90+ Python Interview Questions and Answers (2025 Edition)

BEGINNER BEST OF TECH INTERVIEW PREP INTERVIEWS PYTHON PYTHON

Introduction

Welcome to the first step of preparing for your Data Science job interview. Here's a comprehensive and extensive list of Python interview questions and answers to help you ace that interview and get your dream job! Python is an interpreted and general-purpose **programming language** in high demand nowadays due to its usage in Artificial Intelligence(AI). Therefore, it becomes indispensable for every Data Science aspirant to have a strong understanding of functions used in Python. As you delve into mastering Python for your upcoming interview, these Python interview questions will surely guide you towards success.

In this article, you will get insights into essential Python coding interview questions and effective strategies to tackle common Python coding questions. We'll also provide valuable Python interview questions and answers to help you prepare thoroughly for your next interview.

Python Interview Questions and python coding interview questions and answers for Freshers

Q1. Convert a given string to int using a single line of code.

Ans. We can convert a given string to an integer using a built-in function *int()*. e.g.-

```
a = '5' print(int(a))
```

Variable ‘a’ is a string that is now converted to an integer, as shown **below**:

Output:

5

Q2. Write a code snippet to convert a string to a list.

Ans. Below is the code to convert a string to a list in Python.

```
str1 = "Analytics Vidhya" print(str1.split(" "))
```

The split() function separates the given string by the defined delimiter, i.e., space(” “) here. Therefore, *Analytics* and *Vidhya* break down into two strings in a list.

Output:

[‘Analytics’, ‘Vidhya’]

Q3. Write a code snippet to reverse a string.

Ans. Here, we have reversed a string without using any in-built function.

```
str1 = "Analytics Vidhya" str2 = "" for i in str1: str2 = i + str2 print("The original string is: ", str1) print("The reversed string is: ", str2)
```

The above code picks the first letter, i.e., ‘A’, then adds ‘n’ at the beginning.
Further, ‘nA’ is taken as str2, ‘ a’ is added before it, and so on.
Then ‘anA’ becomes str2, and the next letter, i.e., ‘l’, is appended at the start of str2 to make it ‘lanA.’
This is how the above code works to reverse the string.

Output:

ayhdiV scitylanA

Q4. Write a code snippet to sort a list in Python.

Ans. We can sort a list in Python using the sort() function. The following code will illustrate this:

```
my_list = [3, 2, 1] my_list.sort() print(my_list)
```

The above code sort the list using the sort() function.

Output:

[1, 2, 3]

Q5. What is the difference between mutable and immutable?

Ans. Mutable objects: They can be updated once defined. e.g., list.
Immutable objects: They cannot be updated. e.g., tuples.

Q6. How can you delete a file in Python?

Ans. We can delete the file in Python using the os module. The remove() function of the os module is used to delete a file in Python by passing the filename as a parameter. e.g.

```
import os os.remove("txt1.txt")
```

Q7. How to access an element of a list?

Ans. The element in a list can be accessed using list_name [index]. For instance:

Given a list [1, 2, 3, 4].

The indexing of the list starts from 0.

The first element of the list can be accessed using list[0], which will print element “1”.

The second element can be accessed using list[1] and so on.

Q8. Discuss different ways of deleting an element from a list.

Ans. There are two ways in which we can delete elements from the list:

1. By using the remove() function

The remove () function deletes the mentioned element from the list.

```
list1 = [1, 2, 3, 4] list1.remove(2) print(list1)
```

Output:

[1, 3, 4]

2. By using the pop() function

Pop() function delete element mentioned at a specific index from the list

```
list1.pop(1) print(list1)
```

Output:

[1, 4]

Q9. Write a code snippet to delete an entire list.

Ans. We can delete a list in Python using the clear() function. The following code will illustrate this:

```
list1 = [1, 2, 3, 4] list1.clear()
```

It will delete the entire list.

Q10. Write a code snippet to reverse an array.

Ans. The two ways of reversing an array are as follows:

1. Using the flip() function

```
import numpy as np arr1 = np.array([1, 2, 3, 4]) arr2 = np.flip(arr1) print(arr2)
```

Output:

[4, 3, 2, 1]

2. Without using any function

```
import numpy as np arr1 = np.array([1, 2, 3, 4]) arr2 = arr1[::-1] print(arr2)
```

Output:

[4,3,2,1]

Q11. Write a code snippet to get an element, delete an element, and update an element in an array.

Ans. **Access:** We can access an element of an array by using array_name[index].

```
print(arr[index])
```

Delete: We can delete the element of an array using the delete() function.

```
import numpy as np arr2 = [1, 2, 3, 4] x = np.delete(arr2, 0) print(x)
```

Output:

[2,3,4]

Update: We can update the element of an array using the below syntax:

```
array_name[index] = element
```

Q12. Write a code snippet to concatenate lists.

Ans. Suppose, given two lists are:

```
List1= [“W”, “a”, “w”,”b”]
List2 = [“e”, “ “,”riting”,”log”]
```

And the output should be:

[‘We’, ‘a ‘, ‘writing’, ‘blog’]

This can concatenate the two lists using the zip() function, which iterates through both lists and combines them index-wise.

```
lst1 = ['W', 'a', 'w', 'b'] lst2 = ['e', ' ', 'riting', 'log'] lst3 = [x + y for x, y in zip(lst1, lst2)] print(lst3)
```

Output:

[‘We’, ‘a ‘, ‘writing’, ‘blog’]

Q13. Write a code snippet to generate the square of every element of a list.

Ans. First, create an empty list. We used a for loop to iterate through every element of a list and multiply the element by itself to generate a square of it. Then, append it to the newly generated list.

```
lst = [1, 2, 3, 4] lst_final = [] for x in lst: lst_final.append(x * x) print(lst_final)
```

The for loop takes the first element, i.e., 1, multiply it with itself, and then appends it to the list. It then takes the second element, i.e., 2, multiplies it by itself, appends it to the list, and so on.

Input: [1, 2, 3, 4]

Output: [1, 4, 9, 16]

Q14. What is the difference between range and xrange?

Ans. In Python 2, range() returns a list, and xrange() returns an iterator. But in Python 3 xrange() no longer exists while range() returns an iterator.

Q15. What is pickling and unpickling in Python?

Ans. Pickling is converting a Python object (list, dict, function, etc.) to a byte stream(0s and 1s), and unpickling is converting the byte stream back to a python object. It is used to transfer and store various Python objects. We can use pickle or dill Python packages for this.

Q16. What is init in Python?

Ans. **__init__** method is used in Python to initialize the attributes of the object when the object is created. So, it is similar to the constructor in Java or C++. It is declared within the class as a reserved method.

Q17. What is the PEP-8 Style Guide?

Ans. It is the recommended coding conventions guide to follow for easier readability of the code. Since Multiple people work on the same project, it is preferable to follow a similar style for better readability and consistency. However, we can use our judgment about what style to follow, If there is any need to deviate from conventions.

Q18. Which is faster, Python list or Numpy arrays, and why?

Ans. NumPy arrays are notably faster than Python lists for numerical operations. NumPy is an open-source library designed for efficient array operations in Python, leveraging optimized implementations in C. Unlike Python lists, which are interpreted, NumPy arrays are executed in a compiled language, enhancing their performance significantly.

Python also includes a built-in `array` module for basic operations, which can be imported using `import array as arr`.

Q19. What is the difference between a Python list and a tuple?

Ans. In Python, a list is an ordered collection of objects that can be of different types. Lists are mutable, which means that you can change the value of a list element or add or remove elements from a list. Lists are created using square brackets and a comma-separated list of values.

A tuple is also an ordered collection of objects, but it is immutable, which means that you cannot change the value of a tuple element or add or remove elements from a tuple.

Lists are defined using square brackets ([‘’]), while tuples are defined using parentheses ((‘’,)).

Lists have several built-in methods for adding, removing, and manipulating elements, while tuples do not have these methods.

In general, tuples are faster than lists in Python.

Q20. What are Python sets? Explain some of the properties of sets.

Ans. In Python, a set is an unordered collection of unique objects. Sets are often used to store a collection of distinct objects and to perform membership tests (i.e., to check if an object is in the set). Sets are defined using curly braces ({ and }) and a comma-separated list of values.

Here are some key properties of sets in Python:

- Sets are unordered: Sets do not have a specific order, so you cannot index or slice them like you can with lists or tuples.
- Sets are unique: Sets only allow unique objects, so if you try to add a duplicate object to a set, it will not be added.
- Sets are mutable: You can add or remove elements from a set using the add and remove methods.
- Sets are not indexed: Sets do not support indexing or slicing, so you cannot access individual elements of a set using an index.

- Sets are not hashable: Sets are mutable, so they cannot be used as keys in dictionaries or as elements in other sets. If you need to use a mutable object as a key or an element in a set, you can use a tuple or a frozen set (an immutable version of a set).

Q21. What is the difference between split and join?

Ans. Split and join are both functions of Python strings, but they are completely different when it comes to functioning.

The split function is used to create a list from strings based on some delimiter, for e.g., space.

```
a = 'This is a string' li = a.split(' ') print(li)
```

Output: ['This', 'is', 'a', 'string']

The join() method is a built-in function of Python's str class that concatenates a list of strings into a single string. It is called on a delimiter string and invoked with a list of strings to be joined. The delimiter string is inserted between each string in the list when the strings are concatenated.

Here is an example of how to use the join() method:

```
“ “.join(li)
```

Output: This is a string

Here the list is joined with a space in between.

Q22. Explain the logical operations in Python.

Ans. In Python, the logical operations and, or, and not can be used to perform boolean operations on truth values (True and False).

The **and** operator returns True if both the operands are True, and False otherwise.

The **or** operator returns True if either of the operands is True, and False if both operands are False.

The **not** operator inverts the boolean value of its operand. If the operand is True, **not** return False, and if the operand is False, **not** return True.

Q23. Explain the top 5 functions used for Python strings.

Ans. Here are the top 5 Python string functions:

- **len():** This function returns the length of a string.

```
s = 'Hello, World!' print(len(s))
```

- **strip():** This function removes leading and trailing whitespace from a string.

```
s = ' Hello, World! ' print(s.strip())
```

```
'Hello, World!'
```

- **replace():** This function replaces all occurrences of a specified string with another string.

```
s = 'Hello, World!' print(s.replace('World', 'Universe'))
```

```
'Hello, Universe!'
```

- **split():** This function splits a string into a list of substrings based on a delimiter.

```
s = 'Hello, World!' print(s.split(','))
```

```
['Hello', ' World!']
```

- **upper() and lower():** These functions convert a string to uppercase or lowercase, respectively.

```
s = 'Hello, World!' print(s.upper())
```

```
'HELLO, WORLD!'
```

```
s.lower()
```

```
'hello, world!'
```

In addition to them, string also has capitalize, isalnum, isalpha, and other methods.

Q24. What is the use of the pass keyword in Python?

Ans. Pass is a null statement that does nothing. It is often used as a placeholder where a statement is required syntactically, but no action needs to be taken. For example, if you want to define a function or a class but haven't yet decided what it should do, you can use the pass as a placeholder.

Q25. What is the use of the continue keyword in Python?

Ans. Continue is used in a loop to skip over the current iteration and move on to the next one. When continue is encountered, the current iteration of the loop is terminated, and the next one begins.

Q26. What are immutable and mutable data types?

Ans. In Python, an immutable object is an object whose state cannot be modified after it is created. This means that you can't change the value of an immutable object once it is created. Examples of immutable objects in Python include numbers (such as integers, floats, and complex numbers), strings, and tuples.

On the other hand, a mutable object is an object whose state can be modified after it is created. This means that you can change the value of a mutable object after it is created. Examples of mutable objects in Python include lists and dictionaries.

Understanding the difference between immutable and mutable objects in Python is important because it can affect how you use and manipulate data in your code. For example, if you have a list of numbers and you want to sort the list in ascending order, you can use the built-in sort() method to do this. However, if you have a tuple of numbers, you can't use the sort() method because tuples are immutable. Instead, you would have to create a new sorted tuple from the original tuple.

Q27. What is the use of try and except block in Python?

Ans. The try and except blocks in Python are used to handle exceptions. An exception is an error that occurs during the execution of a program.

The try block contains code that might cause an exception to be raised. The except block contains code that is executed if an exception is raised during the execution of the try block.

Using a try-except block will save the code from an error to occur and can be executed with a message or output we want in the except block.

Q28. Name 2 mutable and 2 immutable data types in Python.

Ans. 2 **mutable** data types are **Dictionary** and **List**. You can change/edit the values in a Python dictionary and a list. It is not necessary to make a new list which means that it satisfies the property of mutability.

2 **immutable** data types are **Tuples** and **String**. You cannot edit a string or a value in a tuple once it is created. You need to either assign the values to the tuple or make a new tuple.

Q29. What are Python functions, and how do they help in code optimization?

Ans. In Python, a function is a block of code that can be called by other parts of your program. Functions are useful because they allow you to reuse code and divide your code into logical blocks that can be tested and maintained separately.

To call a function in Python, you simply use the function name followed by a pair of parentheses and any necessary arguments. The function may or may not return a value that depends on the usage of the turn statement.

Functions can also help in code optimization:

- Code reuse: Functions allow you to reuse code by encapsulating it in a single place and calling it multiple times from different parts of your program. This can help to reduce redundancy and make your code more concise and easier to maintain.

- Improved readability: By dividing your code into logical blocks, functions can make your code more readable and easier to understand. This can make it easier to identify bugs and make changes to your code.
- Easier testing: Functions allow you to test individual blocks of code separately, which can make it easier to find and fix bugs.
- Improved performance: Functions can also help to improve the performance of your code by allowing you to use optimized code libraries or by allowing the Python interpreter to optimize the code more effectively.

Q30. Why does NumPy have huge popularity in the field of data science?

Ans. NumPy (short for Numerical Python) is a popular library for scientific computing in Python. It has gained a lot of popularity in the data science community because it provides fast and efficient tools for working with large arrays and matrices of numeric data.

NumPy provides fast and efficient operations on arrays and matrices of numerical data. It uses optimized C and Fortran code behind the scenes to perform these operations, which makes them much faster than equivalent operations using Python's built-in data structures. It provides fast and efficient tools for working with large arrays and matrices of numeric data.

NumPy provides a large number of functions for performing mathematical and statistical operations on arrays and matrices.

It allows you to work with large amounts of data efficiently. It provides tools for handling large datasets that would not fit in memory, such as functions for reading and writing data to disk and for loading only a portion of a dataset into memory at a time.

NumPy integrates well with other scientific computing libraries in Python, such as SciPy (Scientific Python) and pandas. This makes it easy to use NumPy with other Python libraries to perform more complex data science tasks.

Q31. Explain list comprehension and dict comprehension.

Ans. List comprehension and dict comprehension are both concise ways to create new lists or dictionaries from existing iterables.

List comprehension is a concise way to create a list. It consists of square brackets containing an expression followed by a for clause, then zero or more for or if clauses. The result is a new list that evaluates the expression in the context of the for and if clauses.

Dict comprehension is a concise way to create a dictionary. It consists of curly braces containing a key-value pair, followed by a for clause, then zero or more for or if clauses. A result is a new dictionary that evaluates the key-value pair in the context of the for and if clauses.

Q32. What are global and local variables in Python?

Ans. In Python, a variable that is defined outside of any function or class is a global variable, while a variable that is defined inside a function or class is a local variable.

A global variable can be accessed from anywhere in the program, including inside functions and classes. However, a local variable can only be accessed within the function or class in which it is defined.

It is important to note that you can use the same name for a global variable and a local variable, but the local variable will take precedence over the global variable within the function or class in which it is defined.

```
# This is a global variable x = 10 def func(): # This is a local variable x = 5 print(x) func() print(x)
```

Output: This will print 5 and then 10

In the example above, the x variable inside the func() function is a local variable, so it takes precedence over the global variable x. Therefore, when x is printed inside the function, it prints 5; when it is printed outside the function, it prints 10.

Q33. What is an ordered dictionary?

Ans. An ordered dictionary, also known as an OrderedDict, is a subclass of the built-in Python dictionary class that maintains the order of elements in which they were added. In a regular dictionary, the order of elements is determined by the hash values of their keys, which can change over time as the dictionary grows and evolves. An ordered dictionary, on the other hand, uses a doubly linked list to remember the order of elements, so that the order of elements is preserved regardless of how the dictionary changes.

Q34. What is the difference between return and yield keywords?

Ans. Return is used to exit a function and return a value to the caller. When a return statement is encountered, the function terminates immediately, and the value of the expression following the return statement is returned to the caller.

Yield, on the other hand, is used to define a generator function. A generator function is a special kind of function that produces a sequence of values one at a time instead of returning a single value. When a yield statement is encountered, the generator function produces a value and suspends its execution, saving its state for later.

Q35. What are lambda functions in Python, and why are they important?

Ans. Python supports the lambda function, which is a small anonymous function. You can use lambda functions when you don’t want to define a function using the def keyword.

Lambda functions are useful when you need a small function for a short period of time. They are often used in combination with higher-order functions, such as map(), filter(), and reduce().

Here’s an example of a lambda function in Python:

```
a = lambda x: x + 10 a(5)
```


In this example, the lambda function takes one argument (x) and adds 10 to it. The lambda function returns the result of this operation when it is called.

Lambda functions are important because they allow you to create small anonymous functions in a concise way. They are often used in functional programming, a programming paradigm that emphasizes using functions to solve problems.

Q36. What is the use of the 'assert' keyword in Python?

Ans. In Python, the assert statement is used to test a condition. If the condition is True, then the program continues to execute. If the condition is False, then the program raises an AssertionError exception.

The assert statement is often used to check the internal consistency of a program. For example, you might use an assert statement to check that a list is sorted before performing a binary search on the list.

It's important to note that the assert statement is used for debugging purposes and is not intended to be used as a way to handle runtime errors. In production code, you should use try and except blocks to handle exceptions that might be raised at runtime.

Q37. What are decorators in Python?

Ans. In Python, decorators are a way to modify or extend the functionality of a function, method, or class without changing their source code. Decorators are typically implemented as functions that take another function as an argument and return a new function that has the desired behavior.

We can use the decorator function by adding it just before the function we are applying using @decorator_function syntax.

Q38. What are built-in data types in Python?

Ans. The 5 built-in data types in Python are: None Type, Numeric Types (int, float, complex, bool), Sequence Types (list, tuple, range, str), Map Type (dictionary), and Set types (set, frozenset).

Q39. What's the difference between a set and a frozenset?

Ans. A frozenset is an immutable variant of a set, like how a tuple is an immutable variant list.

Q40. Where can we use a tuple instead of a list?

Ans. We can use tuples as dictionary keys as they are hashable. Since tuples are immutable, it is safer to use if we don't want values to change. Tuples are faster and have less memory, so we can use tuples to access only the elements.

Q41. Is removing the first item or last item takes the same time in the Python list?

Ans. No, removing the last item is O(1), while removing the first item is O(n)

Q42. How can we remove any element from a list efficiently?

Ans. We can use a deque from the collections module, which is implemented as a doubly linked list, to remove elements faster at any index.

Q43. What is negative indexing?

Ans. Python sequence data types can be accessed with both positive and negative numbers. With negative indexing, -1 refers to the last element, -2 refers to the penultimate element, and so on.

Q44. Why do floating-point calculations seem inaccurate in Python?

Ans. While representing floating point numbers like 2.5 is easier in the decimal system, representing the same in a binary system needs a lot of bits of information. Since the number of bits is limited, there can be rounding errors in floating-point calculations.

Q45. What is docstring in Python?

Ans. Docstring is the short form for documentation string. It is a type of string used to document how a function or a class works, its various input parameters, and any examples for the usage of the function or class. It is enclosed by triple quotes(“”” “””). A well-written docstring helps people to understand the usage of the function or class.

Q46. What are *args and* *kwargs in Python?

Ans. *args and* *kwargs are syntax used for denoting variable length arguments and variable length keyword arguments, respectively. Here *and* * represents the syntax, while the args and kwargs are words used by convention. We can also use other words.

Q47. What are generators in Python?

Ans. A generator in Python returns an iterator that produces sequences of values one at a time. We use the yield keyword to produce values.

Q48. What is the use of generators in Python?

Ans. Since the generator doesn’t produce all the values at the same time, it saves memory if we use the generator to process the sequence of values without the need to save the initial values.

Q49. How can we iterate through multiple lists at the same time?

Ans. We can use zip() function to aggregate multiple lists and return an iterator of tuples where each tuple contains elements of different lists of the same index.

Q50. What are the various ways of adding elements to a list?

Ans. Here are the various ways of adding elements to a list:

1. We can use insert() to add a given index to the existing list.
2. We can use append() to add at the end of a list a single item.
3. We can use extend() to add each element of an iterable(list, tuple, or set) separately at the end of the list.
4. We can also use the + operator to concatenate two lists, similar to extend, but it works only with one list to another list but not one list to another tuple or set.

Q51. Write a program to check whether a number is prime or not.

Ans.

```
from math import sqrt def prime_or_not(number): for i in range(2, int(sqrt(number)) + 1): if number % i == 0: return 0 return 1
```

Q52. What is the time complexity of the above program, and is there a faster way to do it?

Ans. It has a time complexity of O(sqrt(n)). We can do it faster with the sieve of Eratosthenes.

Q53. What are the ways to swap the values of two elements?

Ans. One way is by using a third variable

```
temp = a a = b b = temp
```

In Python, we can also do it without using the third variable

a, b = b, a

Q54. Write a program in Python to return the factorial of a given number using recursion.

Ans.

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

Q55. Is there a way to calculate factorial faster than above?

Ans. We can use the divide and conquer algorithm technique to calculate faster. It is implemented in a built-in math library with math.factorial().

Q56. How do you find the minimum value in a list with a lambda function?

Ans.

```
from functools import reduce reduce(lambda x, y: x if x < y else y, b)
```

This will give the minimum value in the list.

Q57. Write a code to convert a list of characters to a string of characters separated by a comma.

Ans.

```
‘,’.join(list)
```

Q58. Write a code to select only odd numbers using list comprehension.

Ans.

```
[i for i in range(n) if i%2 != 0]
```

Q59. What is the difference between del and remove on lists?

Ans. ‘Remove’ removes the first occurrence of a given element. ‘Del’ removes elements based on the given index.

Q60. Write a code to get the minimum value in a dictionary.

Ans.

```
dict_[min(dict_.keys(), key=(lambda k: dict_[k]))
```

Q61. Write a program to return the mean value of a tuple of tuples.

Ans.

```
def mean_tuple(numbers): result = [sum(x) / len(x) for x in zip(*numbers)] return result
```

Q62. What is the use of self in Python?

Ans. Self is used to represent the instance of the class. With this, we can access the attributes and methods of the class with an object. It binds the attributes of the object with the given arguments. Self is not a keyword. We can also use any other non-keyword though it is better to follow convention.

Q63. What are the different types of variables in Python OOP?

Ans: The 3 different types of variables in Python OOP (object-oriented programming) are:

1. Class variables: They are defined inside the class but outside other methods and are available to access for any instance of the class.
2. Instance variables: They are defined for each instance of the class separately and accessible by that instance of the class.
3. Local variables: They are defined inside the method and accessible only inside that method.

Q64. What are the different types of methods in Python OOP?

Ans: The 3 different types of methods in Python OOP are:

1. Class methods: They can access only class variables and are used to modify the class state.
2. Instance methods: They can access both class and instance variables and are used to modify the object (instance of a class) state.
3. Static methods: They can’t access either class or instance variables and can be used for functions that are suitable to be in class without accessing class or instance variables.

Q65. What is inheritance, and how is it useful?

Ans. With inheritance, we can use the attributes and methods of the parent class in the child class. We can also modify the parent class methods to suit the child class.

Q66. What are access specifiers?

Ans. We can use *(underscore) or* _(double underscore) to denote protected and private variables. They can be used to restrict access to the attributes of the class.

Q67. In numpy, how is array[:, 0] is different from array[:[0]]?

Ans. array[:, 0] returns 1 st column as 1D array.

array[:, [0]] returns 1st columns as multi-dimensional array.

Q68. How can we check for an array with only zeros?

Ans. We can use the size method of the array to check whether it returns 0. Then it means the array can contain only zeros.

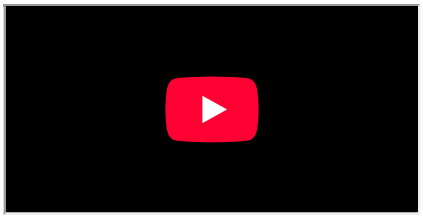
Q69. How can we concatenate two arrays?

Ans. We can use concatenate method.

```
import numpy as np a = np.array([[10, 4], [8, 12]]) b = np.array([[15, 6]]) c = np.concatenate((a, b), axis=0)
```

The result will be
[[10, 4]
[8, 12]
[15, 6]]

Also you can Watch the Video for Python Interview Questions and Answers



Q70. How can we check for the local maxima of an array?

Ans. A local maxima is a point greater than both of its neighbors from either side. In a given array, the following code can produce local maxima points.

```
np.where((arr[1:-1] > arr[0:-2]) * (arr[1:-1] > arr[2:]))[0] + 1
```

Q71. What's the difference between split() and array_split()?

Ans. The split() function is used to split an array in n number of equal parts. If it is not possible to split it into an equal number of parts, split() raises an error. On the other hand, array_split() splits an array into n unequal parts.

Q72. Write code to insert commas between characters of all elements in an array.

Ans.

```
resulted_arr = np.char.join(",", array)
```

Q73. How can you add 1 to all sides of an existing array?

Ans. We can use the pad method of numpy.

```
New_arr = np.pad(existing_arr, pad_width=1, mode='constant', constant_values=1)
```

Q74. How can we swap axes of a numpy array?

Ans. We can use the swapaxes method.

```
np.swapaxes(arr,0,1)
```

Q75. How to get the indices of n maximum values in a given array?

Ans. argsort() method returns indices that can sort the array.

```
array.argsort( ) [ -N: ][: : -1]
```

This gives the indices of n maximum values.

Q76. What is categorical data in pandas?

Ans. When a variable takes a limited number of possible values, we can use category datatype for that variable which is internally represented with numbers, so faster to process.

Q77. How can we transform a true/false value to 1/0 in a dataframe?

Ans.

```
df["column"] = df["column"].astype(int)
```

Q78. How are loc() and iloc() different?

Ans. loc() is used to access the rows or columns using labels, while iloc() is used to access using position-based indexing.

Q79. How do you sort a dataframe by two columns?

Ans.

```
df.sort_values(by=['col1', 'col2'])
```

Q80. Find the row which has the maximum value of a given column.

Ans. We can use idmax method for that.

```
df[‘column’].idxmax()
```

This returns the row index with the maximum value of the column specified.

Q81. How can you split a column which contains strings into multiple columns?

Ans.

```
df[‘column’].str.split(pat=” ”, expand=True)
```

Here we can specify the pattern by which to split the column with the pat argument.

Q82. What is the difference between map() and applymap()?

Ans. While they can be used for elementwise operations, map() is used for series, while applymap() is used for dataframes.

Q83. How can we convert the True values of a query to False and vice-versa?

Ans. We can use the tilde(~) operator to convert True values to False and vice versa.

Q84. How can we find a change in percentage from one row to another?

Ans. We can use the following code to find the percentage change between the previous row to the current row.

```
df.pct_change(periods=1)
```

Q85. How can we find the coefficient of variance for data frame columns?

Ans. Since coefficient of variance is standard deviation divided by mean, we can use df.std()/df.mean()

Q86. How can we remove the leading whitespace for a string?

Ans. We can use the .lstrip() method to remove the leading whitespace for a string.

Q87. What is enumerate() in Python?

Ans. enumerate() in Python iterates a sequence and returns the index position and its corresponding value.

Q88. What are deepcopy and shallowcopy?

Ans. Deepcopy copies the contents of the object to another location. Changes made in one object do not affect the other. In shallow copy, only the reference is copied. So, changes made in one affect the other object.

Q89. What is a callable object in Python?

Ans. An object which can invoke a process is a callable object. It uses the **call** method. Functions are examples of that. Callable objects have () at the end, while non-callable methods don't have () at the end.

Q90. How can we find unique elements and value counts of a list using Numpy?

Ans. We can use numpy.unique(list, return_counts=True)
This will return values and their counts in two arrays.

Q91. What is the difference between indexing and slicing in NumPy?

Ans. Indexing creates an index, whereas slicing makes a shallow copy. Different types of indexing are possible, but there is no slicing category.

Conclusion

Phew! We've finally reached the end of this very comprehensive article. Hope these Python **interview questions** and answers have been beneficial to you. I'm sure you've all learned a few things from this and are now confident about your next interview. These Python coding Interview questions also act as Python tutorials for freshers and intermediate-level programmers and cover functions that most Python developers frequently use. We have cover everything in this article of Python interview Questions.

Hope you like the article! Python coding interview questions often focus on basic topics like syntax, data structures, and algorithms. Knowing common Python coding questions can really help you do well in interviews. Practicing Python interview questions and answers is a great way to get ready for technical tests. Understanding these Python interview questions will make you feel more confident and improve your problem-solving skills during the interview.

Reading List

You can go through the following articles to learn and practice python topics for your next Python interview Questions or upcoming python viva:

- 30+ MCQs on Basic Python with Answers
- 30+ Multiple-Choice Questions on Python Variables
- 30+ Multiple-Choice Questions on Python Data Types
- 30+ Multiple Choice Questions on Python Syntax and Semantics
- 30+ MCQs on Python Operators and Expressions
- 30+ MCQs on Python Control Flow
- 30+ MCQs on Python Functions
- 30+ MCQs on Python Modules and Packages
- 30+ MCQs on Python Error Handling
- 30+ MCQs on Python File I/O
- 30+ MCQs on Python String Manipulation
- 30+ MCQs on Python Tuple Manipulation
- 30+ MCQs on Python List Manipulation
- 30+ MCQs on Python Dictionary Manipulation
- 30+ MCQs on Python Sets and Sets Operations
- 30+ MCQs on Python OOPs Concepts
- 30+ MCQs on Python Inheritance and Polymorphism
- 30+ MCQs on Python Abstraction and Encapsulation
- 30+ MCQs on Python Special Methods
- 30+ MCQs on Python Recursion
- 30+ MCQs on Python Lambda Functions
- 30+ MCQs on Python Exception Handling
- 30+ MCQs on Python Regular Expression
- 30+ MCQs on Python Map, Filter and Reduce Functions
- 30+ MCQs on Python Date and Time Handling
- 30+ MCQs on Database Interaction with Python

Article Url - https://www.analyticsvidhya.com/blog/2022/07/python-coding-interview-questions-for-freshers/

saumyab271