimiter
    words = sentence.split()
    # Count the number of words
    return len(words)

# Example usage
input_sentence = "This is a sample sentence."
word_count = count_words(input_sentence)
print("Number of words in the sentence:", word_count)
```

Output:

```
Number of words in the sentence: 5
```

With Built-in Fucntion:

Copy Code

```
sentence = "This is a sample sentence."
word_count = len(sentence.split())
print("Number of words in the sentence:", word_count)
```

Output:

```
Number of words in the sentence: 5
```

Without Built-in Function:

Copy Code

```
sentence = "This is a sample sentence."
word_count = 0
# Flag to indicate if the current character is part of a word
in_word = False
# Iterate through each character in the sentence
for char in sentence:
    # If the character is not a space and we are not already in a word
    if char != ' ' and not in_word:
        # Increment word count and set the flag to indicate we are in a word
```

```
print("Number of words in the sentence:", word_count)
```

Output:

```
Number of words in the sentence: 5
```

Q24. Write a Python program to Convert Decimal to Binary?

Solution:

Copy Code

```
def decimal_to_binary(decimal):
    binary = ""
    quotient = decimal
    while quotient > 0:
        remainder = quotient % 2
        binary = str(remainder) + binary
        quotient //= 2
    return binary

# Example usage
decimal_number = 10
binary_number = decimal_to_binary(decimal_number)
print("Binary representation of", decimal_number, "is", binary_number)
```

Output:

```
Binary representation of 10 is 1010
```

Q25. Write a Python program to Find Second Largest Element in a List?

Solution:

Copy Code

```
def second_largest(nums):
```

```
numbers = [10, 30, 20, 40, 50]
result = second_largest(numbers)
if result is not None:
    print("Second largest element in the list:", result)
else:
    print("The list has less than two elements.")
```

Output:

```
Second largest element in the list: 40
```

Q26. Write a Python program to Reverse Words in a String?

Solution:

Copy Code

```
def reverse_words(input_string):
    # Split the string into words
    words = input_string.split()
    # Reverse the order of words
    reversed_words = words[::-1]
    # Join the reversed words back into a string
    reversed_string = " ".join(reversed_words)
    return reversed_string

# Example usage
input_string = "Hello World"
reversed_string = reverse_words(input_string)
print("Original string:", input_string)
print("Reversed string:", reversed_string)
```

Output:

```
Original string: Hello World
Reversed string: World Hello
```

Copy Code

```
def is_prime_factor(number, potential_factor):
    if number <= 1 or potential_factor <= 1:
        return False  # Numbers less than or equal to 1 are not considered prime
    return number % potential_factor == 0

# Example usage
number = 15
potential_factor = 3
if is_prime_factor(number, potential_factor):
    print(potential_factor, "is a prime factor of", number)
else:
    print(potential_factor, "is not a prime factor of", number)
```

Output:

```
3 is a prime factor of 15
```

Q28. Write a Python program to check if a Number is a Power of Two?

Solution:

Copy Code

```
def is_power_of_two(number):
    if number <= 0:
        return False  # Numbers less than or equal to 0 are not powers of two
    while number > 1:
        if number % 2 != 0:
            return False  # If the number is not divisible by 2, it's not a power
        number //= 2
    return True

# Example usage
number = 16
if is_power_of_two(number):
    print(number, "is a power of two.")
else:
```

Q29. Write a Python program to convert Celsius to Fahrenheit?

Solution:

Copy Code

```
def celsius_to_fahrenheit(celsius):
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

# Example usage
celsius_temperature = 25
fahrenheit_temperature = celsius_to_fahrenheit(celsius_temperature)
print("Celsius:", celsius_temperature, "Fahrenheit:", fahrenheit_temperature)
```

Output:

```
Celsius: 25 Fahrenheit: 77.0
```

Q30. Write a Python program to calculate LCM (Least Common Multiple) of Two Numbers?

Solution:

Copy Code

```
import math

def lcm(a, b):
    return abs(a * b) // math.gcd(a, b)

# Example usage
num1 = 12
num2 = 18
result = lcm(num1, num2)
print("LCM of", num1, "and", num2, "is", result)
```

Output:

Conclusion

Completing practice with Python coding interview questions is essential for those who want to succeed in data science and software development positions. This extensive compilation addresses a broad range of basic ideas, from arithmetic operations and list manipulation to string manipulation. There are thorough answers for every query, complete with concise justifications and useful code samples. Candidates who actively engage with these questions not only demonstrate their mastery of Python but also develop critical thinking abilities that are necessary for acing technical interviews and landing a variety of jobs in the tech sector.

Also Read: Top 12 Free Python Courses

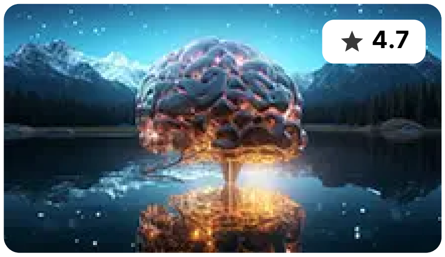
Here are a few articles that might be useful for your interview:

- How to Prepare for an AI Job Interview?
- 5 Strategies for Acing Your AI Technical Interview
- Top 10 Telephonic Interview Tips
- How to Introduce Yourself in a Job Interview?
- Tips and Tricks to Crack Campus Placement in Data Science
- Top 21 Python Interview Questions & Answers
- Python Coding Interview Questions & Answers – Part 1

My name is Ayushi Trivedi. I am a B. Tech graduate. I have 3 years of experience working as an educator and content editor. I have worked with various python libraries, like numpy, pandas, seaborn, matplotlib, scikit, imblearn, linear regression and many more. I am also an author. My first book named #turning25 has been published and is available on amazon and flipkart. Here, I am technical content editor at Analytics Vidhya. I feel proud and happy to be AVian. I have a great team to work with. I love building the bridge between the technology and the learner.

Beginner Interview Prep Interviews Python Python

Free Courses



Generative AI - A Way of Life

Explore Generative AI for beginners: create text and images, use top AI tools, learn practical skills, and ethics.

Master Large Language Models (LLMs) with this course, offering clear guidance in NLP and model training made simple.

Building LLM Applications using Prompt Engineering

This free course guides you on building LLM apps, mastering prompt engineering, and developing chatbots with enterprise data.

Improving Real World RAG Systems: Key Challenges & Practical Solutions

Explore practical solutions, advanced retrieval strategies, and agentic RAG systems to improve context, relevance, and accuracy in AI-driven applications.

Microsoft Excel: Formulas & Functions

Master MS Excel for data analysis with key formulas, functions, and LookUp tools in this comprehensive course.

Check if Element Exists in List in Python

30+ MCQs on Python List Manipulation

30+ MCQs on Python Functions

30+ MCQs on Python Recursion

30+ MCQs on Python Control Flow(If Statements a...

Prime Number Program in Python

40 Helpful Python tips to Boost your speed as a...

15 Python Built-in Functions fo Data Science

40 Data Science Coding Questions and Answers fo...

What are your thoughts?...

Submit reply

Flagship Programs

GenAI Pinnacle Program | GenAI Pinnacle Plus Program | AI/ML BlackBelt Program | Agentic AI Pioneer Program

Free Courses

Generative AI | DeepSeek | OpenAI Agent SDK | LLM Applications using Prompt Engineering | DeepSeek from Scratch | Stability.AI | SSM & MAMBA | RAG Systems using LlamaIndex | Building LLMs for Code | Python | Microsoft Excel | Machine Learning | Deep Learning | Mastering Multimodal RAG | Introduction to Transformer Model | Bagging & Boosting | Loan Prediction | Time Series Forecasting | Tableau | Business Analytics | Vibe Coding in Windsurf | Model Deployment using FastAPI | Building Data Analyst AI Agent | Getting started with OpenAI o3-mini | Introduction to Transformers and Attention Mechanisms

Popular Categories

AI Agents | Generative AI | Prompt Engineering | Generative AI Application | News | Technical Guides | AI Tools | Interview Preparation | Research Papers | Success Stories | Quiz | Use Cases | Listicles

Generative AI Tools and Techniques

GANs | VAEs | Transformers | StyleGAN | Pix2Pix | Autoencoders | GPT | BERT | Word2Vec | LSTM | Attention Mechanisms | Diffusion Models | LLMs | SLMs | Encoder Decoder Models | Prompt Engineering | LangChain | LlamaIndex | RAG | Fine-tuning | LangChain AI Agent | Multimodal Models | RNNs | DCGAN | ProGAN | Text-to-Image Models | DDPM | Document Question Answering | Imagen | T5 (Text-to-Text Transfer Transformer) | Seq2seq Models | WaveNet | Attention Is All You Need (Transformer Architecture) | WindSurf | Cursor

Claude 3.5 Sonnet | Phi 4 | Phi 3.5 | Mistral Small 3.1 | Mistral NeMo | Mistral-7b | Bedrock | Vertex AI | Qwen QwQ 32B | Qwen 2 | Qwen 2.5 VL | Qwen Chat | Grok 3

AI Development Frameworks

n8n | LangChain | Agent SDK | A2A by Google | SmolAgents | LangGraph | CrewAI | Agno | LangFlow | AutoGen | LlamaIndex | Swarm | AutoGPT

Data Science Tools and Techniques

Python | R | SQL | Jupyter Notebooks | TensorFlow | Scikit-learn | PyTorch | Tableau | Apache Spark | Matplotlib | Seaborn | Pandas | Hadoop | Docker | Git | Keras | Apache Kafka | AWS | NLP | Random Forest | Computer Vision | Data Visualization | Data Exploration | Big Data | Common Machine Learning Algorithms | Machine Learning | Google Data Science Agent

Become an Author

Become a Speaker

Become a Mentor

Become an Instructor

Our Offerings

Trainings

Data Culture

AI Newsletter

Terms & conditions • Refund Policy • Privacy Policy • Cookies Policy